CityNavAgent: Zero-shot Vision-and-Language Navigation in Outdoor Environment with LLM-empowered Agent

Anonymous ACL submission

Abstract

A fundamental objective within the field of AI research is the development of intelligent agents capable of engaging in natural language interactions with humans, comprehending their surroundings, and executing tasks in real-world settings. One of the most representative tasks, named vision-and-language navigation (VLN), requires the agent to navigate to a specific location following human language instruction. Although existing VLN agents achieved notable results in indoor settings, they mostly rely on navigation graphs to select the next potential waypoint, which is generally impractical in continuous 3D spaces in urban environments. In this work, to address it, we propose **CityNavAgent**, a large language model-empowered agent for VLN in continuous outdoor environments. To track the navigation task, we decompose the task into hierarchical sub-goals, inducing the agent to reach the target progressively. We also leverage the commonsense knowledge of LLM to facilitate fine-grained action reasoning. Extensive benchmark experiments show that our method achieves state-of-the-art performance with significant improvement. Further experiments demonstrate the effectiveness of different modules of CityNavAgent for zero-shot VLN in continuous city environments.

1 Introduction

Visual-and-language navigation is a fundamental task wherein the agent navigates to a landmark or a spot following language instructions (Anderson et al., 2018; Gu et al., 2022; Gao et al., 2024). Such a task is easy for humans but challenging for AI agents, especially in open and dynamic scenarios. Specifically, the task requires the agent to have strong capacities in both vision and language understanding, along with the ability to execute complicated actions in interactive environments. As a task that measures the intelligence level of human-like perception and decision-making in embodied environments (Shridhar et al., 2020; Wen et al., 2023), VLN has attracted much attention in embodied artificial intelligence.

Most existing methods (Anderson et al., 2018; Zhou et al., 2024; Chen et al., 2024; Hong et al., 2021; Long et al., 2024; An et al., 2023; Chen et al., 2022) consider the task of indoor VLN, and rely on a pre-defined topological map, and then navigation can be regarded as a graph-search task. In these works, the agent teleports from one node to its neighbors without motion error until it reaches the node closest to the target. However, when it comes to VLN in the outdoor environments, the space is continuous and thus those methods suffer from the absence of nav-graph and navigation error accumulation. That is, although existing solutions have achieved some progress for indoor tasks, they fail in outdoor scenarios.

Specifically, for the outdoor VLN tasks, the agent has to accurately follow the language instruction to navigate in continuous space and an open environment, which has two folds of critical challenges.

- The first challenge is **fine-grained action reasoning based on coarse instructions** since the environment is diverse. In outdoor environments, different from indoor tasks, the task descriptions in the instructions are coarse-grained (especially for the diverse elements in the environment). For example, when given the instruction *"turn left and fly to ..."*, the agent has to rely entirely on its scene and task understanding capacity to translate the coarse descriptions into detailed action commands.

- The second challenge is **the long-term path following** since the environment is on a large scale. The space of the outdoor environments is far larger than the traditional indoor environment, which thus requires the agents to face very long-term planning and navigation. Further, due to

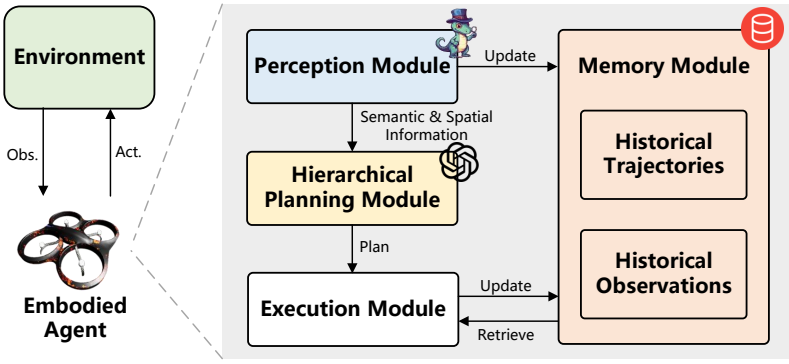

Figure 1: The overall workflow of the proposed CityNavAgent.

imperfect action reasoning and execution, navigation errors always exist. Those errors will accumulate significantly in long-term navigation without intermediate checks, causing the agent to deviate from the correct path and get lost in an unexplored area.

In this work, to address the above challenges, we propose CityNavAgent, an embodied agent empowered by large language models. To address the first challenge, CityNavAgent leverages LLM's commonsense knowledge to reason for the semantic and spatial relationships between targets and surroundings. Based on these relationships and the action descriptions in the instruction, the agent could first reason the waypoint and then plan fine-grained actions. To address the second challenge, CityNavAgent leverages LLM's hierarchical task-planning capacity to decompose the original task into multi-level sub-goals which are landmarks on the path. Navigating to these landmarks is easier than the original task and the agent could track the correct path by achieving these sub-goals step by step. Moreover, a memory module is constructed to enhance the agent's embodied learning abilities from experience. CityNavAgent could retrieve the optimal path from the memory to the location visited before.

It's important to note that the entire navigation pipeline is realized in a zero-shot manner, which is merely explored in the outdoor VLN task. We evaluate the proposed with extensive baselines, of which the results show CityNavAgent's effectiveness in urban VLN tasks. CityNavAgent exhibits superior performance in both success rate (SR) and success weighted by path length (SPL), outperforming state-of-the-art (SOTA) approaches by $19-30\%$ and $10-20\%$. The main contributions of this work are as follows:

- To the best of our knowledge, we take the pioneering step to investigate the zero-shot VLN task in an open outdoor environment, which is a fundamental task for embodied intelligence.
- We design a large language model-based agent, CityNavAgent, which leverages commonsense knowledge for fine-grained action reasoning and progressively achieves the target by navigating to the landmarks on the path step by step.
- We conduct extensive experiments on benchmark evaluations, and the results show that our method achieves SOTA performance across multiple urban scenarios, demonstrating promising improvements over existing baselines.

2 Problem Formulation

VLN is a challenging task that requires an agent to interpret human instructions and navigate to the target location in unseen environments by executing a sequence of actions. Given a natural language instruction $\mathcal{I}$ and the agent's egocentric observation $\mathcal{O}$, the VLN agent has to determine a sequence of action to reach the target location p_d in a continuous 3D space. At each action step t, the agent follows a policy π taking current observation o_t and instruction $\mathcal{I}$ as input to predict the next action a_t and move to location p_t by its kinematic model $\mathcal{F}$, which is given by:

$$p_t = \mathcal{F}(p_{t-1}, \pi(o_t, \mathcal{I})), \tag{1}$$

Given a sequence of action, the agent reaches a final position, and the probability of reaching the target P_s is

$$P_s = P(||\mathcal{F}(\pi(p_0, \mathcal{O}, \mathcal{I})) - p_d|| < \epsilon), \tag{2}$$

where $||\cdot||$ is the Euclidean distance and ϵ is the threshold that indicates if the target is reached.

Thus, the goal of VLN is to find a policy π^* that maximizes the success rate, given by:

$$\pi^* = \text{argmax}_\pi P_s. \quad (3)$$

3 CityNavAgent

In this section, we present the workflow of the proposed CityNavAgent for zero-shot aerial VLN in urban environments. As shown in Figure 1, the CityNavAgent framework comprises four modules: a perception module, an LLM hierarchical planning module, a memory module, and an execution module. The agent first extracts the structured semantic and spatial information from its panoramic observation via a multimodal foundation model. Then, the hierarchical planning module leverages LLM to decompose the original task into sub-goals and reasoning for the plan through its task planning and commonsense reasoning capacities. The sub-goal plans are fed into the execution module for low-level motion control. The memory module stores a topological graph constructed by historical trajectories and facilitates the agent's navigation to the previously visited target.

Similar to recent work (An et al., 2024; Liu et al., 2023), we assume that the agent has access to the ground-truth pose provided by the simulator for mapping and control. This paper proceeds by introducing the perception module in §3.1, followed by LLM hierarchical planning in §3.2, and the presentation of the memory module in §3.3.

3.1 Scene Perception Module

To perceive the open urban environment, the aerial agent has to recognize the objects in the scene and extract their semantic and spatial information. At each action step, the perception module captures the first-view panorama of the agent to ensure comprehensive scene information. Subsequently, structured semantic and spatial information are extracted from the panorama for the reasoning and decision-making processes of the LLM planner.

Scene Semantic Information To extract scene semantic information, the agent has to recognize and detect objects in the observation views. Since the urban environment contains dense open-vocabulary objects, we employ GPT-4V (Achiam et al., 2023) as an open-set image captioner to combine with GroundingSAM (Ren et al., 2024). This integration enables open-vocabulary captioning and segmentation based on arbitrary images. As shown in Figure 2, GPT-4V is prompted to describe visible objects in each observation view. The object description and RGB images are fed into GroundingSAM for dense detection and segmentation. GroundingSAM outputs bounding boxes and semantic masks whose similarity scores with object descriptions are higher than a threshold. Note that the object descriptions by GPT-4V are sometimes hallucinations, we set a slightly high threshold ($\eta = 0.3$) for GroundingSAM to mitigate this issue.

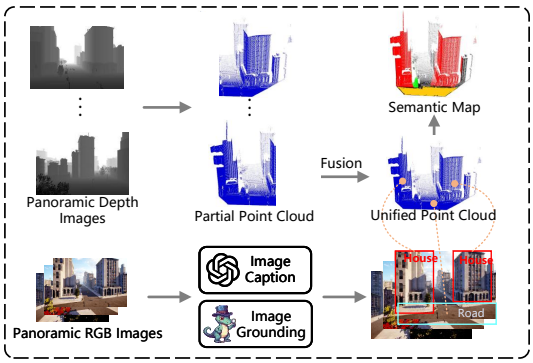

Figure 2: The perception module of the agent.

Scene Spatial Information Spatial information is represented as a local point cloud map that combines with semantic information for the agent's waypoint planning. As shown in Fig.2, with the camera intrinsic matrix K and agent's pose (p, α), pixels p_o of the depth image of observation view can be projected to a 3D point cloud in the world coordinate system. The final point cloud can be obtained by merging the corresponding point cloud of each viewpoint. Note that RGB-D images in each observation view are well-aligned, meaning that the extracted semantic masks of RGB images have their counterparts in depth images as well as in the point cloud. To this end, a local map containing both semantic and spatial information is constructed.

3.2 LLM Hierarchical Planner

3.2.1 LLM High-level Planner

Since the aerial VLN task usually involves a long-range decision-making process (Liu et al., 2023; Chen et al., 2019), directly assigning the entire navigation task to the agent can make it challenging to accurately align the task status with the instruction and track the task progress. A more practical approach is breaking down the overall task into a sequence of achievable sub-goals. By addressing these sub-goals step by step, the final goal can be achieved progressively. To this end, we propose

the LLM high-level task planner that parses the free-form instruction and extracts a sequence of landmarks on the path to the destination. This landmark sequence serves as sub-goals for the agent to navigate from one to another. We employ a standard LLM GPT-4V (Achiam et al., 2023) to extract phrases of common objects or landmarks in urban environments. We introduce explicit reasoning to enable the LLM to analyze the path described in the instruction before generating the sequence. We present a simple prompt as follows (more details in Appendix A):

> *You need to extract a landmark sequence from the given instruction. The sequence order should be consistent with their appearance order on the path.*
> *Your output should be in JSON format and must contain two fields: "Landmark sequence" and "Thought".*

3.2.2 LLM Low-level Planner

After high-level task planning and obtaining a sequence of sub-goals, the low-level planner employs the LLM to further decompose these sub-goals into more achievable steps for the agent. The key idea is to leverage the commonsense knowledge of the LLM to reason for the visible region most pertinent to the invisible sub-goal in the current panorama. This region is referred to as the region of interest (ROI) in this paper. For example, if the agent only sees the buildings and a road in the current view while its sub-goal is the traffic light, by commonsense reasoning, the next ROI it should go is the road. To induce such reasoning capacity, we design a prompt that comprises the original navigation instruction, scene semantic information, and current sub-goals. Its template is as follows (more details in Appendix A):

> *The navigation instruction is: .... Your next navigation subgoal is: ... Objects or areas you observed: ...*
>
> *Based on the instruction, next subgoal, and observation, list 3 objects most pertinent to the subgoal or you will probably go next from your [Observed Objects]. Output should be in descending order of probability.*

The prompt is fed into the LLM to obtain three potential ROIs. By default, we select the ROI with the highest probability for the next waypoint planning. If the most probable ROI is hallucination, the second ROI is selected, and so on. If all ROIs do not exist in current observations, the agent stops and the task is labeled as failure.

3.2.3 Waypoint Planner

Based on the reasoned ROI and constructed semantic map, the precise waypoint can be computed by averaging the corresponding points in the map. To avoid the collision, the agent stops moving if the average depth value of the front-view depth image is below a threshold. Given the predicted coordinate of the waypoint, the motion planner will output an action sequence by the greedy algorithm for the agent to reach the waypoint.

3.3 Memory Module

Since the agent sometimes navigates to the target or passes landmarks it has visited before, we designed a memory module storing historical trajectories to fully exploit the agent's experience.

Memory Graph Construction During the navigation, the agent records its waypoint and panoramic observations at each action step in an undirected topological graph $G(N, E)$. Each graph node $n \in N$ encapsulates both the coordinates of the waypoints and their associated panoramic observations. An edge $e \in E$ is established between two nodes only when their respective waypoints are adjacent. An edge's weight is determined based on the spatial distance between the connected waypoints. Every historical navigation path can be represented as such a topological graph to form an entire memory graph.

Memory Graph Merge Since the agent continues to generate new topological graphs as it moves, the memory module updates the memory graph by merging the new topological graph. Firstly, the node sets and edge sets of the two graphs are merged directly. Then, if the new graph and the memory graph are "close" enough, a new edge is generated to link them. We calculate the distance between every pair of points in the two graphs. If the distance between any pair of points is less than a threshold H=15m, it is inferred that these points are adjacent, prompting the addition of a new edge to the merged graph.

Graph Search for Navigation When the agent reaches a waypoint in the memory graph, the agent

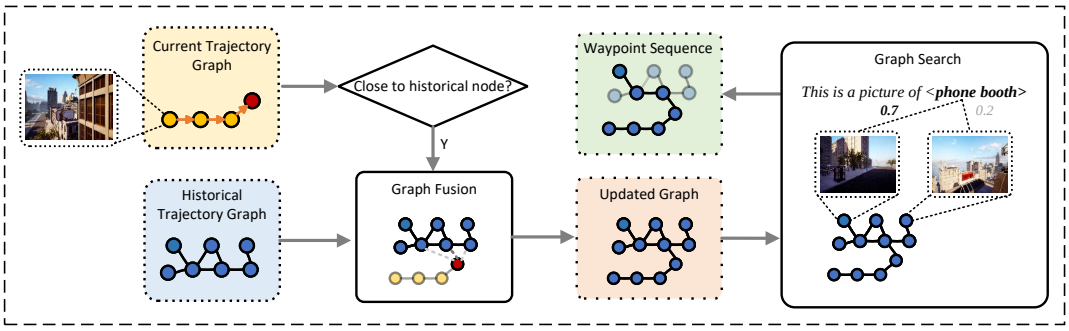

Figure 3: The illustration of the memory module. The agent stores its visual observation in the current trajectory graph. Once it reaches the node in the historical trajectory graph, the agent fuses these two graphs and searches for a path with the highest probability to the target. The probability is measured by a similarity score between the landmark phase in the instruction and visual observation stored in the node.

determines a path in the graph with the highest likelihood of traversing all landmarks. We adopt the graph search algorithm in LM-Nav (Shah et al., 2023). Given a memory graph $G(N, E)$ and a sequence of landmarks L, the goal is to determine a sequence of waypoints W in the graph that maximizes the possibility of traversing these landmarks. A scoring function Q is defined to weigh the reward of the agent's movement between adjacent nodes. Then, a graph search method integrated with the Dijkstra algorithm (Dijkstra, 2022) is designed for calculating the best waypoint sequence W^*. For more details, readers please refer to Alg.1 in Appendix B.

4 Experiments

4.1 Experimental Setup

Datasets We evaluate CityNavAgent on AirVLN (Liu et al., 2023), a simulator with city environments. Due to the coarse original instructions, we further enrich some of their instructions as our test set. To fully evaluate the capacity of exploration and exploration of the agent, the navigation task comprises the first unseen path and the following visited path in the memory. The agent spawned at a random location first explores the path to the memory and uses the memory graph to navigate to the target. The VLN tasks are categorized into three difficulty levels based on their trajectory length: easy tasks traverse two landmarks, normal tasks pass through three to four landmarks, and hard tasks involve navigating past five or more landmarks.

Metrics We adopt standard metrics (Zhou et al., 2023) to measure navigation performance. Navigation Error (NE): average distance between the agent's stop location and the target. Success Rate (SR): success ratio of the navigation task. Success weighted by Path Length (SPL): SR normalized by path length. The task is successfully completed if NE is within 20 meters which is a standard threshold defined in AirVLN.

Baselines We choose two mainstream types of continuous VLN baselines: **statical-based** and **learning-based** methods. We also compare CityNavAgent with **zero-shot LLM-based** methods.

- **Statistical-based Methods:** Agents predict the next action by the statistical characteristic of the dataset. We use **random sample (RS)** that agents uniformly select an action from the action space at each step and **action sample (AC)** that agents sample actions according to the action distribution of the dataset as our baselines.
- **Learning-based Methods:** We choose two typical learning-based methods **Seq2Seq** (Anderson et al., 2018) and **CMA** (Krantz et al., 2020) as our baselines. Consistent with AirVLN, we train these methods in an offline teacher-forcing training manner.
- **Zero-shot LLM-based methods:** We use NavGPT (Zhou et al., 2024), MapGPT (Chen et al., 2024), VELMA (Schumann et al., 2024) and LM-Nav (Shah et al., 2023) as our baselines. Since the first three methods rely on the navgraph which is unavailable in our scenario, we replace their graph-based action space with the CityNavAgent's mentioned in §3.2.3. All methods use GPT-4V for scene captioning and action reasoning in the evaluation.

4.2 Overall Performance (RQ1)

In Table 1, we compare the overall performance of CityNavAgent with SOTA navigation methods, from which we make the following observations.

Table 1: Overall performance comparisons under different task difficulties.

Method	Easy			Normal			Hard			Mean		
	SR↑	SPL↑	NE↓	SR↑	SPL↑	NE↓	SR↑	SPL↑	NE↓	SR↑	SPL↑	NE↓
RS	0.0	0.0	85.6	0.0	0.0	127.9	0.0	0.0	164.9	0.0	0.0	129.6
AC	0.0	0.0	242.2	0.0	0.0	315.6	0.0	0.0	263.2	0.0	0.0	290.4
Seq2seq	0.0	0.0	201.1	0.0	0.0	359.4	0.0	0.0	713.3	0.0	0.0	398.5
CMA	0.0	0.0	152.1	0.0	0.0	317.6	0.0	0.0	286.8	0.0	0.0	278.3
NavGPT	0.0	0.0	79.8	0.0	0.0	126.1	0.0	0.0	177.6	0.0	0.0	127.2
MapGPT	0.0	0.0	97.5	5.6	2.9	135.2	0.0	0.0	165.5	3.3	1.5	133.7
VELMA	0.0	0.0	76.5	0.0	0.0	141.7	0.0	0.0	192.4	0.0	0.0	138.0
LM-Nav	15.4	13.7	123.1	22.2	18.1	124.3	33.3	**28.1**	**114.2**	23.6	19.2	119.4
CityNavAgent	**25.0**	**21.3**	**74.7**	**27.8**	**23.3**	**93.4**	**33.3**	26.3	121.5	**28.3**	**23.5**	**95.1**

The vast action space exhibits great challenge for baselines. Both the SR and SPL of statistical-based and learning-based methods are close to 0, and the NE of these two methods are significantly high. This result illustrates that the long-term outdoor VLN involves an extremely large action space and even learning-based methods failed to traverse the whole space to find optimal paths.

LLM fails to be a low-level action planner in continuous space. Zero-shot LLM-based methods that leverage LLM for scene understanding and motion planning also have 0 SR and high NE. We attribute this result to the fact that general-purpose LLMs like GPTs fail to reason for the low-level action due to the lack of domain-specific knowledge. Besides, directly predicting a long-term action sequence without intermediate checks will cause errors that accumulate along the path, leading to great deviation from the target.

Our CityNavAgent method significantly outperforms previous SOTAs on all metrics. Specifically, CityNavAgent improves SR by approximately 28.3% and 4.7% over VELMA and LM-Nav, respectively. This demonstrates the critical importance and navigation efficiency of semantic map-based waypoint planning for continuous outdoor navigation. Furthermore, in terms of SPL, our approach achieves improvements of 23.5% and 2.5% compared to VELMA and LM-Nav, respectively, indicating that CityNavAgent predicts navigation paths that are closer to the ground truth. The lowest error rate shows that even for those failed cases, our method still stops relatively close to the target. For easy and normal tasks, our method consistently surpasses the baseline by at least 5% in SR and SPL. For hard tasks, our method still has a similar performance to the best method.

Table 2: Effectiveness of different modules in CityNavAgent. MG and SM represent the memory graph and semantic map, respectively.

Modules	SR↑	SPL↑	NE↓
w/o MG	11.7	9.1	206.1
w/o SM	23.6	19.2	119.4
w/ GPT-3.5	23.3	16.1	98.9
w/ GPT-4V (ours)	28.3	23.5	95.1

Table 3: Comparison of waypoint predictor under different scenarios.

	Inputs	$\lvert\triangle\rvert$	$d_{rel}\downarrow$	$d_C\downarrow$	$d_H\downarrow$
Outdoor	RGBD	1.66	0.88	6.46	5.16
	RGB	1.59	0.88	**6.40**	**5.15**
	Depth	1.60	0.88	6.48	5.26
Indoor	RGBD	1.40	-	1.05	2.01
	RGB	1.38	-	1.08	2.03
	Depth	1.39	-	**1.04**	**2.01**

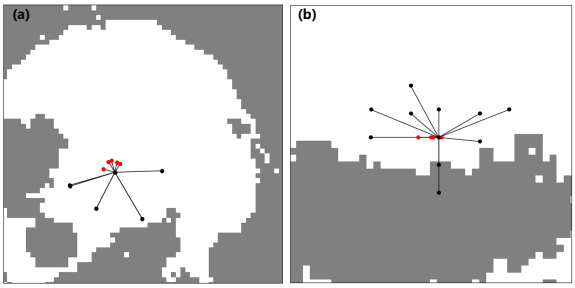

Figure 4: The qualitative result of CWP. From left to right are the top-down view and the front-facing view. ***Red*** and ***black*** circles denote predicted waypoints and reference waypoints on real trajectories.

4.3 Ablation Study (RQ2)

Effect of semantic map-based exploration. To evaluate the effectiveness of semantic map-based waypoint prediction, we substitute this module with

a random walk strategy. As shown in Table 2, the agent without the semantic map (second row) suffers a 4.7% and 4.3% drop in SR and SPL, and a 25.6% increase on NE with CityNavAgent (last row). This result reveals that the semantic map extracts structured environmental information, facilitating the LLM in commonsense reasoning so that the agent navigates to the region or objects that are more relevant to the navigation task. Consequently, the accuracy and efficiency of the navigation is improved.

Effect of memory graph-based exploitation. In this case, we omit the memory module during the navigation and replace the graph search algorithm with the random walk strategy. Presented in the first row in Table 2, the lack of memory graph results in a 16.6%, 14.4% decrease in SR, SPL, and a 116.7% increase in NE over CityNavAgent. Moreover, the memory graph has a more noticeable impact on the agent's navigation performance compared with the semantic map. This indicates that the memory graph effectively prevents the agent from falling into dead ends or engaging in blind exploration in long-distance outdoor navigation scenarios, thereby ensuring the stability of navigation performance.

Effect of different LLMs. We also evaluate the effectiveness of different LLMs for commonsense reasoning in hierarchical planning (§3.2). Although CityNavAgent with GPT-3.5 demonstrates a SOTA performance, replacing GPT-3.5 with GPT-4 which has enhanced reasoning capability results in further performance improvement, *e.g.*, 5.0% and 7.4% increases in SR and SPL, respectively. We attribute this improvement to the fact that GPT-4 has a lower hallucination rate and stronger reasoning ability. Thus, it generates more contextually appropriate responses based on the semantic map and navigation instruction to facilitate the agent in exploring areas most relevant to the target.

4.4 Effectiveness of Indoor Waypoint Prediction (RQ3)

To evaluate the effectiveness of the waypoint prediction method CWP (Hong et al., 2022) in previous continuous VLN methods (An et al., 2024; Koh et al., 2021; Wang et al., 2023a,b; Krantz and Lee, 2022; Wang et al., 2024), we compare the predicted waypoint with target waypoints in outdoor environments. We apply waypoint metrics (Hong et al., 2022) to assess the quality of predicted waypoints. These metrics are as follows: $|\triangle|$ measures the difference in the number of target waypoints and predicted waypoints. d_{rel} measures the ratio of average waypoint distance. d_C and d_H are the Chamfer distance and the Hausdorff distance, respectively, commonly used to measure the distance between point clouds. As depicted in Table 3, CWP achieves the best performance in indoor environments with 1.04 d_C and 2.01 d_H while in outdoor environments with 6.4 d_C and 5.15 d_H. This result indicates that although the predicted indoor waypoints by CWP are close to the indoor target waypoints, predicted outdoor waypoints are far from outdoor target waypoints, which is illustrated intuitively in Figure 4(a). We attribute this to the scale difference between indoor and outdoor environments, specifically that the free space in outdoor scenes is much larger than in indoor scenes. Besides, the dimensional difference is another negative factor for CWP. Depicted in Figure 4(b), CWP only predicts waypoints in 2D space and cannot be applied to open urban environments.

4.5 Effectiveness of Commonsense Reasoning and Memory (RQ4)

In this part, we illustrate the qualitative navigation process of CityNavAgent to further illustrate how the commonsense reasoning and memory graph work. As depicted in Figure 5, the agent is spawned at a random location with a navigation instruction. The agent has to explore the ordered landmarks in the instruction based on its visual observation. Thanks to its reasoning capabilities, the agent infers objects in its FOV that are semantically related to landmarks, even when those landmarks are not visually observed. In the given example, the agent tries to find the obelisk in the park which is currently invisible. Hinted by the instruction that the obelisk is in the park, CityNavAgent reasons that the trees probably appear in the park and decides to explore the areas near the trees while LM-Nav is gradually lost due to its lack of exploration ability (first three columns of Figure 5). Once the agent reaches a place visited before, CityNavAgent leverages the memory graph to search for a path to the target while VELMA only relies on LLM for action planning and is trapped in an unfamiliar place (see the last columns of Figure 5).

5 Related Works

Continuous VLN

Krantz et al. (Krantz et al., 2020) introduced R2R-CE tasks that adapt R2R (Anderson et al.,

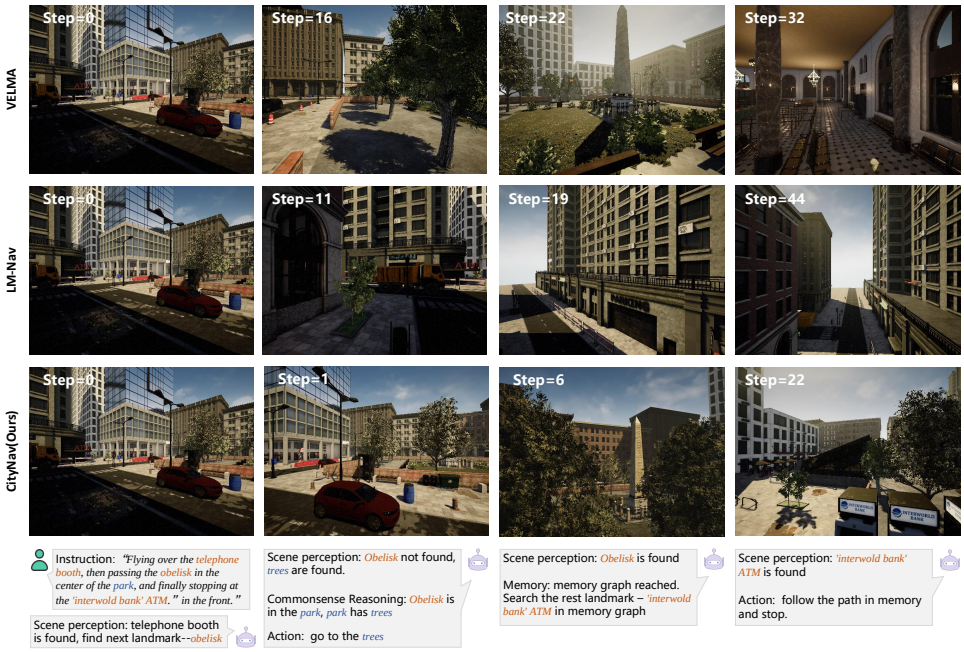

Figure 5: Qualitative result of the navigation process. The first three rows are the first-person view of VELMA, LM-Nav and CityNavAgent during the navigation. The last row is the reasoning process of CityNavAgent. *Orange* and *blue* represents the navigable landmarks in the instruction and common objects that semantically relevant to the landmarks, respectively.

2018) trajectories for continuous environments. Irshad et al. (Irshad et al., 2021) developed the RoboVLN dataset, which provides long-distance navigation in a continuous action space, making VLN tasks more practical and challenging. A popular branch of methods (Krantz et al., 2020; Raychaudhuri et al., 2021; Liu et al., 2023) for continuous VLN applies sequence-to-sequence models to directly predict the agent's low-level action. Some of these methods (Irshad et al., 2022; Chen et al., 2021; Krantz et al., 2021) integrate cross-modal attention for low-level action reasoning with multi-modal inputs. waypoint-based methods such as CWP (Hong et al., 2022), ETPNav (An et al., 2024), Dreamwalker (Wang et al., 2023a), and HNR (Wang et al., 2024), construct online maps and waypoints to narrow down the possible locations of the agent during the navigation. While promising results have been achieved in indoor simulated environments, these methods cannot be directly applied to 3D outdoor scenarios due to differences in scale and dimensionality. The limited 3D outdoor datasets exacerbate the challenges of transferring these learning-based methods to outdoor environments.

Large Pre-trained Model for VLN Recently, there has been a trend toward large foundation model-based methods (Touvron et al., 2023; Chiang et al., 2023; Achiam et al., 2023; Brown et al., 2020) for zero-shot VLN. Pioneering works (Dorbala et al., 2022) use foundation models for scene understanding and instruction interpretation. NavGPT (Zhou et al., 2024) and MapGPT (Chen et al., 2024) utilize LLM's commonsense reasoning ability to select the next viewpoint in the navigraph. A^{2}Nav (Chen et al., 2023) combines GPT-3 (Brown et al., 2020) with a small policy model from high-level decision and low-level planning. For outdoor VLN, VELMA (Schumann et al., 2024) uses CLIP (Radford et al., 2021) and GPT-4 for scene captioning and action reasoning on a discrete graph. LM-Nav (Shah et al., 2023) designed a dynamic programming algorithm integrated with LLM for zero-shot outdoor VLN. Most zero-shot foundation model-based methods rely on the predefined navgraph which is not always available. In this work, we propose CityNavAgent, an agent for outdoor VLN that could navigate in continuous space even without predefined navigation graphs.

6 Conclusion

In this paper, we approach the problem of zero-shot vision-language navigation by proposing an embodied aerial agent, CityNavAgent, which leverages the pre-trained knowledge in large foundation models and historical experience to deal with long-term navigation in urban spaces. The experimental results illustrate the efficacy and robustness of our method from different perspectives.

Limitations

One limitation of our work is that the whole system has not been deployed on a real drone. Though our methods achieve promising results in simulated outdoor environments, low-level motion control problems such as self-pose estimation, control latency, and control errors are not considered in our work. The second is that the agent lacks a backtracking mechanism. CityNavAgent so far only relies on the sub-goal decomposition to track the navigation path.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Dong An, Yuankai Qi, Yangguang Li, Yan Huang, Liang Wang, Tieniu Tan, and Jing Shao. 2023. Bevbert: Multimodal map pre-training for language-guided navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2737–2748.

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. 2024. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12538–12547.

Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee K Wong. 2024. Mapgpt: Map-guided prompting for unified vision-and-language navigation. *arXiv preprint arXiv:2401.07314*.

Kevin Chen, Junshen K Chen, Jo Chuang, Marynel Vázquez, and Silvio Savarese. 2021. Topological planning with transformers for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11276–11286.

Peihao Chen, Xinyu Sun, Hongyan Zhi, Runhao Zeng, Thomas H Li, Gaowen Liu, Mingkui Tan, and Chuang Gan. 2023. A^2 nav: Action-aware zero-shot robot navigation by exploiting vision-and-language ability of foundation models. *arXiv preprint arXiv:2308.07997*.

Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. 2022. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16537–16547.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Edsger W Dijkstra. 2022. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290.

Vishnu Sashank Dorbala, Gunnar Sigurdsson, Robinson Piramuthu, Jesse Thomason, and Gaurav S Sukhatme. 2022. Clip-nav: Using clip for zero-shot vision-and-language navigation. *arXiv preprint arXiv:2211.16649*.

Peng Gao, Peng Wang, Feng Gao, Fei Wang, and Ruyue Yuan. 2024. Vision-language navigation with embodied intelligence: A survey. *arXiv preprint arXiv:2402.14304*.

Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Eric Wang. 2022. Vision-and-language navigation: A survey of tasks, methods, and future directions. *arXiv preprint arXiv:2203.12667*.

Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. 2022. 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*, pages 15439–15449.

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

Muhammad Zubair Irshad, Chih-Yao Ma, and Zsolt Kira. 2021. Hierarchical cross-modal agent for robotics vision-and-language navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13238–13246. IEEE.

Muhammad Zubair Irshad, Niluthpol Chowdhury Mithun, Zachary Seymour, Han-Pang Chiu, Supun Samarasekera, and Rakesh Kumar. 2022. Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 4065–4071. IEEE.

Jing Yu Koh, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. 2021. Pathdreamer: A world model for indoor navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14738–14748.

Jacob Krantz, Aaron Gokaslan, Dhruv Batra, Stefan Lee, and Oleksandr Maksymets. 2021. Waypoint models for instruction-guided navigation in continuous environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15162–15171.

Jacob Krantz and Stefan Lee. 2022. Sim-2-sim transfer for vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pages 588–603. Springer.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. 2020. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 104–120. Springer.

Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. 2023. Aerialvln: Vision-and-language navigation for uavs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15384–15394.

Yuxing Long, Xiaoqi Li, Wenzhe Cai, and Hao Dong. 2024. Discuss before moving: Visual language navigation via multi-expert discussions. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 17380–17387. IEEE.

Tong Qin, Peiliang Li, and Shaojie Shen. 2018. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Sonia Raychaudhuri, Saim Wani, Shivansh Patel, Unnat Jain, and Angel X Chang. 2021. Language-aligned waypoint (law) supervision for vision-and-language navigation in continuous environments. *arXiv preprint arXiv:2109.15207*.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. 2024. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*.

Raphael Schumann, Wanrong Zhu, Weixi Feng, Tsu-Jui Fu, Stefan Riezler, and William Yang Wang. 2024. Velma: Verbalization embodiment of llm agents for vision and language navigation in street view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18924–18933.

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

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hanqing Wang, Wei Liang, Luc Van Gool, and Wenguan Wang. 2023a. Dreamwalker: Mental planning for continuous vision-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10873–10883.

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, Junjie Hu, Ming Jiang, and Shuqiang Jiang. 2024. Lookahead exploration with neural radiance representation for continuous vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13753–13762.

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. 2023b. Gridmm: Grid memory map for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15625–15636.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2023. Empowering llm to use smartphone for intelligent task automation. *arXiv preprint arXiv:2308.15272*.

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

Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. 2023. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pages 42829–42842. PMLR.

A Appendix

A.1 A Prompt Engineering

In this part, we discuss the prompt used in this work. <> denotes the content needs to be filled in the prompting process.

LLM Prompt for High-level Planner The complete prompt is:

Task Planning Prompt

You are a navigation aircraft, and now you need to navigate to a specified location according to a natural language instruction. You need to extract a landmark sequence from the instruction. The sequence order should be consistent with their appearance on the path. Your output should be in JSON format and must contain two fields: "Landmark sequence" and "Thought." "Landmark sequence" is your thinking result comprised of landmark phrases in the instruction. "Thought" is your thinking process. The instruction is < "First, you need to find a stop sign. Then take left and right and continue until you reach a square with a tree. Continue first straight, then right, until you find a white truck. The final destination is a white building." >

The example output is:

"Landmark sequence": [
"stop sign",
"square with a tree",
"white truck",
"white building",
]
"Thought": "The instruction outlines a series of landmarks that need to be followed in order. The first landmark is a 'stop sign,' which I need to locate first. After finding the stop sign, I will navigate to a 'square with a tree' by taking left and right turns. Then, I will continue straight and turn right until I encounter a 'white truck.' Finally, my destination is a 'white building.' I have listed these landmarks in the order they appear in the navigation process."

The Prompt for Low-level Planner The complete prompt for the LLM commonsense reasoning in the low-level planner is:

Commonsense Reasoning Prompt

Based on the instruction, next navigation subgoal, and observation, list 3 objects you will probably go next from your OBSERVED OBJECTS in descending order of probability. *You are a drone and your task is navigating to the described target location!*

Navigation instruction: Start from the 'open' logo, fly forward and pass the yellow 'restaurant' logo in the front road. Then fly forward to the left, pass the circle grassland, and turn right. Finally, stop in front of the glass door with the trash bin and bench.

Your next navigation subgoal: yellow 'restaurant' logo

Objects or areas you observed: building with stairs, road, street lamp, street lamp, 'open' logo, building with stairs

Based on the instruction, next navigation subgoal, and observation, list 3 objects you will probably go next from your OBSERVED OBJECTS in descending order of probability.

A.2 Graph Search Algorithm

The graph search problem is formulated as given a memory graph $G(N, E)$ and a sequence of landmark phrases $L = (\ell_1, \ell_2, \ldots, \ell_n)$ extracted from the language instructions, the goal is to determine a sequence of waypoints $W = (w_0, w_1, \ldots, w_m)$ that maximizes $P(r_L = 1|W, L)$, where $r_L = 1$ indicates that the sequence of the landmarks is traversed successfully. A scoring function $Q(i, w_k)$ is defined to represent the max probability of a path ending in w_k that visited the landmarks $(\ell_1, \ldots, \ell_i)$ and $P(r_L = 1|W, L) = Q(n, W)$. Then, a graph search method integrated with the Dijkstra algorithm (Dijkstra, 2022) is designed for calculating W^*.

Algorithm 1 Graph Search (Shah et al., 2023)

Input: Landmarks $(\ell_1, \ell_2, \ldots, \ell_n)$.
Input: Graph $G(N, E)$.
Input: Starting node S.
$\forall i = 0, \ldots, n, \forall w \in N, Q[i, w] \leftarrow -\infty$
$Q[0, S] \leftarrow 0$
DIJKSTRA_ALGORITHM$(G, Q[0, *])$

for $i \in 1, 2, \ldots, n$ **do**
  $\forall w \in W, Q[i, w] \leftarrow Q[i-1, w] +$ LLM(w, ℓ_i)
  DIJKSTRA_ALGORITHM$(G, Q[i, *])$
end for
$destination \leftarrow \arg\max(Q[n, *])$
return BACKTRACK$(destination, Q[n, *])$

A.3 Point Clould Construction

As shown in Fig.2, with the camera intrinsic matrix K and agent's pose (p, α), pixels p_o of the depth image of observation view can be projected to a 3D point cloud in the world coordinate system as $P_w^{\alpha'} = R_{\alpha'} \cdot Z \cdot K^{-1} \cdot p_o + p$, where $R_{\alpha'}$ is the rotation matrix of observation view and Z denotes depth values of pixels. The final point cloud $\mathcal{M}_s$ is given by $\mathcal{M}_s = P_w^{\alpha-90^\circ} \odot P_w^{\alpha-45^\circ} \odot P_w^{\alpha} \odot P_w^{\alpha+45^\circ} \odot P_w^{\alpha+90^\circ}$. Note that RGB-D images in each observation view are well-aligned, meaning that the extracted semantic masks of RGB images have their counterparts in depth images as well as in the point cloud. To this end, a local map containing both semantic and spatial information is constructed.

A.4 Implementation Details

For all experiments, we employ GPT-4V (Achiam et al., 2023) for object reasoning and landmark phase extraction. During the image grounding, a target is considered successfully detected if the bounding box's confidence score exceeds the threshold $\theta = 0.4$. The agent is equipped with an aligned RGB-D camera with 512x512 resolution and 90° field of view (FOV), capturing panoramic observations by rotating itself. The panoramic view directions are set at $p - 90^\circ$, $p - 45^\circ$, p°, $p + 45^\circ$, and $p + 90^\circ$ where p represents the agent's heading direction. The agent's low-level action space is (*"move forward"*, *"turn left"*, *"turn right"*, *"go up"*, and *"go down"*). The moving step is 5 meters and each rotation turns the agent by 15°. The agent will receive its GPS location at each step.

Before the evaluation, the agent initializes a memory graph of the current city scene. The agent is randomly spawned at an arbitrary location and cruises in the city by manual control. The agent records its visual observations and GPS coordinates every 5 meters. Note that GPS information is unnecessary since the off-the-shelf outdoor SLAM algorithm such as VINS (Qin et al., 2018) could be adopted to obtain the agent's accurate poses. Subsequently, a memory graph is initialized following the same procedures mentioned in §3.3. The graph contains some landmark observations in the test set and helps the agent's visual language navigation. Although the agent captures panoramic observations, only the front and back observations of the agent are stored to reduce the memory consumption of the memory graph.