DIVSCENE: Towards Open-Vocabulary Object Navigation with Large Vision Language Models in Diverse Scenes

Zhaowei Wang[1], **Hongming Zhang**[2], **Tianqing Fang**[1,2], **Ye Tian**[3], **Yue Yang**[4], **Kaixin Ma**[2], **Xiaoman Pan**[2], **Yangqiu Song**[1], **and Dong Yu**[2]
[1]CSE Department, HKUST, [2]Tencent AI Lab, Bellevue, USA
[3]Robotics X, Tencent, [4]University of Pennsylvania
zwanggy@cse.ust.hk, hongmzhang@global.tencent.com

Abstract

Large Vision-Language Models (LVLMs) have achieved significant progress in tasks like visual question answering and document understanding. However, their potential to comprehend embodied environments and navigate within them remains underexplored. In this work, we first study the challenge of open-vocabulary object navigation by introducing DIVSCENE, a large-scale dataset with 4,614 houses across 81 scene types and 5,707 kinds of target objects. Our dataset provides a much greater diversity of target objects and scene types than existing datasets, enabling a comprehensive task evaluation. We evaluated various methods with LVLMs and LLMs on our dataset and found that current models still fall short of open-vocab object navigation ability. Then, we fine-tuned LVLMs[1] to predict the next action with CoT explanations. We observe that LVLM's navigation ability can be improved substantially with only BFS-generated shortest paths without any human supervision, surpassing GPT-4o by over 20% in success rates.

1 Introduction

Large Vision-Language Models (LVLMs), such as Qwen2.5-VL (Bai et al., 2025), GPT-4o (OpenAI, 2023), and others (Zhu et al., 2025), have demonstrated state-of-the-art performance on a variety of vision-and-language tasks. However, their ability to comprehend embodied environments and navigate within them has been less explored, primarily due to the limited diversity of scenes and objects in current navigation benchmarks. For example, Matterport-3D (Chang et al., 2017) only considers 21 types of target objects in 90 private homes, and ProcTHOR (Deitke et al., 2022) contains 16 object types in four kinds of rooms (i.e., bedroom, living room, kitchen, and bathroom).

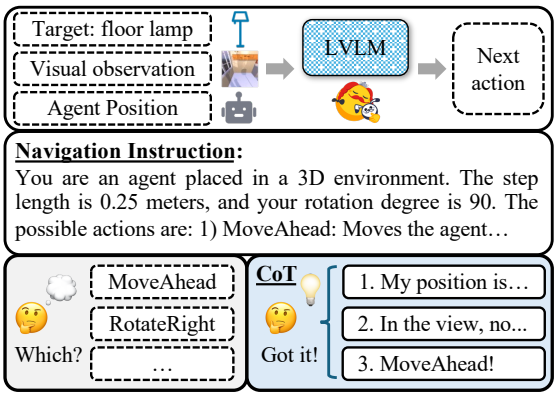


Figure 1: Illustration of our open-vocabulary object navigation. When fine-tuning LVLMs, we build CoT explanation traces to help them better grasp the rationale of object navigation.

In this work, we propose the new task of open-vocab object navigation and first study LVLMs' performance on the task, where an agent is required to navigate to a wide range of target objects without a pre-defined vocabulary. With this purpose, we introduce a new dataset, DIVSCENE, which features the most comprehensive range of scene types and target objects to the best of our knowledge. Specifically, we collect 81 scene types based on the MIT Scenes Dataset (Quattoni and Torralba, 2009). Then, we use LLMs to automatically compose diverse house descriptions by adding attributes to those scene types, such as "a *bakery* with tile-patterned walls." We input these descriptions into a language-guided framework, Holodeck (Yang et al., 2024), to build houses automatically with the strong ability of GPT-4 (OpenAI, 2023). In total, we compile 4,614 houses across 81 distinct scene types on the AI2THOR platform (Kolve et al., 2017). For benchmarking and training LVLMs, we further sampled shortest-path episodes in the houses from DIVSCENE. Specifically, we discretize houses into grid maps with a fixed step size and randomly sample target objects in each house. Then, we search for the shortest paths from

[1]Our code and data are available at https://github.com/zhaowei-wang-nlp/DivScene.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 9666–9686
November 4-9, 2025

the agent's initial position to target objects with BFS. In total, we collect about 23K shortest-path episodes, forming the DIVSCENE$_{ep}$. In total, our episode data contains 5,707 types of target object, significantly surpassing existing object navigation datasets (Deitke et al., 2020; Chang et al., 2017). This makes it a comprehensive testbed for evaluating the open-vocab navigation capabilities.

With our collected data, we benchmarked various methods based on LVLMs and LLMs, including blind LLMs, LLMs with image captioning, open-source LVLMs, and proprietary LVLMs. Our results show that most current models are still unable to explore environments and navigate toward open-vocabulary objects. Specifically, most models failed to outperform a random baseline, while GPT-4o achieves a success rate of slightly above 30%, far from adequate for real-world applications. To enhance their navigation ability, we further studied fine-tuning LVLMs on the data sampled in our DIVSCENE. Specifically, we propose a new navigational LVLM, called NATVLM (**Na**vigational Chain-of-**T**hought **VLM**), which is fine-tuned from Idefics 2 (Laurençon et al., 2024) on our sampled episodes in DIVSCENE$_{ep}$. As shown in Figure 1, the model is tuned to process the current observation, such as the egocentric view and the agent's status, and generate the next action. We train Idefics 2 with imitation learning (IL; Brohan et al., 2022) using BFS-generated shortest paths in DIVSCENE$_{ep}$. To further improve the navigation ability, we also manually collect CoT explanation traces of each action prediction (Mitra et al., 2023; Ho et al., 2023) to help Idefics 2 understand the underlying rationale behind the navigation task.

In our experiments, we surprisingly discovered that simply imitating the shortest paths constructed by breadth-first search can be an effective approach to enhance the open-vocab navigation ability of LVLMs. Our NATVLM model outperforms off-the-shelf LVLMs and LLMs by a large margin, achieving a success rate approximately 20% higher than GPT-4o. Compared to existing IL-based methods, which rely on training models with large corpora of costly human demonstrations (Brohan et al., 2022; Wei et al., 2023), our approach offers a far more efficient and economical alternative. Further, we carry out thorough ablation studies to show the efficacy of CoT explanation traces in action prediction. Moreover, few-shot experiments demonstrate the robustness of our agent. Last but not least, we validate the generalization ability of our agent on three unseen datasets: ProcTHOR (Deitke et al., 2022), iTHOR (Weihs et al., 2021), and HM3D (Ramakrishnan et al., 2021).

2 Related Work

Object Navigation: The Reinforcement Learning has long been used for tackling object navigation tasks (Zhu et al., 2017; Druon et al., 2020; Ehsani et al., 2021). However, recent studies (Ehsani et al., 2023) show that RL requires extensive reward shaping, limiting its practicality. Alternatively, imitation learning (Pomerleau, 1988; Zhang et al., 2018) has also been applied to these tasks, inspiring many subsequent works (Brohan et al., 2022, 2023; Ehsani et al., 2023). However, existing works only focus on closed-vocab object navigation tasks, mainly due to the limited diversity of current benchmarks. For example, Matterport-3D (Chang et al., 2017) only considers 21 kinds of target objects in private homes. ProcTHOR (Deitke et al., 2023a) solely contains 16 target objects in four kinds of rooms. Although HM3D-OVON (Yokoyama et al., 2024b) attempts to broaden the diversity, the number of object categories is still limited to 379, and there are only 181 scenes, making it difficult to fully reflect the open-vocab nature of real-world scenarios. In constrast, we introduce DIVSCENE, a dataset comprising 5,707 categories of target objects among 22,696 object types, spanning 4,614 distinct scenes. Therefore, our dataset reflects a more diverse nature of real-world settings with an open vocabulary.

VLMs and LLMs for Embodiment: LLMs and VLMs (Achiam et al., 2023; Touvron et al., 2023; Lu et al., 2024) have emerged as the foundation for solving embodied tasks (Pan et al., 2023; Majumdar et al., 2024). A few methods have employed contrastive VLMs (Radford et al., 2021; Li et al., 2022) as the visual encoders (Khandelwal et al., 2022; Majumdar et al., 2022) or object-grounding tools (Gadre et al., 2023; Dorbala et al., 2023) for navigation. Yu et al. (2023) utilizes an LLM as the planning backbone with a captioning model for perception. Following them, many improvements have been proposed (Cai et al., 2024; Chen et al., 2023; Zhou et al., 2023; Shah et al., 2023). While these works still focus on a small set of target objects, we explore the potential of recent LVLMs for open-vocabulary object navigation.

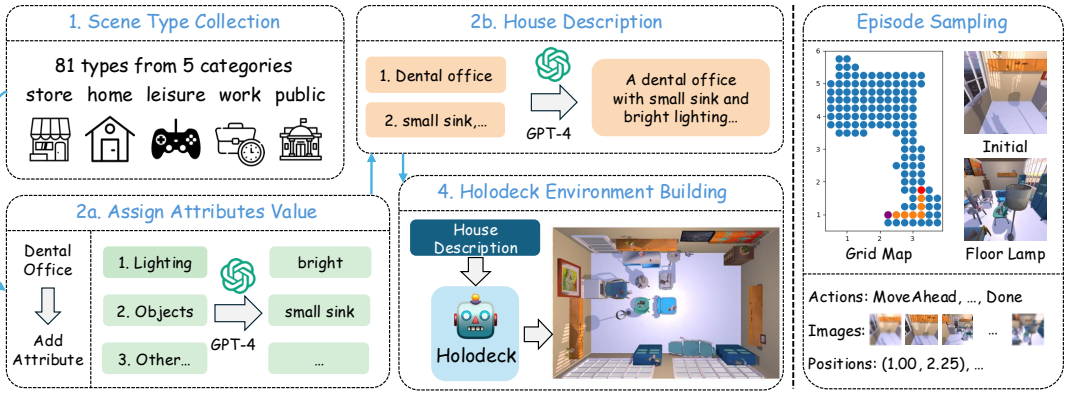


Figure 2: Data collection process. On the left, we show the process of collecting scenes. We prompt GPT-4o to collect textual house descriptions and use Holodeck to build houses. On the right, we show an episode built in the house, where we use BFS to find the shortest path. Then, actions and observations are collected.

3 Task Definition

We study the open-vocab object navigation task involving an agent in an environment to find the target object belonging to a given category. The object categories in the open-vocabulary setting are denoted by $C = \{c_0, c_1, c_2, \ldots\}$, which is not restricted to a predefined set. The houses can be described by $H = \{h_0, h_1, \ldots, h_n\}$, and n is the total number of scenes. In each episode, the embodied agent is initialized at a random position p_i with rotation r_i in a house h_i. Then, the embodied agent is required to perform instance-level object navigation, where it gets a target object within the category c_i at the position o_i. Thus, an episode can be represented as $E^{(i)} = \{h_i, p_i, r_i, c_i, o_i\}$. At each time step t, the embodied agent observes the environment and predicts the next action a_t. Following previous work (Yu et al., 2023; Zhu et al., 2024), the observation comprises an RGB image of the egocentric view and the agent status (i.e., its position and rotation).

Our new houses are all collected on the AI2THOR platform (Kolve et al., 2017), and thus the action space, signified as A, covers: MoveAhead, RotateRight, RotateLeft, and Done. By default, the MoveAhead action moves the agent 25 centimeters, and RotateLeft (or RotateRight) rotates the agent 90 degrees. The Done action is used by the agent to indicate that the navigation task is complete. When the agent takes the Done action or reaches the max action limit, the episode is considered successful if the distance to the target object is below 1.5 meters. Based on the actions, an environment can be discretized as a 0.25 $\times$ 0.25-meter grid map of all reachable positions.

Figure 3: Examples of houses with different scene types.

4 Data Collection for DivScene and DivScene$_{ep}$

Existing object navigation studies only focus on limited types of scenes and objects. To fill this gap, we first curate a large-scale scene dataset DivScene, featuring 4,614 scenes. Then, 23K shortest-path episodes with 5,707 kinds of target objects were sampled using breadth-first search (BFS), forming the DivScene$_{ep}$ dataset. We illustrate the details in Figure 2.

4.1 Scene Collection for DivScene

We adopt Holodeck (Yang et al., 2024) to build scenes, easing human labor. Holodeck takes textual house descriptions as input and uses GPT-4 to decide the layout, styles, and object selections. To collect diverse houses, we first manually compile 81 scene types across five categories by supplementing the MIT scene dataset (Quattoni and Torralba, 2009), like music studio and home office.

Then, we build textual house descriptions based on randomly chosen scene types by adding house attributes. We consider 12 house attributes, such as room style, users of the room, etc. We randomly sample 1-3 attributes and prompt GPT-4 to assign specific values to them. Given the scene type and attribute values, a house description is then writ-

ten by GPT-4. Here, we prompt GPT-4 under the in-context learning setting (Brown, 2020), and five exemplars are provided to generate the description for the final test example. See full details of scene types and attributes in Appendix A.1 and concrete prompts in Appendix A.2. With a strict data filtering (Appendix A.3), we obtained 4,614 houses. We present a few houses in Figure 3 and more examples in Appendix E.

4.2 Episode Collection for DIVSCENE$_{ep}$

There are three steps in our episode sampling method. First, we sample the agent's initial position and the target object. Then, we use BFS to find the shortest paths in the 0.25×0.25-meter grid map. Finally, we obtain the action sequence and corresponding observations. We show an example on the right side of Figure 2.

First, as discussed in Section 3, houses are discretized as grid maps of the fixed step size. We randomly sample an initial position from the grip map for the agent. Then, a target object is randomly sampled from all available objects in the environment. To encourage diversity, objects of the same type as those sampled in previous episodes are removed from the pool when sampling for new episodes in the same house. We also impose that the target objects are between 0.3m and 2.0m in height to ensure they are observable to the agent.

Second, we find the shortest path in the grid map that navigates a multi-room, cluttered environment. We can obtain the ground truth information from AI2-THOR, like reachable positions and objects' coordinates. We use BFS to find the shortest path from the initial position to the target object. More details are in Appendix A.4. This ground truth information is not provided to agents for inference and is only used to produce episodes for training.

With the shortest path, we then derive the sequence of actions needed to achieve navigation. Basically, between two adjacent positions in the shortest path, we add a MOVEAHEAD action if the agent's rotation remains unaltered. Otherwise, we first use ROTATERIGHT or ROTATELEFT to adjust the orientation and then add a MOVEAHEAD action (more details in Appendix A.5). Thus, we obtain an action sequence that can steer the agent to the target object. We execute all actions in AI2THOR and collect observations at each step, including the agent's status and egocentric images.

Metric	ProcTHOR	iTHOR	DivScene
Scene types	4	4	81
Rooms per house	3.78	1.00	2.35
Room size (m^2)	25.21	34.24	30.15
Obj. types per house	16.25	30.92	32.17
Obj. per house	35.64	47.26	111.42
Obj. types	38	116	22,696

Table 1: A comparison of ProcTHOR, iTHOR, and our dataset. "Obj." refers to objects.

4.3 Statistics and Comparison

In DIVSCENE, we collected 4,614 houses across 81 scene types. To the best of our knowledge, this dataset covers the widest range of scenes. We show the diversity of our collected houses in Figure 6 by plotting most types under each category. Thanks to Objaverse (Deitke et al., 2023b), the collected houses contain objects from 22,696 different types, including very common objects such as fridges, beds, shelves, and sofas, and rare objects such as multicolored bookshelf and vintage wooden bench.

Then, we randomly pick one house of each scene type to build a test set of 81 houses. Similarly, we randomly select 27 houses of distinct scene types as the validation set. DIVSCENE is divided into training, validation, and test sets, covering 4,506, 27, and 81 houses. On the training set, we sample five episodes in each scene with different target objects. Four episodes are selected in each house in the evaluation sets to balance the evaluation efficiency and accuracy. In total, the DIVSCENE$_{ep}$ dataset contains 22,962 episodes and 5,707 different kinds of target objects.

We provide a detailed comparison of scene complexity between our dataset and the other two datasets: iTHOR (Weihs et al., 2021) and ProcTHOR (Deitke et al., 2022). As shown in Table 1, our dataset includes the most objects and scenes, and room sizes and numbers in our dataset remain consistent with manually created ones in iTHOR. More statistics are shown in Appendix A.6.

5 NatVLM

In this section, we describe our NATVLM model. It is fine-tuned from Idefics 2 (Laurençon et al., 2024) through imitation learning on shortest paths from DIVSCENE$_{ep}$ to enhance the navigation ability. At each time step t, the model processes environment observations s_t and generates the next action a_t in a manner consistent with instruction tuning. Figure 1 provides an overview of NATVLM.

5.1 Instruction Compilation

We manually build instructions for each time step in an episode, which contains four parts: (i) the instruction provides a brief introduction to the object navigation task, such as possible actions and step length; (ii) we add the current environment observation, including the target object, its position, the agent's position and rotation, and visual observation; (iii) we provide the agent's positions and actions for the recent $M = 8$ steps, along with the visual observations from the recent $K = 4$ steps, to help the agent capture navigation history; (iv) the instruction asks the model to predict the next action by considering the current observation and navigation history in accordance with the CoT explanation in the responses. We show the specific instruction template in Appendix B.1.

5.2 Response with CoT Explanation

We tune the Idefics 2 to generate the next action when instructions are given. Specifically, we encode actions as natural language and use the model's generative capabilities to decode the next action. Nonetheless, we found that merely requiring LVLMs to output the next action leads to unsatisfactory results. The model only grasps the surface-level styles of the prompt but misses the underlying rationales of object navigation.

To enhance its understanding of the object navigation task, we manually build responses with CoT explanation traces during the navigation process. The structure of the responses covers three steps. In the first step, we have the agent compare its current position with the target and determine whether it needs to move forward or take other actions. After this, the agent is asked to check the obstacles in the visual observation to see whether it needs to rotate. In the last step, the agent gives the final decision based on the analyses in the first two steps. In contrast to writing explanations with LLMs (Mitra et al., 2023), we manually write the prompt template of explanation traces, leaving the position information to be filled in with coordinates at each step. Meanwhile, we employ a few postprocessing steps, like action balancing and conflict filtering. We show concrete prompts and postprocessing details in Appendices B.2 and B.3, respectively.

5.3 Imitation Learning Objective

With collected instructions and responses, we employ imitation learning to train NATVLM to mimic decision-making from shortest-path demonstrations. Specifically, we adopt the *Behavior Cloning* (BC) framework, where the goal is to learn a policy $\pi_\theta(a_t|s_t)$ that maps the current state s_t (environment observation) to the action a_t at each time step t. We minimize the negative log-likelihood of the shortest-path actions over collected observation-action pairs (s_t, a_t) obtained from trajectories in DIVSCENE$_{ep}$, which is denoted as $\mathcal{D}$. The objective function is given by:

$$\mathcal{L}_{\text{BC}}(\theta) = -\mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\big[\log \pi_\theta(a_t|s_t)\big], \quad (1)$$

where θ represents the learnable parameters of Idefics 2. As we aforementioned, the state s_t is included in the instruction, and the action a_t includes the CoT explanation besides the next action.

6 Experiment

We conduct extensive experiments to evaluate both off-the-shelf LVLMs and LLMs, as well as models that have been fine-tuned using DIVSCENE$_{ep}$.

6.1 Metrics

A model navigates in a house until it chooses the DONE action or reaches the maximum action limit of 200 steps. We report the metrics Success Rate (SR), Success weighted by Path Length (SPL; Anderson et al., 2018), and Success weighted by Episode Length (SEL; Eftekhar et al., 2023). An episode is considered successful when the target object appears in the agent egocentric observation and is less than 1.5 meters away. Specifically, SR and SPL are computed as $\frac{1}{N}\sum_{i=1}^{N} S_i$ and $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(l_i,p_i)}$, where N is the number of episodes, S_i is the indicator of success, l_i is the length of the shortest path, and p_i is the length of the predicted trajectory. For SEL, we replace l_i and p_i with the action number of the shortest and predicted paths.

6.2 Evaluated Methods

Besides NATVLM, we comprehensively evaluate four off-the-shelf methods and one fine-tuned method on our sampled dataset[2]:

Blind LLMs are text-only LLMs that simply predict the next action based on the textual instruction without considering any visual information. This

[2]We do not include complicated frameworks built on LVLMs as they are out of the evaluation scope. More in Appendix C.5.

Methods	Backbone	Valid			Test			All		
		SR	SPL	SEL	SR	SPL	SEL	SR	SPL	SEL
Random	-	9.26	8.19	9.26	6.79	5.77	6.79	8.03	6.98	8.03
Blind LLMs	Llama 2 (7B)	8.33	7.26	3.64	9.57	7.63	6.28	8.95	7.45	4.96
	Llama 2 (13B)	9.26	7.69	3.12	10.19	8.62	4.14	9.72	8.15	3.63
	Llama 3.1 (8B)	11.11	9.40	5.56	12.04	9.50	6.19	11.57	9.45	5.88
	Mistral (7B)	8.33	7.16	4.13	9.88	7.89	3.78	9.11	7.53	3.96
LLMs w/ Captions	Llama 2 (7B)	11.11	9.30	5.06	12.96	10.90	8.19	12.04	10.10	6.62
	Llama 2 (13B)	9.26	7.56	4.03	12.35	9.93	5.95	10.80	8.74	4.99
	Llama 3.1 (8B)	12.96	10.73	2.75	16.67	13.50	6.28	14.82	12.12	4.52
	Mistral (7B)	11.11	9.65	3.43	11.76	9.65	2.72	11.43	9.65	3.07
Open LVLMs	Qwen-VL (7B)	10.19	8.75	9.14	7.41	6.05	6.66	8.80	7.40	7.90
	Llava 1.5 (7B)	12.04	10.07	9.88	12.35	10.03	10.30	12.20	10.05	10.09
	Llava 1.5 (13B)	12.04	10.50	11.05	10.62	8.73	9.75	11.33	9.62	10.40
	Idefics 2 (8B)	21.30	17.88	14.31	20.68	17.18	16.49	20.99	17.53	15.40
	Qwen2.5-VL (7B)	11.11	8.96	10.94	10.20	8.14	9.70	10.65	8.55	10.32
	InternVL3 (8B)	11.43	9.62	10.26	11.46	9.59	11.46	11.45	9.61	10.86
	Gemma3 (12B)	15.74	12.96	5.94	22.22	18.36	5.17	18.98	15.66	5.55
API LVLMs	GPT-4v	33.33	28.79	18.81	32.10	26.39	18.26	32.72	27.59	18.54
	GPT-4o	37.04	31.82	29.47	38.27	31.74	27.92	37.66	31.78	28.70
Fine-Tuned LVLMs	Idefics 2 (8B)	29.63	25.01	23.18	26.54	22.11	21.42	28.09	23.56	22.30
NatVLM (Ours)	Idefics 2 (8B)	**57.41**	**47.84**	**47.90**	**54.94**	**44.45**	**45.83**	**56.17**	**46.15**	**46.86**

Table 2: Performance of all models DivScene$_{ep}$. "All" means the average of both evaluation sets. The highest scores are bolded. In the "LLMs w/Captions" baseline, we use Llava 1.5 as the captioning model.

method references how far we can get solely using prior world knowledge and random guessing. For the LLM choice, we evaluate Llama 2 (7B, 13B) (Touvron et al., 2023), Llama 3.1 (8B) (Dubey et al., 2024), and Mistral (7B) (Jiang et al., 2023).

Socratic LLMs w/ Image Captions is the simplest method that leverages visual information. Here, we use an image captioning model to convert egocentric images into language descriptions, allowing LLMs to obtain the content of visual information. We employ Llava 1.5 (Liu et al., 2024a) as the captioning model while using the same LLMs as those in the "Blind LLM" baseline.

Open-Source LVLMs are directly tested without any further tuning. They are capable of processing images in addition to textual queries. Here, we test Llava 1.5 (7B, 13B) (Liu et al., 2024b,a), Qwen-VL (7B) (Bai et al., 2023), Idefics 2 (8B) (Laurençon et al., 2024), Qwen2.5-VL (7B) (Bai et al., 2025), InternVL3 (8B) (Zhu et al., 2025), and Gemma3 (12B) (Team et al., 2025).

API-based LVLMs we evaluated include GPT-4v (OpenAI, 2023) and GPT-4o (OpenAI, 2024). They can process multiple images and achieve state-of-the-art performance on multimodal tasks.

Fine-Tuned LVLM: For fine-tuned LVLMs on our data, we also evaluated fine-tuning Idefics 2 (Laurençon et al., 2024) on our sampled trajectories without any CoT explanation traces.

6.3 Main Evaluation

We present the results of all off-the-shelf and fine-tuned methods on the validation and test sets in Table 2. We also include the performance of selecting a random action at each step (i.e., **Random**) as a reference. In general, the fine-tuned NatVLM achieves the best performance on object navigation, exceeding other methods by a large margin. For example, NatVLM can successfully navigate to 57.41% of episodes on the validation set, increasing by about 20% compared to the GPT-4o baseline. Meanwhile, according to the higher SPL and SEL on both test and validation sets, NatVLM can navigate to target objects with better efficiency.

For the off-the-shelf methods, the blind LLMs achieve performance slightly higher than the random results. For example, Llama 3.1 (8B) achieves a success rate of 11.57%, about 4 points higher than the random guess. In the meantime, we observe that the performance of all LLMs only improved marginally when we added the captioning model to provide additional perceptual information. This result shows that the captioning model can miss important image content details, leading to unsatisfactory improvement. On the other hand, we find that LVLMs can achieve the best results across all baselines. For example, the closed-source LVLMs, GPT-4v and GPT-4o, can successfully navigate to

Methods	Valid			Test			Test Diff		
	SR	SPL	SEL	SR	SPL	SEL	Δ_{SR}	Δ_{SPL}	Δ_{SEL}
NATVLM (Ours)	**57.41**	**47.84**	**47.90**	**54.94**	**44.45**	45.83	-	-	-
◇ w/o ET	29.63	25.01	23.18	26.54	22.11	21.42	↓28.40	↓22.34	↓24.41
◇ w/o ET & w Gold	28.70	24.12	23.38	30.86	25.46	25.58	↓24.08	↓18.99	↓20.25
◇ w Gold Label	59.26	49.01	51.33	62.96	50.54	54.12	↑8.02	↑6.09	↑8.29
◇ w Diff-EQ	54.63	45.02	46.48	54.32	43.59	**46.84**	↓0.62	↓0.86	↑1.01

Table 3: The ablation study. Δ_* columns show score differences. We remove explanation traces and test different methods of position comparisons. We bold the highest scores except for the gold label test (◇ w Gold Label).

target objects in more than 30% cases. In addition, Idefics 2 (8B) attains success rates exceeding 20% on both validation and test sets.

6.4 Ablation Evaluation

To better understand the role of CoT explanation traces in NATVLM, we conduct two ablation studies to analyze its contribution. First, we verify the efficacy of CoT explanation traces. Then, we analyze different ways to compare the positions of the agent and target object. The results of experiments are shown in Table 3.

First, we remove explanation traces in the instruction tuning data (◇ w/o ET). Thus, we fine-tune the agent to only generate the next action. We find that the performance of our agent drastically drops, verifying that explanation traces can help LVLMs to better understand the underlying rationales of object navigation. Furthermore, we enhance the agent's input by providing the gold label of the positional difference between the agent and the target object (◇ w/o ET & w Gold). We observe that the gold labels cannot help much, as the performance only fluctuates somewhat.

Then, we study the effects of the position comparison, the crucial component in our explanation traces. The default prompt is to directly generate the position difference: "the difference to the target object is **[position_diff]**" as shown in Table 11, where **[position_diff]** is a placeholder. First, we provide the global label of positional differences in the input instructions (◇ w Gold Label). The results in Table 3 show that providing global labels can improve the performance of our agent, suggesting that our agent may occasionally compute the positional difference with errors. Next, we test the difference-equation prompt (◇ w Diff-EQ), where we fine-tune our agent to generate an equation for computing the positional difference. However, writing the equation of computation only leads to small variations in performance. All the abovementioned prompts are shown in Appendix C.1.

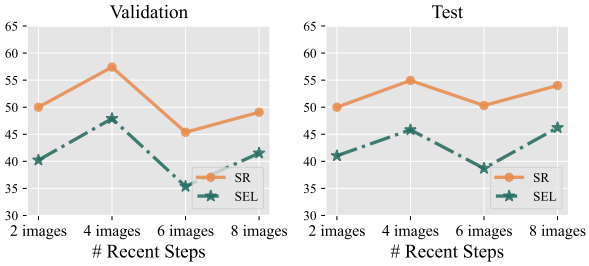


Figure 4: Design investigation. We provide different numbers of recent visual observations with the embodied agent. See scores of all metrics in Appendix C.2

6.5 Design Investigation

In this experiment, we thoroughly investigate a few design decisions regarding the image number and step number of recent positions and actions. We also test NATVLM with different prompts shown in Appendices C.2 and C.3

The default design provides NATVLM with four images. In addition, we test its performance using different numbers of input images, including 2, 6, and 8 images. The results are plotted in Figure 4. First, we observe a decline in the agent's performance when provided with only two images, showing that using fewer images leads to worse performance. For instance, our agent finishes the navigation successfully only 50% of the time, underperforming the 4-image baseline. Moreover, increasing the number of images to 6 or 8 does not result in further improvements. Thus, we choose to provide NATVLM with four images for the trade-off between accuracy and efficiency.

We also investigate the effect of recent positions and actions on navigation performance. By default, we provide NATVLM with information about the recent 8 steps. Here, we test the performance when we provide the positions and actions of 4, 12, and 16 steps. As illustrated in Table 4, a substantial performance improvement is evident when increasing

Number	Valid			Test		
	SR	SPL	SEL	SR	SPL	SEL
4 steps	46.30	38.66	39.82	45.99	37.45	39.60
8 steps	**57.41**	**47.84**	**47.90**	**54.94**	**44.45**	**45.83**
12 steps	42.59	35.92	36.43	50.93	41.15	43.20
16 steps	51.85	43.01	42.72	53.40	43.03	44.08

Table 4: Our agents receive various numbers of recent actions and positions. See all metrics in Appendix C.3.

% Data	Test			Test Diff w GPT-4o		
	SR	SPL	SEL	Δ_{SR}	Δ_{SPL}	Δ_{SEL}
20	38.89	31.40	32.13	↑0.62	↓0.34	↑4.21
40	38.27	31.00	30.79	↓0.00	↓0.74	↑2.87
60	52.12	42.25	44.40	↑13.85	↑10.51	↑16.48
80	49.69	40.26	42.49	↑11.42	↑8.52	↑14.57
100	**54.94**	**44.45**	**45.83**	↑16.67	↑12.71	↑17.91

Table 5: Few-shot learning ability. See scores of the validation set and more prompts in Appendix C.4.

the number of steps from 4 to 8. However, further increases in step count yield limited returns. Thus, we provide our agent with 8 steps.

6.6 Few-shot Learning Evaluation

NATVLM undergoes instruction tuning with an extensive training dataset, DIVSCENE$_{ep}$. In this section, we design a few-shot experiment to confirm its ability to generalize with fewer data. While the original training data contains five episodes per house, we train the model with only 1, 2, 3, and 4 episodes, representing 20%, 40%, 60%, and 80% of the full data. The results are shown in Table 5. For clarity, we also compare our models with the GPT-4o baseline. We can find that our agent has strong few-shot learning abilities. With only 20% percent training data, our agent can perform similarly to GPT-4o and generalize well on unseen houses. Meanwhile, the results demonstrate a gradual performance improvement as we incrementally increase the data volume, with performance plateauing at approximately 80% of the full dataset. We provide more analysis in Appendix C.4.

6.7 Zero-shot Transferring Evaluation

We further evaluate NATVLM on other house datasets: iTHOR (Weihs et al., 2021), ProcTHOR (Deitke et al., 2022), and HM3D (Ramakrishnan et al., 2021). We directly use NATVLM tuned on DIVSCENE$_{ep}$ to conduct zero-shot transferring evaluation. Since we do not tune hyperparameters on these datasets, we treat each dataset as a whole test set without any validation set. We show the results on iTHOR and ProcTHOR in Table 6. We also report the performance of open-source VLMs and some ablated models as baselines. The results show that our agent surpasses all the baselines on both datasets, indicating that our framework has a strong ability to generalize in other environments. We include results on HM3D and setup details in Appendix C.5.

Models	iTHOR			ProcTHOR		
	SR	SPL	SEL	SR	SPL	SEL
Qwen-VL (7B)	23.67	19.96	19.27	10.83	9.04	6.28
Llava 1.5 (7B)	24.12	20.32	20.34	16.04	13.53	14.02
Llava 1.5 (13B)	19.47	16.21	17.48	13.12	11.07	11.85
Idefics 2 (8B)	28.54	23.39	18.49	17.29	14.33	11.05
NATVLM	**72.79**	**59.34**	59.28	53.12	44.37	43.04
◇ w/o ET	38.27	32.15	31.43	31.25	26.87	26.20
◇ w Diff-EQ	72.32	58.98	**62.35**	**54.59**	**45.53**	**46.80**

Table 6: Zero-shot transferring on iTHOR and ProcTHOR.

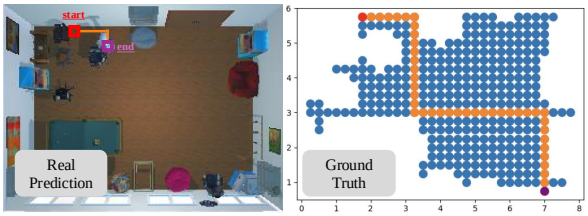


Figure 5: Error analysis. The agent needs to find a soda can in the lower right corner of a game room. The left image shows the predicted path, and the right image shows the ground truth in the grid map. The agent fails to find the soda can and wanders within a limited area.

6.8 Case Study

In Figure 5, we provide an example of error analysis of NATVLM. The left image shows the predicted path in the real scene, and the right image shows the ground truth in the corresponding grid map. Here, the agent needs to find a soda can after walking through the whole room. There are 46 actions in the shortest path. Instead of heading towards the goal, the agent just meanders in a limited area. This shows that NATVLM cannot finish the navigation with a long trajectory and has limited exploration. We leave the development of models better at exploring environments for future work.

7 Conclusion

In this work, we first study the open-vocab object navigation and evaluate extensive LVLMs and LLMs. Meanwhile, we collect a large-scale scene dataset, DIVSCENE, featuring 4,614 scenes. Over 22K episodes are sampled with the shortest paths

by BFS. Our evaluation shows that current LVLMs and LLMs still fall short of the ability of open-vocab object navigation. Then, we also fine-tuned LVLMs with the shortest paths through imitation learning, where we also introduced CoT explanations. Experiments show that the navigation ability of LVLMs can be improved significantly. For future work, a promising direction is to enable LVLMs to navigate over longer horizons.

Limitation

While we conducted extensive experiments, our model still has some failure cases due to long navigation horizon or exploration, as we discussed in Section 6.8. This might be because, due to the limited context window, we can only provide existing models with recent historical information. Thus, a promising direction for future work is to improve the memory (Du et al., 2025) or long-context (Team et al., 2025; Wang et al., 2025) capabilities of LVLMs to enable navigation over longer horizons.

Ethics Statement

Our scene dataset DIVSCENE is built upon the publicly available AI2THOR platform (Kolve et al., 2017) and the Holodeck framework (Yang et al., 2024). We further extend our experiments on iTHOR (Weihs et al., 2021) and ProcTHOR (Deitke et al., 2022), both of which are open-source datasets. We provide the complete details of the implementation of our NATVLM agent and baselines in Appendix D, including the learning rate, batch size, hard device, API access, etc. Meanwhile, we show the details of all the post-processing data in Appendices A.3 and B.3.

Acknowledgements

The authors of this paper were supported by the ITSP Platform Research Project (ITS/189/23FP) from ITC of Hong Kong, SAR, China, and the AoE (AoE/E-601/24-N), the RIF (R6021-20) and the GRF (16205322) from RGC of Hong Kong, SAR, China. This work was done when Zhaowei Wang worked as an intern at Tencent AI Lab.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and 1 others. 2018. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, and 1 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, and 1 others. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Wenzhe Cai, Siyuan Huang, Guangran Cheng, Yuxing Long, Peng Gao, Changyin Sun, and Hao Dong. 2024. Bridging zero-shot object navigation and foundation models through pixel-guided navigation skill. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5228–5234. IEEE.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*.

Junting Chen, Guohao Li, Suryansh Kumar, Bernard Ghanem, and Fisher Yu. 2023. How to not train your dragon: Training-free embodied object goal navigation with semantic frontiers. *arXiv preprint arXiv:2305.16925*.

Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, and 1 others. 2020. Robothor: An open

simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3164–3174.

Matt Deitke, Rose Hendrix, Ali Farhadi, Kiana Ehsani, and Aniruddha Kembhavi. 2023a. Phone2proc: Bringing robust robots into our chaotic world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9665–9675.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. 2023b. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. 2022. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994.

Vishnu Sashank Dorbala, James F Mullen Jr, and Dinesh Manocha. 2023. Can an embodied agent find your" cat-shaped mug"? llm-guided exploration for zero-shot object navigation. *arXiv preprint arXiv:2303.03480*.

Raphael Druon, Yusuke Yoshiyasu, Asako Kanezaki, and Alassane Watt. 2020. Visual object search by learning spatial context. *IEEE Robotics and Automation Letters*, 5(2):1279–1286.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z Pan. 2025. Rethinking memory in ai: Taxonomy, operations, topics, and future directions. *arXiv preprint arXiv:2505.00675*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ainaz Eftekhar, Kuo-Hao Zeng, Jiafei Duan, Ali Farhadi, Ani Kembhavi, and Ranjay Krishna. 2023. Selective visual representations improve convergence and generalization for embodied ai. *arXiv preprint arXiv:2311.04193*.

Kiana Ehsani, Tanmay Gupta, Rose Hendrix, Jordi Salvador, Luca Weihs, Kuo-Hao Zeng, Kunal Pratap Singh, Yejin Kim, Winson Han, Alvaro Herrasti, and 1 others. 2023. Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. *arXiv preprint arXiv:2312.02976*.

Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. 2021. Manipulathor: A framework for visual object manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4497–4506.

Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. 2023. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. 2022. Simple but effective: Clip embeddings for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14829–14838.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, and 1 others. 2017. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *arXiv preprint arXiv:2405.02246*.

Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, and 1 others. 2022. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024b. Visual instruction tuning. *Advances in neural information processing systems*, 36.

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

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Yaofeng Sun, and 1 others. 2024. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*.

Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. 2022. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. *Advances in Neural Information Processing Systems*, 35:32340–32352.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, and 1 others. 2024. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16488–16498.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, and 1 others. 2023. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

NVIDIA. 2021. Megatron-LM & Megatron-Core: GPU optimized techniques for training transformer models at-scale. https://github.com/NVIDIA/Megatron-LM.

OpenAI. 2023. Gpt-4v(ision) system card.

OpenAI. 2024. Gpt-4o system card.

Bowen Pan, Rameswar Panda, SouYoung Jin, Rogerio Feris, Aude Oliva, Phillip Isola, and Yoon Kim. 2023. Langnav: Language as a perceptual representation for navigation. *arXiv preprint arXiv:2310.07889*.

Dean A Pomerleau. 1988. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1.

Ariadna Quattoni and Antonio Torralba. 2009. Recognizing indoor scenes. In *2009 IEEE conference on computer vision and pattern recognition*, pages 413–420. IEEE.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, and 1 others. 2021. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*.

Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. 2023. Pirlnav: Pretraining with imitation and rl finetuning for objectnav. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17896–17906.

Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. 2023. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Conference on Robot Learning*, pages 2683–2699. PMLR.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, and 1 others. 2021. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Zhaowei Wang, Wei Fan, Qing Zong, Hongming Zhang, Sehyun Choi, Tianqing Fang, Xin Liu, Yangqiu Song, Ginny Y Wong, and Simon See. 2024. Absinstruct: Eliciting abstraction ability from llms through explanation tuning with plausibility estimation. *arXiv preprint arXiv:2402.10646*.

Zhaowei Wang, Wenhao Yu, Xiyu Ren, Jipeng Zhang, Yu Zhao, Rohit Saxena, Liang Cheng, Ginny Wong, Simon See, Pasquale Minervini, Yangqiu Song, and Mark Steedman. 2025. Mmlongbench: Benchmarking long-context vision-language models effectively and thoroughly. *Preprint*, arXiv:2505.10610.

Yao Wei, Yanchao Sun, Ruijie Zheng, Sai Vemprala, Rogerio Bonatti, Shuhang Chen, Ratnesh Madaan, Zhongjie Ba, Ashish Kapoor, and Shuang Ma. 2023. Is imitation all you need? generalized decision-making with dual-phase training. In *Proceedings*

of the IEEE/CVF International Conference on Computer Vision, pages 16221–16231.

Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. 2021. Visual room rearrangement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5922–5931.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, and 1 others. 2024. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16227–16237.

Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. 2024. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. *arXiv preprint arXiv:2410.08189*.

Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. 2024a. Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–48. IEEE.

Naoki Yokoyama, Ram Ramrakhya, Abhishek Das, Dhruv Batra, and Sehoon Ha. 2024b. Hm3d-ovon: A dataset and benchmark for open-vocabulary object goal navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5543–5550. IEEE.

Bangguo Yu, Hamidreza Kasaei, and Ming Cao. 2023. L3mvn: Leveraging large language models for visual target navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3554–3560. IEEE.

Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. 2018. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5628–5635. IEEE.

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.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, and 1 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.

Jun Zhu, Zihao Du, Haotian Xu, Fengbo Lan, Zilong Zheng, Bo Ma, Shengjie Wang, and Tao Zhang. 2024. Navi2gaze: Leveraging foundation models for navigation and target gazing. *arXiv preprint arXiv:2407.09053*.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE.

A DIVSCENE and DIVSCENE$_{ep}$ Details

A.1 Scene Type Details and Attribute List

This section lists all scene types we collected by complementing the MIT scene dataset. In total, there are 81 scene types across five different categories, as shown in Table 7.

To collect diverse house descriptions, we add various attributes to a randomly sampled scene type. Here, we consider 12 different attributes as shown in Table 8. We ask GPT-4 to assign a value to each attribute and write a house description.

A.2 Textual House Description Prompt

We use in-context learning to prompt the GPT-4 to write textual house descriptions of 81 different scene types. Here, we show the concrete prompt we used in Table 9. We randomly sample 1-3 house attributes and ask GPT-4 to assign a value to them. Then, a house description is written based on the given scene type and attribute values. We use five exemplars in the in-context learning setting.

A.3 Postprocessing of Textual House Description

After we collect textual house descriptions from GPT-4, we also introduce three filters to ensure their diversity and quality.

- We first introduce a ROUGE-L filter. To encourage diversity, a new description is discarded when its ROUGE-L similarity (Lin, 2004) with any existing description is above 0.8, following previous works (Wang et al., 2022, 2024).
- Second, if we cannot correctly parse the output in the first step and find the values of the given attributes, we remove the example.
- Third, if we cannot find the house description in the second step in the output from GPT-4, we remove the example. The last two steps mean that the output does not follow the output format specified in the instructions and exemplars (see Table 9).

A.4 Breadth-First Search for Shortest Path

We use a BFS-based planner to find the shortest path from the initial position to a target point on the grid map. Notice that the target object is not necessarily anchored to a point on the grid map for realism. Thus, we find the grid point nearest to the target object as the destination of the navigation.

The algorithm is shown in Algorithm 1, which is based on a priority queue. We design the BFS-based planner to pick the path with the fewest rotations to make it easier for LLMs to imitate. In detail, we add more costs when rotation changes since the agent needs one more rotation action before moving ahead, as shown at the 12th line in Algorithm 1.

A.5 Action Derivation Algorithm

We show the action derivation algorithm in Algorithm 2. Between two adjacent positions in the shortest path, we add a MOVEAHEAD action if the agent's rotation remains unaltered. Otherwise, we first use ROTATERIGHT or ROTATELEFT to adjust the orientation and then add a MOVEAHEAD action. After reaching the target object, we then rotate the agent so that the object is approximately centered in the agent's egocentric view.

A.6 Data Diversity

To study what types of scenes are gathered under each category, we identify the category-type structure of houses in DIVSCENE. We plot the top 10 most common scene types under each category in Figure 6. Then, we study the diversity of target objects in the episodes we sampled in DIVSCENE$_{ep}$. We plot the top 15 most common scene types and their top 5 target object types in Figure 7. Overall, we see quite diverse scenes and target objects in our datasets.

B NATVLM Instruction Data

In this section, we give the concrete prompts used in fine-tuning NATVLM.

B.1 Instruction Template

We show the instruction template we used in the Table 10. There are four parts in the instruction template, including a brief task introduction, episode-specific information, the status of recent steps, and the prediction steps that need to be considered. We leave a lot of placeholders for the episode and step information.

B.2 Response Template

Previous works usually collect explanation traces using GPT-4 (Mitra et al., 2023; Mukherjee et al., 2023; Ho et al., 2023). In contrast, we collect explanation traces with manually written templates. The templates and examples are shown in Tables 11 and 13. For ROTATERIGHT and ROTATELEFT, we

Category	Scene Type
Store (16 types)	bakery, grocery store, clothing store, deli, laundromat, jewellery shop, bookstore, video store, florist shop, shoe shop, toy store, furniture store, electronics store, craft store, music store, sporting goods store
Home (21 types)	bedroom, nursery, closet, pantry, children room, lobby, dining room, corridor, living room, bathroom, kitchen, wine cellar, garage, sunroom, cabinet, study room, apartment, home office, basement, attic, laundry room
Public spaces (9 types)	prison cell, library, waiting room, museum, locker room, town hall, community center, convention center, recreation center
Leisure (14 types)	buffet, fast-food restaurant, restaurant, bar, game room, casino, gym, hair salon, arcade, spa, concert hall, ski lodge, lounge, club
Working place (21 types)	hospital room, kindergarten, restaurant kitchen, art studio, classroom, laboratory, music studio, operating room, office, computer room, warehouse, greenhouse, dental office, TV studio, meeting room, school room, conference room, factory floor, call center, reception area, nursing station

Table 7: The categories and scene types used in our DIVSCENE dataset. In total, there are five categories and 81 scene types in our dataset.

Attribute	Example Value	Attribute	Example Value
Room Style	victorian, rustic	Flooring	soft and cushioned, hard
Objects in the Room	computers, desks, chairs, servers	Theme	industrial, contemporary
Number of Rooms	single room	Lighting	bright, warm ambient
Configurations	individual cubicles	Window	small, slightly slanted
Users of the Room	children of various ages	Room Size	spacious, medium-sized
Era	contemporary, modern	Wall Treatment	artistic paintings, calming color

Table 8: The 12 attributes we used to collect house descriptions. We also provide an example value of each attribute.

identified the three most common scenarios for rotation based on heuristic rules. Then, we wrote the template for each of them. The first scenario is that the distance difference between the agent and the target object becomes zero in the agent's rotation. Thus, the agent needs to navigate in the other direction. The second scenario involves the presence of obstacles in the agent's current path, necessitating a rotation to navigate around them. The final scenario involves adjusting the agent's rotation to center the target within its field of view, occurring at the end of the navigation process.

B.3 Data Postprocessing

Directly using all actions in every trajectory to conduct imitation learning brings about an extremely imbalanced dataset. As shown in Table 12, 77.94% actions are MOVEAHEAD. Then we downsampled MOVEAHEAD actions in the instruction dataset. We only retain 25% of the MOVEAHEAD actions resulting in a more balanced dataset.

Then, we also remove conflicting data. We find that steps from different trajectories within the same house occasionally exhibit conflicting information. They have the exact same input information but different action predictions. This happens when two overlapped trajectories diverge at some point due to different target objects. Those conflicting data can confuse the fine-tuned LVLM and lead to worse performance. Thus, we remove those conflicting data from our dataset.

We show the final distribution of our dataset in Table 12 (w/ Postproc).

C Supplementary Experiment Details

In this section, we provide supplementary experiment results and show the prompt details.

C.1 Ablation Study Prompt

We change the prompt templates for tuning our agent NATVLM in the ablation studies.

(1) For adding the gold label of position difference (◇ w/o ET & w Gold and ◇ w Gold Label), we append a new sentence "The difference to the target object is **[position_diff]**" to the end of the **Episode-Specific Information** part of the instruction template.

(2) For removing the explanation traces, the **Prediction Steps** part of the instruction template is replaced with one shorter sentence, "Please generate the next step given the above states."

Task Instruction: Create a detailed and fluent description for a house based on the given scene type and features in two steps. Step 1: provide the value of each feature. Step 2: write a short phrase to describe the scene type with the values.
Exemplar1 Input: The given house type is "arcade." The feature list is: "(1) Objects in the room." **Exemplar1 Output:** Step 1: (1) a pool table\n Step 2: An arcade with a pool table
Following Exemplars: Exemplar 2, ..., Exmplar 5
Testing Input: The given house type is "office." The feature list is: "(1) Number of Rooms (2) Users of the Room (3) Configurations."

Table 9: The prompt we used to collect textual house descriptions using GPT-4. Here, we use 5 exemplars in the in-context learning. We show one example here for saving space.

Figure 6: The top 10 most common room types (outer circle) under each room category (inner circle) in the collected houses.

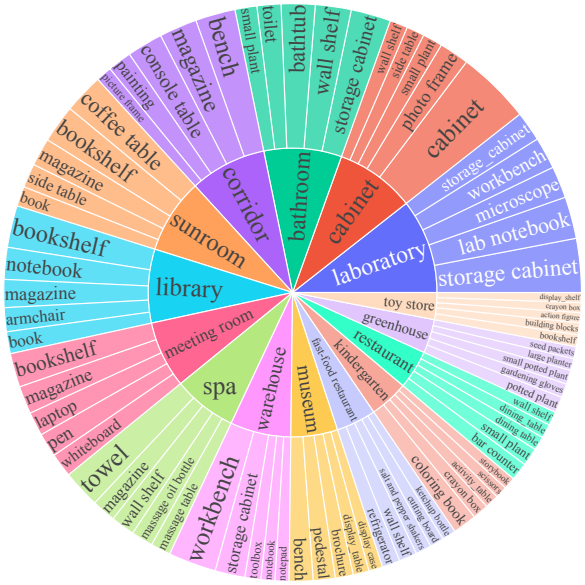


Figure 7: The top 15 most common scene types (inner cycle) and their top 5 target object types (outer cycles).

(3) For the **difference-equation prompt** in the ◇ w Diff-EQ experiment, we replace **[position_diff]** in all explanation traces with the equation: **[grid_obj_pos] - [agent_pos] = [position_diff]**.

C.2 Full Results of the Image Number Experiment

We provide the full results of the image number experiment of all metrics in Table 14. Besides the default prompt structure, we also conduct experiments with the difference-equation prompt, which is introduced as "◇ w Diff-EQ" in the ablation study.

C.3 Full Results of the Action Number Experiment

We provide the full results of the action number experiment with the default prompt in Table 15. We also provide the results with the difference-equation prompt in Table 16. The prompt is used as "◇ w Diff-EQ" in the ablation study.

C.4 Full Results of the Few-Shot Experiment

We provide the full results of the few-shot learning with the default prompt in Table 17. We also provide the results with the difference-equation prompt in Table 18. The prompt is used as "◇ w Diff-EQ" in the ablation study.

C.5 Zero-shot Transferring Evaluation

iTHOR and ProcTHOR both encompass four distinct scene types: bedrooms, living rooms, kitchens, and bathrooms. iTHOR has 30 rooms for each scene type, designed by professional 3D artists. Meanwhile, ProcTHOR is a procedural house-generation system that constructs 10,000 unique houses automatically. Similar to iTHOR, we sample 30 houses for each scene type from ProcTHOR. Four episodes are sampled in each house to evaluate our agent.

On the HM3D dataset, we test NATVLM on all

Algorithm 1 BFS Search for Shortest Paths

Input: $reachable\_pos, start\_point, end\_point, start\_rotation$
Initialize priority queue Q, distance map d_map, and parent map p_map
Enqueue $(0, start\_point, start\_rotation)$ to Q
while Q not empty **do**
    $(cost, current, rotation) \leftarrow$ Dequeue from Q
    if $current = end\_point$ **then**
        return ReconstructPath(p_map, $current$)
    end if
    for $r \in$ {North, East, South, West} **do**
        $neighbor \leftarrow$ GetNeighbor$(current, r)$ ▷ Four cardinal directions
        if $neighbor \in reachable\_pos$ **then**
            $newCost \leftarrow cost+$ (1 if $r = rotation$ else 2) ▷ More cost if rotation changed
            if $neighbor$ not visited or $newCost < d\_map[neighbor]$ **then**
                $d\_map[neighbor] = newCost$ ▷ Update distance
                $p\_map[neighbor] = current$ ▷ Update parent
                Enqueue $(newCost, neighbor, r)$ to Q
            end if
        end if
    end for
end while
return No path found

1. Brief Introduction: You are an agent placed in a 3D environment. Your step length is 0.25 meters, and your rotation degree is 90.
The possible actions are:
1. MoveAhead: Moves the agent forward by 0.25 meters in the direction it is currently facing. For example, if the agent is at (x, y) facing 0 degrees (north), MoveAhead will result in (x, y + 0.25). If the agent is facing 90 degrees (east), MoveAhead will result in (x + 0.25, y). If the agent is facing 180 degrees (south), MoveAhead will result in (x, y - 0.25). If the agent is facing 270 degrees (west), MoveAhead will result in (x - 0.25, y).
2. RotateRight: Rotate right for 90 degrees (clockwise).
3. RotateLeft: Rotate left for 90 degrees. (counterclockwise).
4. Done: Indicate that you are near to the target object and finish the task.

2. Episode-Specific Information: You need to find a **[obj_type]** at the position **[obj_pos]**. To achieve this, we recommend you move to the position **[grid_obj_pos]** with a rotation of **[grid_obj_rotation]**.
Currently, you are at **[agent_pos]** with a rotation of **[agent_rotation]**.

3. Status of Recent Steps: The history of recent states are:
Position: **[recent_agent_pos]**, Rotation: **[recent_agent_rotation]**, Action: **[recent_action]**
...
Position: **[recent_agent_pos]**, Rotation: **[recent_agent_rotation]**, Current View: **[recent_agent_image]**, Action: **[recent_action]**

4. Prediction Steps: Please generate the next step given the above states with the following steps: 1) Consider your rotation and position. 2) Check the images to see obstacles or the target object. 3) Decide the action.

Table 10: Instruction Template we used to fine-tune the LVLM: Idefics 2. There are four steps in the template, and we leave step-wise and episode information with placeholders. **[obj_type]** and **[obj_pos]** are the type of target object and its location. On the grid map, we also provide the nearest point on the grid map **[grid_obj_pos]** with a rotation **[grid_obj_rotation]**. **[agent_pos]** and **[agent_rotation]** are the agent's position and rotation. There are also placeholders for the recent status: **[recent_agent_pos]**, **[recent_agent_rotation]**, **[recent_agent_image]**, and **[recent_action]**. Notice that we provide the information of the recent 8 steps and only provide the recent 4 images for inference efficiency.

houses. We compare our model with a lot of baselines on the HM3D dataset, including two lines of work: fine-tuned methods and zero-shot methods. As shown in Table 19, the SPL scores of baselines

Algorithm 2 Get a Sequence of Actions from a Shortest Path

Input: $shortest\_path, start\_rotation, target\_object$
$agent\_rotation \leftarrow start\_rotation$
Initialize empty $action\_list$
$prior\_p \leftarrow shortest\_path[0]$
for $current\_p$ in $shortest\_path[1:]$ **do**
    $path\_rotation \leftarrow$ compute_path_rotation($current\_p$, $prior\_p$)
    if $path\_rotation \neq agent\_rotation$ **then**
        Append appropriate rotation action(s) to $action\_list$ ▷ RotateRight or RotateLeft
        $agent\_rotation \leftarrow path\_rotation$
    end if
    Append "MoveAhead" to $action\_list$
end for
$object\_rotation \leftarrow$ compute_object_rotation($target\_object, shortest\_path[-1]$)
if $object\_rotation \neq agent\_rotation$ **then**
    Append final rotation action(s) to $action\_list$ ▷ Adjust the view for the target object
end if
Append "Done" to $action\_list$
return $action\_list$

MOVEAHEAD
Template: 1) In the direction of my rotation, **[agent_rotation]** degrees (**[cardinal_direction]**), the difference to the target object is **[position_diff]** m. I need to move further **[cardinal_direction]**. 2) There is no obstacle in front of me in recent images. 3) MoveAhead
Example: 1) In the direction of my rotation, 90 degrees (east), the difference to the target object is 0.5m. I need to move further east. 2) There is no obstacle in front of me in recent images. 3) MoveAhead
DONE
Template: 1) My position and rotation are equal to the recommended one. 2) I can see the target **[obj_type]** in the image of the current state. 3) Done
Example: 1) My position and rotation are equal to the recommended one. 2) I can see the target label marker in the image of the current state. 3) Done

Table 11: Response templates we used to build CoT explanation traces for MOVEAHEAD and DONE.

Action	w/o Postproc		w/ Postproc	
	# Num	% Prop	# Num	% Prop
MOVEAHEAD	221,598	77.94%	57,760	49.32%
ROTATELEFT	19,596	6.89%	18,412	15.72%
ROTATERIGHT	19,527	6.87%	18,403	15.72%
DONE	23,610	8.30%	22,529	19.24%

Table 12: The distribution of collected actions before and after post-processing. The original dataset is very imbalanced since most of the actions are MOVEAHEAD. Then, we downsample the MOVEAHEAD actions with a rate of 0.25. We also filtered the conflicting data.

First scenario: distance difference becomes zero
Template: 1) In the direction of my rotation, **[agent_rotation]** degrees (**[cardinal_direction]**), the difference to the recommended position is 0.00m. Thus, I need to move in another direction, where the difference is **[other_position_diff]** m, and the rotation is **[other_agent_rotation]** degrees. 2) Obstacles don't affect rotation. 3) RotateRight/RotateLeft
Example: 1) In the direction of my rotation, 180 degrees (south), the difference to the recommended position is 0.00m. Thus, I need to move in another direction, where the difference is 1.25m, and the rotation is 90 degrees. 2) Obstacles don't affect rotation. 3) RotateLeft
Second scenario: Obstacles
Template: 1) In the direction of my rotation, **[agent_rotation]** degrees (**[cardinal_direction]**), the difference compared to the target object is **[position_diff]** m. 2) There are obstacles in front of me, as shown in current images. I need to rotate in another direction. In the other direction, the difference is **[other_position_diff]** m, and the rotation is **[other_agent_rotation]** degrees. 3) RotateRight/RotateLeft
Example: 1) In the direction of my rotation, 180 degrees (south), the difference compared to the target object is 1.50m. 2) There are obstacles in front of me, as shown in current images. I need to rotate in another direction. In the other direction, the difference is 1.25m, and the rotation is 270 degrees. 3) RotateRight
Third scenario: View Adjustion
Template: 1) My position is the same as the recommended one: **[grid_obj_pos]**. However, my rotation is **[agent_rotation]** degrees, facing **[cardinal_direction]**. I need to adjust the rotation to center the target within its field of view. 2) Obstacles don't affect rotation. 3) RotateRight/RotateLeft
Example: 1) My position is the same as the recommended one: (0.50, 1.25). However, my rotation is 90 degrees, facing east. I need to adjust the rotation to center the target within its field of view. 2) Obstacles don't affect rotation. 3) RotateRight

Table 13: Response templates we used to build CoT explanation traces for three common rotation scenarios.

Number	**Valid**			**Test**			**All**		
	SR	**SPL**	**SEL**	**SR**	**SPL**	**SEL**	**SR**	**SPL**	**SEL**
2 images	50.00	41.64	40.23	50.00	40.71	41.03	50.00	41.17	40.63
4 images	**57.41**	**47.84**	**47.90**	**54.94**	**44.45**	45.83	**56.17**	**46.15**	**46.86**
6 images	45.37	37.61	35.37	50.31	40.88	38.70	47.84	39.25	37.03
8 images	49.07	40.82	41.52	54.01	43.61	**46.22**	51.54	42.22	43.87

(a) Performance of the default prompt.

Number	**Valid**			**Test**			**All**		
	SR	**SPL**	**SEL**	**SR**	**SPL**	**SEL**	**SR**	**SPL**	**SEL**
2 images	46.30	38.54	38.93	52.16	42.13	44.76	49.23	40.34	41.84
4 images	54.63	45.02	**46.48**	54.94	44.04	**47.40**	54.78	44.53	**46.94**
6 images	49.07	40.82	38.88	51.23	41.56	40.20	50.15	41.19	39.54
8 images	**55.56**	**45.81**	43.40	**57.41**	**46.39**	44.40	**56.48**	**46.10**	43.90

(b) Performance of the difference-equation prompt.

Table 14: The investigation of the hyperparameter: image number. We provide different numbers of recent visual observations of the embodied agent. Besides the default prompt we use, we also evaluate the difference-equation prompt. We bold the best performance.

Number	Valid			Test			All		
	SR	SPL	SEL	SR	SPL	SEL	SR	SPL	SEL
4 steps	46.30	38.66	39.82	45.99	37.45	39.60	46.14	38.05	39.71
8 steps	**57.41**	**47.84**	**47.90**	**54.94**	**44.45**	**45.83**	**56.17**	**46.15**	**46.86**
12 steps	42.59	35.92	36.43	50.93	41.15	43.20	46.76	38.53	39.81
16 steps	51.85	43.01	42.72	53.40	43.03	44.08	52.62	43.02	43.40

Table 15: Hyperparameter investigation. Our agents receive various numbers of recent actions and positions within the default prompt.

Number	Valid			Test			All		
	SR	SPL	SEL	SR	SPL	SEL	SR	SPL	SEL
4 steps	45.37	37.76	40.04	52.47	42.87	46.45	48.92	40.31	43.25
8 steps	54.63	45.02	46.48	54.94	44.04	47.40	54.78	44.53	46.94
12 steps	51.85	42.80	44.06	**60.19**	**48.10**	**51.63**	56.02	45.45	47.84
16 steps	**56.48**	**46.39**	**47.89**	59.26	47.52	50.82	**57.87**	**46.95**	**49.36**

Table 16: Hyperparameter investigation. We provide different numbers of recent actions and positions to the embodied agent. We use the **difference-equation prompt** in this table. The best performance is bolded.

% Data	Valid			Test			Test Diff w GPT-4o		
	SR	SPL	SEL	SR	SPL	SEL	Δ_{SR}	Δ_{SPL}	Δ_{SEL}
20	37.04	30.86	30.22	38.89	31.40	32.13	↑0.62	↓0.34	↑4.21
40	33.33	27.95	27.13	38.27	31.00	30.79	↓0.00	↓0.74	↑2.87
60	49.07	40.68	42.01	52.12	42.25	44.40	↑13.85	↑10.51	↑16.48
80	50.00	41.46	43.36	49.69	40.26	42.49	↑11.42	↑8.52	↑14.57
100	**57.41**	**47.84**	**47.90**	**54.94**	**44.45**	**45.83**	↑16.67	↑12.71	↑17.91

Table 17: Few-shot learning ability. We test our agent with different proportions of training data with the default prompt.

% Data	Valid			Test			Test Diff w GPT-4o		
	SR	SPL	SEL	SR	SPL	SEL	Δ_{SR}	Δ_{SPL}	Δ_{SEL}
20	37.96	32.01	36.72	32.10	26.45	31.15	↓6.17	↓5.29	↑3.23
40	38.89	32.76	38.14	33.33	27.07	31.94	↓4.94	↓4.67	↑4.02
60	**62.96**	**51.47**	**52.19**	**58.95**	**47.34**	**49.53**	↑20.68	↑15.60	↑21.61
80	57.41	46.79	45.73	57.10	45.98	46.58	↑18.83	↑14.24	↑18.66
100	54.63	45.02	46.48	54.94	44.04	47.40	↑16.67	↑12.30	↑19.48

Table 18: The evaluation of the few-shot learning ability of our agent NATVLM. We test our agent with different proportions of sampled episodes in each room. We compare the results with the GPT-4o and test the **difference-equation prompt** in this table.

are mostly lower than 30%. In contrast, the SPL score of NATVLM is 34.11%, 10 points higher than theirs.

Meanwhile, we want to emphasize that those LLM or VLM-based complicated navigation framework (Ramrakhya et al., 2023; Majumdar et al., 2022; Zhou et al., 2023; Yokoyama et al., 2024a; Yin et al., 2024; Long et al., 2024) usually involve a lot more information than NATVLM. For example, our method only takes ego-centric RGB images as input, while those methods also get depth information, scene graphs, local policy navigation tools, extra VLMs (like GPT-4) besides backbone VLM, semantic segmentation, and panoramic field of views. This makes the comparison **inherently unfair**, as their methods rely on significantly more resources and additional modalities.

Meanwhile, we also **do not include** those complicated frameworks in other experiments due to the following reasoning: (1) They fall outside the evaluation scope of the current LVLMs; (2) They utilize different simulation platforms, such as Habitat (Szot et al., 2021), rather than AI2THOR, which introduces significant compatibility issue with AI2THOR; (3) As abovementioned, they also introduce a lot of extra information beyond than ego-centric RGB, making the evaluation unfair.

Methods	SPL	SR
Fine-tuned		
ProcTHOR (Deitke et al., 2022)	31.8	54.4
PIRLNav (Ramrakhya et al., 2023)	27.1	64.1
Zero-Shot		
ProcTHOR (zero-shot)	7.7	13.2
ZSON (Majumdar et al., 2022)	12.6	25.5
ESC (Zhou et al., 2023)	22.3	39.2
VLFM (Yokoyama et al., 2024a)	30.4	52.5
SG-Nav (Yin et al., 2024)	24.8	53.9
InstructNav (Long et al., 2024)	20.9	**58.0**
NATVLM (Our)	**34.1**	41.5

Table 19: Comparison of different models on HM3D, including fine-tuned and zero-shot methods.

D Implementation Details

We train our agent NATVLM from Idefics 2 on 8 NVIDIA A100 GPUs using the Megation-LM framework (NVIDIA, 2021). All parts of the Idefices 2 are fine-tuned, including the LLM, vision encoder, and modality projector. We load the model in BF16 and fine-tune it for one epoch with the learning rate and batch size of 2e-5 and 64, respectively. The best checkpoint is selected according to the sum of all metrics on the validation set. The image size sampled from the AI2THOR is 300 $\times$ 300. For the baselines, we use the same instructions as our agent and ask them to predict the next action directly. Similarly, we also provide the history of the recent 8 steps (i.e., actions and agent positions) and the visual observation of the recent 4 steps. The exceptions are blind LLMs and Llava 1.5, which can handle zero and one image, respectively. We access the closed-source LVLMs via the OpenAI API[3] with the specific versions of `gpt-4-vision-preview` and `gpt-4o-2024-08-06`.

Here, we discuss the computational cost. For training, we use 8 $\times$ A100 GPUs to train our model. After adding the CoT traces, the training time increases from 10 hours to 17 hours, which is acceptable. Meanwhile, we use 1 A100 GPU to deploy the model. The generation time increases from 0.28s to 1.03s per query when we add the CoT traces. In practice, we can use more GPUs to speed up the inference. For example, RT-2 (Brohan et al., 2023) uses a multi-TPU cloud service to achieve a frequency of 5Hz for a 5B model and 3Hz for a 55B model. The computational cost, both in training and inference, is reasonable in our experiments.

[3]https://platform.openai.com/docs/api-reference

E More House Examples

In this section, we present more houses built in our DIVSCENE dataset in Figure 9.

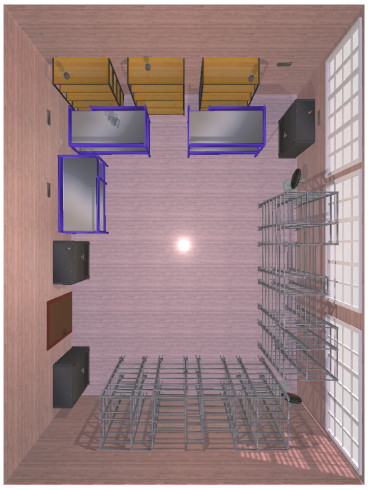
E.1 Comparison with ProcTHOR

In Figure 8, we compare houses from our dataset and ProcTHOR with the same number of rooms (8 rooms). Obviously, our scene is more complex with more objects. Quantitively, our scene contains 466 objects, and the scene from ProcTHOR contains only 74 objects.

(a) a community center with versatile flooring

(b) a house from ProcTHOR

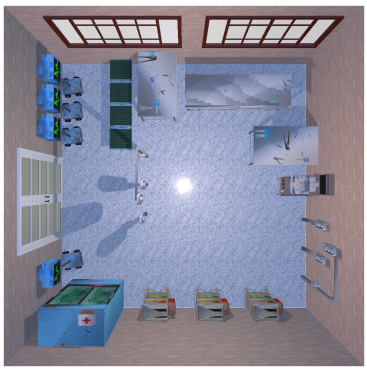
Figure 8: Comparing our houses with ProcTHOR.



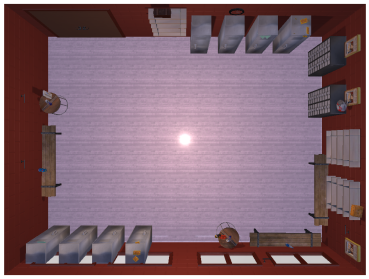
(a) a warehouse with large windows

(b) a meeting room with artistic paintings

(c) a toy store with a magical kingdom theme.



(d) an operating room used by surgeons and nurses featuring light-colored tiles on the walls.



(e) an industrial locker room

(f) a community center with a versatile and inclusive theme featuring a spacious room size.

Figure 9: Examples of different houses in DIVSCENE.