IMPROVING GROUNDING ABILITY OF VISION AND LANGUAGE NAVIGATION AGENTS

By

Yue Zhang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science — Doctor of Philosophy

2025

ABSTRACT

Understanding and following human instructions is crucial for intelligent agents to interact with humans in real-world environments. One formal problem setting designed to facilitate advancing this direction of research is Vision-and-Language Navigation (VLN). VLN requires the agent to carry out a sequence of actions in a photo-realistic simulated indoor environment in response to natural language instructions. Although significant progress has been achieved in this direction, navigation agents still have challenges understanding instructions and accurately grounding them in their visual perception.

To elaborate, the challenges include 1) lacking explicit learning of spatial semantics in both text and vision modalities, 2) difficulties in handling ambiguous instructions and the lack of explainability, and 3) gaps in language understanding for navigation in realistic environments, such as continuous and 3D spaces. In this thesis, we develop new techniques to address these challenges. *First*, we explicitly model spatial semantics to improve the navigation agent's grounding by incorporating navigation progress, the alignments between textual landmarks and visual objects, and the corresponding spatial directions. Besides, we design specialized modules to capture distinct semantic aspects through corresponding pre-training tasks, enabling the effective acquisition of the respective skills. *Second*, to help agents deal with ambiguous instructions, we introduce a translator to convert the original ambiguous instructions into easy-to-following instructions considering recognizable and distinctive landmarks. The designed translator bridges the gap between the instruction given by humans and the agent's visual perception ability. Furthermore, to improve the explainability of the decisions made by the agent, we introduce a language generator for the navigation agent to equip it with the ability to generate explanations about navigation progress, navigation difficulties, and observed visual objects in the selected target view. Such explanations enable the agent to explain the situation from its own perspective, enhancing its ability to interact with humans effectively. *Third*, to advance navigation in a more realistic setting, we contribute to language grounding in continuous and 3D environments. For navigation in continuous environments, we introduce a dual-action-perception module that integrates a low-level action decoder, jointly trained with high-level action prediction. This design enables the VLN agent to learn and ground the selected visual view to the corresponding low-level controls. Additionally, in 3D environments, we develop techniques to enhance the agent's situated spatial understanding, further improving its navigation capabilities in 3D scenarios. We evaluate our proposed methods across different commonly-used navigation benchmarks and provide comprehensive quantitative results and qualitative analysis. The experimental results demonstrate the effectiveness of our explicit grounding modules, the proposed pre-training tasks, and the synthesized data incorporating recognized and distinctive landmarks, significantly enhance navigation performance, generalizability, and language grounding ability. Additionally, our novel architectures designed for continuous and 3D environments push the boundaries of navigation agent research to real-world scenarios. Notably, these advancements contribute to improved interpretability of the agent's decision-making process, offering deeper insights into the rationale behind its navigational actions.

Dedicate to my family.
For their unwavering support, endless encouragement, and unconditional love.

ACKNOWLEDGMENTS

My journey toward earning a doctoral degree at Michigan State University has been an incredibly precious experience, filled with countless cherished memories. Throughout this period of academic pursuit, I have not only deepened my knowledge and expanded my research horizons but also gained invaluable experiences that have profoundly shaped my intellectual growth and career aspirations.I am sincerely grateful to all those who have encouraged, challenged, and inspired me along the way, making this journey both meaningful and rewarding.

First and foremost, I want to extend my gratitude to my advisor, Dr. Parisa Kordjamshidi, for her unwavering support and encouragement throughout my Ph.D. journey. Research is a path filled with challenges and setbacks, and there were moments felt disheartening. However, Dr. Kordjamshidi's trust in me and her steadfast encouragement gave me the strength to persevere. Beyond her support, I deeply admire her exceptional work ethic and unwavering dedication to research. Her commitment to maintaining the highest standards has profoundly shaped my own approach to conducting meaningful and impactful work. Under her guidance, I have not only grown as a researcher but have also learned the importance of persistence, critical thinking, and intellectual curiosity. I am truly grateful for her mentorship and the invaluable lessons she has imparted throughout this journey.

Next, I want to thank my comittee members, Arun Russ, Xiaobo Tan, Yu Kong, and Pang-Ning Tan, for their insightful feedback on my thesis. Their guidance and expertise have been invaluable in refining my research. I am also deeply thankful for my incredible labmates and collaborators. I feel truly fortunate to be a member of the Heterogeneous Learning and Reasoning (HLR) Lab, where I have received constant support and encouragement. I especially want to sincerely thank Dr. Quan Guo for his guidance at the beginning of my Ph.D. journey—his mentorship provided me with essential knowledge and direction. Additionally, I am grateful to Dr. Chen Zheng for his valuable support, both in my research and in daily life.

This degree would not have been possible without the unwavering support of my parents, Mr. Xiuzhong Zhang and Mrs. Meixiang Zhang, my sister, Dr. Bin Zhang, my brother, Chao Zhang, and the rest of my family. My parents' steadfast encouragement, sacrifices, and unwavering belief in my abilities have been the foundation of my academic journey. I am deeply grateful for their dedication to my education, which has shaped me both intellectually and personally. Their endless love, patience, and guidance have given me the strength to overcome challenges and the courage to pursue my dreams. Without them, this achievement would not have been possible, and I am forever thankful.

Finally, I wish to express my deepest gratitude to my husband, Xiao Guo, whose presence has been a constant source of strength and inspiration in my life. His unwavering support has anchored me through every challenge, offering not only encouragement but also the reassurance that I am never alone on this journey. I am especially grateful for his belief in me, even during times when I struggled to believe in myself. His steadfast commitment and unconditional love mean the world to me. I am truly fortunate to walk this path with him by my side.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Robot agents capable of interacting with humans provide significant benefits to humans daily life. Understanding and following natural language instructions is critical for an intelligent agent to interact with humans and the physical world. One of the designed formal problem settings to make advancements in this research direction is Vision-and-Language Navigation (VLN) [4]. VLN requires an agent to carry out a sequence of actions in a photo-realistic simulated indoor environment according to the natural language instructions and observed visual environment. To conduct this task, the agent should be equipped with three abilities: understanding linguistic semantics in the instructions, perceiving the visual images in the environment, connecting the two modalities, and reasoning over both [152, 114]. Unlike other Vision and Language (VL) tasks, such as Visual Question Answering (VQA), the VLN task is more challenging since the visual information dynamically changes during navigation. Besides, VLN problem setting can be seen as a Partially Observable Markov Decision Process (POMDP), where the agent relies heavily on historical information to make the next action decision.

To be specific, as shown in Fig 1.1, at each navigation step, the agent observes the panoramic views of the current visual environment and selects an action based on the given instructions. The task is set in a single-turn scenario, where instructions provide initial guidance and remain unchanged during the navigation. There are two distinct VLN settings: the Discrete Environment Setting (VLN-DE) [4] and the Continuous Environment Setting (VLN-CE) [54]. Two settings employ different simulators: VLN-DE uses Matterport3D [7], while VLN-CE uses Habitat 3D [90]. Another primary difference between these two settings lies in their action space. In VLN-DE, the action space is the selection of images/views, where the agent selects candidate views from panoramic views based on the connectivity graph. In VLN-CE, the action space is low-level controls (*LEFT/RIGHT/FORWARD/STOP*), which are closer to real-world robot operations.

We use the following formulation of this problem in this thesis: given an instruction with a

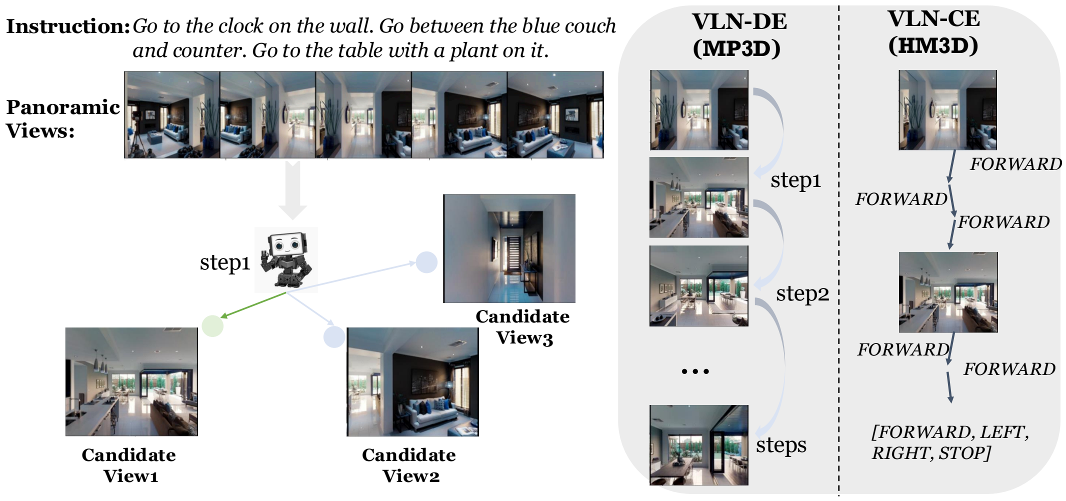


Figure 1.1 VLN Task Demonstration.

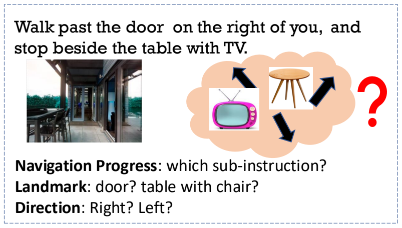
sequence of word tokens, denoted as $X = \{x_1, x_2, \cdots, x_L\}$, where L is the number of tokens, the agent observes a panoramic view including 36 viewpoints[1] at each navigation step t. There are n candidate viewpoints that the agent can navigate to in panoramic images, denoted as $I = \{I_1, I_2, \cdots, I_n\}$. The agent infers an action a_t that transfers the agent from state s_t to a new state s_t. The state consists of navigation history and current spatial position. The process can be formulated as follows:

$$s_t, a_t = \mathtt{NAV}(s_{t-1}, X, I), \tag{1.1}$$

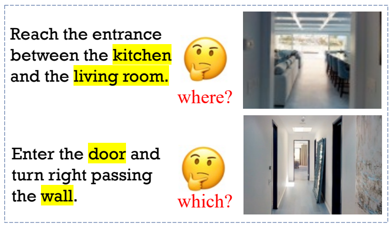
where NAV is the navigation agent. The agent needs to execute a sequence of actions close to a goal destination. The navigation terminates when the navigation agent selects *STOP* action or reaches a pre-defined maximum step.

The VLN task has attracted significant attention leading to many methods being proposed to advance this direction of research [140]. The task is essentially formulated as a Sequence-to-Sequence problem to generate an action sequence. The types of techniques have evolved dramatically during the last few years. The baseline was based on Long Short-Term Memory (LSTM) [4, 106] when the task was initially proposed, and the current state-of-the-art model relies on Transformer architecture [39, 10]. At present, there is a growing trend towards exploring LLM-based navigation agents [145, 79]. In a word, the VLN task serves as a valuable test bed for evaluating the effectiveness of multimodal reasoning and embodied AI.

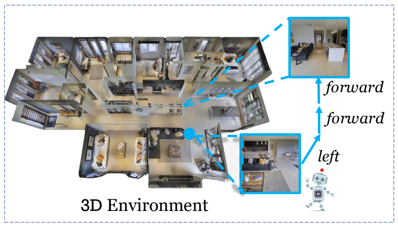
[1] 12 headings and 3 elevations with 30 degree interval.


Walk past the door on the right of you, and stop beside the table with TV.
?
Navigation Progress: which sub-instruction?
Landmark: door? table with chair?
Direction: Right? Left?

(a)


Reach the entrance between the kitchen and the living room.
where?
Enter the door and turn right passing the wall.
which?

(b)



(c)

Figure 1.2 Challenges of Language Grounding in the VLN Task.

1.2 Challenges and Contributions

Although numerous methods are proposed, most of the work focuses mainly on modeling visual information and training strategies. However, our research primarily addresses improving the navigation agent's language understanding and grounding ability. Such ability is important to help the agent comprehend text components in the instruction and align them within the visual environment. We summarize the following three challenges for language grounding in the VLN task and introduce our corresponding solutions in Fig. 1.2, which are also detailed below.

(a) Lacking Explicit Grounding with Entangled Vision and Spatial Understanding. Most of VLN agents primarily rely on attention mechanisms to implicitly learn the correlation between vision and text modalities. However, these approaches often mix different semantic elements from both modalities, leading to an entangled representation that lacks explicit alignment. As shown in Fig. 1.2 (a), it remains challenging to discern whether the failure of navigation comes from the agent incorrectly identifying the progress of the navigation, locating the wrong landmark, or heading in the wrong direction. Distinguishing these different aspects of semantics is important to help the agent better understand instructions and further improve the interpretability of the agent's actions.

Contributions. 1) We propose a method to explicitly align the spatial semantics in the linguistic instructions and the visual environment. We first split a long and complex instruction into spatial configurations, defined as the smallest spatial units containing various spatial components. We then select the landmarks the agent should focus on based on navigation progress and align them with the objects in the visual environment. We also model the spatial relations between landmarks and the agent's position. Our experimental results demonstrate that our explicit modeling helps with both navigation performance and the interpretability of the navigation agent. 2) Unlike previous VLN agents that intertwine the learning of orientation and visual signals, we propose a method that employs specialized modules to learn these signals independently. To achieve this, we introduce specific pre-training tasks to distill more explicit spatial and visual knowledge, which is then effectively utilized within the corresponding modules of the navigation agent. Our modular design interacts with specialized pre-training tasks, enhancing the agent's ability to adapt to downstream navigation. Our results surpass the SOTA model on several navigation benchmarks.

(b) Ambiguous Instructions and Lack of Explainability. Although explicit spatial semantics modeling can help the agent understand instruction, two types of instructions make the grounding very challenging. First, when the instruction contains landmarks that may not be easily recognizable. For example, in Fig. 1.2 (b), identifying the *"kitchen"* and the *"living room"* in the target view may be less straightforward than the *"sofa"* and the *"dining table"*. Another challenge arises when instructions contain landmarks that could potentially apply to multiple targets, such as *"door"* or *"wall"*, which are often observed in every scene. These instructions cause the explicit and fine-grained grounding to be less effective for the VLN task.

Contributions. 1) The first main idea of our work is to introduce a translator module in the VLN agent, which takes the given instruction and visual environment as inputs and then converts them to easy-to-follow sub-instruction representations focusing on two aspects. a) *Recognizable landmarks* chosen based on the navigation agent's visual perception ability. b) *Distinctive landmarks* chosen to help the navigation agent distinguish the targeted viewpoint from the candidate viewpoints. The translator enhances the connection between the given instruction and the agent's observation of the visual environment. We also construct a high-quality synthetic sub-instruction dataset and design specialized tasks for pre-training the translator and navigation agent. We evaluate our method on several navigation benchmarks to show its effectiveness. 2) The translator's design relies on implicit learning, making it challenging to explicitly interpret the ambiguities encountered by the agent. To address this, we use a language model built in our VLN agent and jointly tune it to with VLN task. We aim to generate natural language explanations that describe the agent's difficulties in understanding instructions within the visual environment. Additionally, the explainer provides justifications for the agent's actions based on the navigation process's rationale. To train the explainer, we construct a synthetic dataset that aligns landmarks mentioned in the instructions with visible and distinctive objects in the visual environment. Our empirical results indicate the effectiveness of our approach and achieve SOTA when evaluated on VLN benchmarks.

(c) Language Understanding for Navigation in Continuous and 3D Environment. The ultimate goal of VLN is to develop a navigation agent that can be deployed in real-world robot. While most task settings and simulated environments are discrete, real-world environments are continuous and require agents to develop a 3D understanding for planning low-level actions (as illustrated in Fig 1.2 (c)). Recent VLN-CE (Continuous Environment) research has made significant advancements in building more realistic simulated experimental settings by incorporating continuous environments. However, existing VLN-CE agents exhibit weak grounding abilities, often overlooking the relationship between instructions and corresponding low-level actions within the visual environment. Meanwhile, large language models (LLMs) have shown promise in reasoning over 3D information, making them a compelling choice for enhancing embodied navigation. Despite this, most current 3D-based LLMs lack situated understanding—an essential capability for effective navigation in real-world settings.

Contributions. 1) To strengthen the language grounding ability of the VLN-CE agent, we introduce a dual-action-perception module in which the agent selects high-level viewpoints while generating low-level action sequences simultaneously. The high-level actions serve as guidance, facilitating the agent's understanding of the relationships between low-level actions and navigable areas indicated by high-level actions. Our design enhances the VLN-CE agent's spatial grounding ability to connect actions with visual perception and language understanding. 2) To enhance situated spatial understanding in 3D-based LLMs, we introduce a scalable LLM-generated dataset that incorporates diverse situated spatial information, conditioned on the agent's standpoint and orientation. By fine-tuning existing 3D-based LLMs with our dataset, we significantly improve their ability to comprehend spatial relationships in a 3D world. Moreover, the enhanced spatial understanding shows strong generalization to navigation tasks without training on navigation datasets.

1.3 Thesis Outline

We organize the thesis based on the challenges and contributions mentioned above.

Chapter 2 provides a comprehensive literature review of the aspects of the embodied AI, Vision and Language, and the VLN task.

Chapter 3 introduces two methods for modeling explicit spatial semantics for the VLN agent. First, we present an explicit grounding approach, the **E**xplicit **O**bject **R**elation **A**lignment Agent (**EXOR**), which models spatial information in both instruction and visual environment explicitly. Next, we propose a neural navigation agent, **L**earning **O**rientation and **Vi**sual **S**ignals (**LOViS**), which learns spatial orientation and visual perceptions with disentangled modules. Additionally, we design novel and specialized pre-training tasks to enhance the learning of these modules.

Chapter 4 introduces two methods to address ambiguous instructions. First, we design a translator module for the VLN agent, named **VLN-Trans**, to transfer the original instruction to the easy-to-follow sub-instructions representations focusing on the recognizable and distinctive landmarks based on the agent's visual abilities and observed visual environment. Second, we introduce a hint generator, named **NavHint**, which provides detailed natural language descriptions to assist the navigation agent. NavHint explains the agent's reasoning process by generating the rationale behind its actions during navigation.

Chapter 5 presents our advancements in developing navigation agents aimed at real-world application. We introduce a VLN-CE agent equipped with a dual-action mechanism and a 3D-based LLM model designed for situated spatial understanding. For the VLN-CE agent, we propose a **low-level action decoder** jointly trained with high-level action prediction, allowing the agent to learn and ground the selected visual view into low-level controls. For 3D-based LLMs, we introduce **Spartun3D** (**S**ituated **Spat**ial **Un**derstanding of the 3D World), a scalable dataset designed to enhance situated spatial reasoning in 3D world, thereby advancing navigation in 3D environment.

Chapter 6 provides a comprehensive summary of our research, highlighting key contributions and insights. Additionally, we explore potential future directions, with an emphasis on leveraging large generative models and compositional learning to enhance performance in embodied tasks.

CHAPTER 2

LITERATURE REVIEW

This chapter provides a comprehensive literature review of the background of this research, including Embodied AI, Vision and Language Learning, Vision and Language Navigation datasets, evaluation metrics, backbone architectures, and key approaches.

2.1 Embodied AI

Embodied Artificial Intelligence (Embodied AI) enables AI agents to learn through interactions with their environment from an egocentric perception similar to humans [24]. Different from the traditional AI algorithms to learn from datasets of images, videos or text collected primarily from the Internet, Embodied AI aims to acquire knowledge through interaction with the environment dynamically. Embodied AI has led to significant progress in embodied AI simulators that help replicate the physical world. These simulators work as virtual testbeds to train and test embodied AI agents before deploying them into the real world. The popular embodied AI simulators include MP3D [7], Habitat3D [97, 104], AI2-THOR [50], VirtualHOME [84], iGibson [122], and etc.

Embodied AI simulators have also boosted a series of embodied AI research tasks, including *visual exploration*, *visual navigation*, and *embodied QA* [24]. The tasks increase in complexity as they advance from exploration to QA. In visual exploration, an agent gathers information about a 3D environment through movement and visual perception, done before or concurrently with navigation tasks. The agent is free to explore the environment with a limited number of steps before the start of navigation [3] or builds the map as it navigates in an unseen environment [27, 75, 76]. In visual navigation, an agent navigates in a 3D environment following a natural language instruction [3]. The challenging aspect of visual navigation is it requires agents to make action predictions based on historical visual information and actions. Embodied QA is currently considered the most complicated task in embodied AI research since it needs the agent to possess a wired range of capabilities such as visual recognition, language understanding, question answering, commonsense reasoning, task planning, and goal-driven navigation. A common framework of Embodied QA can be divided into a navigation and a QA task, where the navigation is to explore the environment and the QA module is executed based on the previous paths when the agent decides to stop.

2.2 Vision and Language Learning

In the past few years, Computer Vision (CV) and Natural Language Processing (NLP) have played important roles in deep learning research [126, 31, 101, 33, 119]. In addition to the significant progress in single-modality pre-trained models, there has been an upsurge in this research focused on pre-training large-scale models on both vision and language modalies, called Vision-Language Pre-Training Models (VLMs). Such VLPs are supposed to learn universal cross-modal representations, which are beneficial for achieving strong performance in downstream VL tasks [133, 127, 32, 22, 140, 82].

Generally speaking, given image-text pairs, VLMs employ a text encoder and an image encoder to extract image and text features and then learn the vision-language correlation with pre-training or downstream training objectives. There are two types of mainstream vision and language pre-training architecture. (1) Single-stream [65, 103, 68] fuses the language and vision representations by the joint cross-modal encoder directly. Specifically, VisualBERT [65] utilize segment embedding to indicate input elements from different sources. OSCAR [68] includes object tags detected from the image rather than just focus on image-text pairs. However, single-stream architecture may neglect intra-modality interaction since it directly applies self-attention between different modalities. (2) Double-stream applies intra-modality processing to two modalities separately along with a shared cross-modal encoder. It assumes that the intra-modal interaction and cross-modal interaction are better to be separated to learn the corresponding representations. For example ViLBERT [72] utilizes two transformers to model intra-modality interaction after the cross-modal module. LXMERT [105] uses a self-attention sub-layer after cross-attention to learn internal connections for each modality. ALBEF [64] employs two transformer before cross-attention to decouple the learning of modalities before their interaction.

The VLM pre-training is usually guided by certain vision-language objectives that enable to learn image-text relations from large scale vision and language dataset [89, 123, 125]. For example, CLIP [89] utilizes an image-text contrastive objective to learn the representation that pull the paired images and texts close and pushing others faraway in the embedding space. This is helpful to learn representations that perform zero-shot predictions. Following CLIP, the research mainly focuses on transfer learning to adapt the pre-training VLMs towards various downstream tasks [148, 147, 26, 131], such as the methods of prompt tuning [148], visual adaption [147], etc. There is another research working on knowledge distillation to obtain knowledge from VLMs to downstream tasks for better performance in object detection, semantic segmentation, etc [19, 29, 23].

2.3 Vision and Language Navigation

VLN [3] is a task where agents navigate within the environment by following natural language instructions. The main challenging part of this task lies in the agent's requirement to make action decisions based on visual perception, language understanding, and history memories. Unlike VL tasks like Visual Question Answering (VQA) and image captioning, which typically take a single input question and static images as input, the images in the VLN task dynamically change [130]. The VLN agent needs to ground language to new visual information while considering historical information. The VLN task is crucial for the Intelligent Agent as its broad applications such as autonomous driving, virtual assistants, and augmented reality. In the following sub-sections, we introduce VLN from the aspects of its simulator and dataset, evaluation metrics and solutions.

2.3.1 Simulator and Dataset

Our primary focus is the indoor navigation task, leveraging the Matterport3D [7] and Habitat [97] simulators. Matterport3D [7] contains 10800 panoramic views of 90 scenes, including houses, apartments, hotels, and offices. It supports the agent collecting surrounding visual information, including the simulated RGB and depth images, as well as semantic segmentation. Habitat [97] provides a diverse collection of highly detailed 3D environments, including geometry, texture, and lighting. It offers the visual details for the navigation agent with RGB images, depth maps, and semantic segmentation. With the introduction of the simulators, many navigation datasets were introduced [4, 55, 48, 108]. We mainly work on three VLN datasets: Room-to-Room (R2R) [4], R4R [48] and R2R-CE [54].

R2R is built upon the Matterport3D dataset. The instructions in the R2R are fine-grained navigation commands, such as *"Walking pass between the living room and kitchen, and stop beside the sofa."* This dataset has 7198 paths and 21567 instructions with an average length of 29 words. The whole dataset is partitioned into training, seen validation, unseen validation, and unseen test set. The seen set shares the same visual environments as the training set, while unseen sets contain different environments. There are 4675 trajectories and 340 trajectories in 61 visual scenes for the train set and validation seen set, respectively. Each path is paired with more than 3 instruction. For the validation unseen set, there are 783 trajectories in 11 scenes.

R4R extends the R2R dataset with longer instructions and trajectories by concatenating two adjacent tail-to-head trajectories in R2R. Different from R2R, the trajectories in R4R are less biased as they are not necessarily the shortest path from the start viewpoint to the destination. It also contains three sets: train (61 scenes, $233,613$ instructions), validation seen (61 scenes, $1,035$ instructions), and validation unseen (11 scenes, $45,162$ instructions).

R2R-CE VLN-CE uses the Habitat 3D [104] to render environment observations based on the MP3D dataset [7]. The dataset statistics are the same as the R2R.

2.3.2 Evaluation Metrics

Three main metrics are used to evaluate navigation wayfinding performance [4]: (1) Navigation Error (NE): the mean of the shortest path distance between the agent's final position and the goal destination. (2) Success Rate (SR): the percentage of the predicted final position being within 3 meters from the goal destination. (3) Success Rate Weighted Path Length (SPL): normalizes success rate by trajectory length. Another three metrics are used to measure the fidelity between the predicted and the ground-truth trajectory. (4) Coverage Weighted by Length Score (CLS) [48] (6) nDTW [47]: Normalized Dynamic Time Warping: penalizes deviations from the ground-truth trajectories. (6) Normalized Dynamic Time Warping weighted by Success Rate (sDTW) [47]: penalizes deviations from the ground-truth trajectories and also considers the success rate.

2.3.3 Main Techniques

The VLN baseline model is first proposed by [4] with the R2R dataset that extends the instruction following to the photo-realistic simulated environments. Subsequent studies have emerged with an emphasis on enhancing navigation performance through multi-modal learning [36, 60, 136, 1, 134], map representation learning [41, 9, 2], graph-based explorations [150, 111, 12], data augmentation [118, 61, 57, 56, 58, 117], large language modeling [146, 8, 109, 87], and auxiliary reasoning tasks or pre-training proxy tasks to guide the navigation agent to learn textual and visual representations [151, 10, 34, 88, 137]. In the following sections, we begin by presenting two basic architectures (LSTM-based and Transformer-based) of the VLN agent. Then, we introduce works that focus on enhancing the language grounding and generation ability of the VLN agent.

LSTM-based VLN Agent. The earlier models mostly depend on the LSTM-based based sequence-to-sequence architecture for encoding the text and visual information, establishing the connections with the attention mechanism, and decoding the actions [4, 73, 25]. The encoder is a bidirectional LSTM-RNN with an embedding layer to obtain language representation, denoted as $[s_1, s_2, \cdots, s_l] = BiLSTM(F(< x_1, x_2, \cdots, x_l >)$, where F represents the embedding function. The decoder is also an attentive LSTM-RNN. At each decoding step t of navigation, the agent first attends to the panoramic image representation f^p with the previous hidden context feature $\tilde{h}_{t-1}$. The visual representation of $i-th$ panoramic image is denoted as $f_i^p = [ResNet(I_i^p); d_i]$, which is the concatenation of the ResNet visual features $ResNet(v_i^p)$ and the corresponding 128 dimensional direction encoding d_i. The direction encoding for panoramic images d_i is the replication of $[cos\theta_i, sin\theta_i, cos\phi_i, sin\phi_i]$ by 32 times, where θ_i and ϕ_i are the angles of heading and elevation of ith panoramic image. The attentive panoramic visual feature $\tilde{f}_t^p$ is computed by $\tilde{f}_t^p = SoftAttn(Q = \tilde{h}_{t-1}, K = f_t^p, V = f_t^p)$, and then is used as input to the LSTM of the decoder to represent the agent's current state as,

$$h_t = LSTM([a_{t-1}; \tilde{f}_t^p], \tilde{h}_{t-1}), \tag{2.1}$$

where a_{t-1} is the selected action direction of the previous navigation step, and $\tilde{h}_{t-1}$ is the hidden context after considering the grounded objects.

Transformer-based VLN Agent. Compared with conventional methods, the Transformer-based model in VL tasks show great improvements [105, 13, 72, 68]. The VLN needs to learn the correspondence between language and dynamic visual observation by interacting with the environment. In the past few years, the VLN task has been formulated as a dynamic grounding problem between texts and images. PRESS [59] firstly fine-tunes a pre-trained language model BERT to obtain the text representation. PREVALENT [34] trains a VL Transformer with a large amount of image-text-action triplets to learn cross representations for the navigation task. RecBERT [40] designs a state unit to store history information and train Transformer recurrently for the direct navigation. HAMT [10] proposes to explicitly encode all past observations and actions as history. Also, they improve the performance by changing the fixed vision features to the Vision Transformer, ViT [20].

VLN↻BERT is the most popular backbone of the Transformer-based navigation agent. It is a cross-modal Transformer-based navigation agent with a specially designed recurrent state unit. At each navigation step, the agent takes three inputs: text representation, vision representation, and state representation. The text representation X for instruction W is denoted as $X = [x_1, x_2, \cdots, x_L]$. The vision representation V for candidate viewpoints I is denoted as $V = [v_1, v_2, \cdots, v_n]$. The recurrent state representation S_t stores the history information of previous steps and is updated based on X and V_t at the current step. The state representation S_t along with X and V_t are passed to cross-modal transformer layers and self-attention layers to learn the cross-modal representations and select an action, as follows:

$$\hat{X}, \hat{S}_t, \hat{V}_t = Cross\_Attn(X, [S_t; V_t]), \tag{2.2}$$

$$S_{t+1}, a_t = Self\_Attn(\hat{S}_t, \hat{V}_t), \tag{2.3}$$

we use $\hat{X}$, $\hat{S}_t$, $\hat{V}_t$ to represent text, recurrent state, and visual representations after cross-modal transformer layers, respectively. The action is selected based on the self-attention scores between $\hat{S}_t$ and $\hat{V}_t$. S_{t+1} is the updated state representations and a_t contains the probability of the actions.

Explicit Grounding in the VLN Agent. Mainstream works use Transformer-based models to implicitly capture cross-modality information and demonstrate outstanding navigation performance [40, 34, 30, 10]. There are works modeling the semantic structure explicitly enhances the textual-visual matching [36, 37, 86, 134, 59]. RelGraph [36] builds an implicit language-visual entity relation graph to learn the connection between the text and vision modalities. SpC-NAV [134] first splits the long instructions into spatial configurations [17, 134, 51]. Then, they explicitly align the landmarks and spatial relations in the spatial configuration to the corresponding information in the visual modality. OAAM [86] attempts to decompose the instruction into action and object phrases and relate them to the visual environment to make the final decisions. NvEM [1] extends OAAM to divide the object modules into subject and reference modules and fuse the information from the neighbor views.

Language-Capable VLN Agent. Equipping the navigation agent with the ability to generate textural instructions is one of the primary methods to augment data and improve the agent's generalization ability and explainability. Early works employ the Speaker-Follower framework [25] to produce synthetic VLN instructions. In this framework, a speaker is trained offline using annotated R2R instructions and generates new instructions based on sequences of panoramas along a trajectory. These generated instructions are subsequently employed as augmented data to train the follower. [106] then improves such a Speaker-Follower agent by adding noise into the environment so that the speaker can generate more diverse instructions to further improve the generalizability of the agent. [71] propose to generate cross-connected house scenes as augmented data via mixuping environment to construct difficult paths for the follower and generate the paired instructions as augmentation data. Different from the above-introduced Speaker-Follower methods that integrate the speaker and follower piplines, there are methods working on optimizing two components simultaneously. For example, [21] focus on improving the speaker model to generate higher-quality instruction by directly obtaining feedback from the follower so that the generated instruction is more suitable for the follower. LANA [113] is a language-capable navigation agent which not only executes human-written instructions but also provides route descriptions to humans at the same time. [142] propose an instruction-trajectory compatibility model that operates without reference instruction to improve instruction evaluation.

CHAPTER 3

EXPLICIT SPATIAL UNDERSTANDING AND GROUNDING

3.1 Introduction

Many VLN agents have been developed to establish the connection between text and vision modalities using an attention mechanism to relate the tokens from a given instruction to the images in a panoramic photo [4, 25, 73, 124]. While these models enhance navigation performance, there is no clear evidence that the agent can effectively align components of the visual environment with the natural language instructions [37]. Surprisingly, prior research [45] has shown that successful navigation is still possible even in the absence of visual information, suggesting that these models may not rely on multimodal grounding to make action decisions. This observation highlights the critical need for a more comprehensive investigation into the grounding capabilities of VLN agents, particularly in their ability to establish explicit correspondences between semantic elements in navigation instructions and their visual counterparts in the environment.

Two fundamental abilities are crucial for a navigation agent: *spatial reasoning* and *visual perception*. For example, *spatial reasoning* enables the agent to interpret directional instructions such as "*90-degree left-turn*" or "*on your right*". *Visual perception* allows the agent to recognize and identify landmarks mentioned in the instructions, such as "*walk to the sofa*" or "*pass the table*". To effectively integrate these capabilities, the agent must align motion-related and landmark-related tokens with their corresponding visual representations. This requires understanding spatial relationships to head to the accurate direction and associating objects in the environment with their textual references. In this chapter, we introduce two neural navigation agents designed to enhance a VLN agent's grounding ability from these two perspectives.

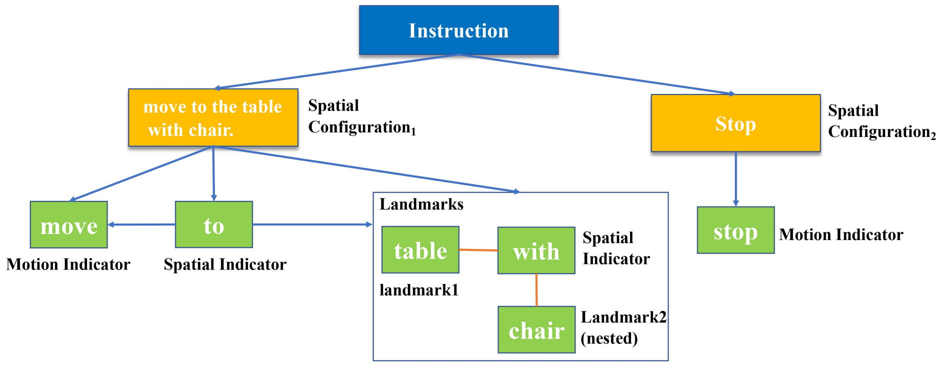
The first navigation agent, named *Explicit Object Relation Alignment Agent (EXOR)* [136] is developed to explicitly align the spatial semantics between linguistic instructions and the visual environment. Specifically, we first split the long instruction into spatial configurations [17, 134], and then we select the important landmarks based on such configurations. After that, in the visual environment, we retrieve the most relevant objects according to their similarity with the selected landmarks in the instructions. Moreover, we obtain **textual spatial relation encoding** to model the spatial relations between the agent and landmarks in the textual instructions, and use **visual spatial relation encoding** to represent the relation between agent and the image in the visual environment. We then establish a mapping between the two encodings to achieve a better alignment. Finally, we use the representations of the aligned objects and spatial relations to enrich the vision representations.

The second navigation agent, called *Learning Orientation and Visual Signals (LOViS)* [137], has different modules to select actions based on orientation and vision perspectives separately. Moreover, we design specific pre-training tasks to distill spatial and visual knowledge independently, which is better utilized in the corresponding modules in our navigation agent. This is different from the majority of methods employing pre-training tasks without considering the needs of the target downstream tasks. Our modular design interacts with modular pre-training, guiding the agents to generate specialized representations which can be better adapted to the downstream tasks.

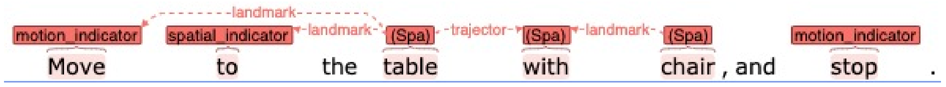
We evaluate our method on the R2R benchmark, and conduct comprehensive ablation studies to further validate the effectiveness of our proposed grounding components. Additionally, we provide qualitative examples to illustrate how our agents leverage different semantics to make action decisions. In summary, our **contributions** are summarized as follows:

1.We focus on different semantic aspects of instructions, particularly visual perception and spatial reasoning. We explicitly model these two aspects from both the instructions and the visual environment, and align them to enhance the agent's navigation performance and the interpretability of its actions.

2. We design two separate modules to capture the orientation and visual information signals for the VLN agent. This enables the agent to select an action more effectively by leveraging both information sources. We design new pre-training tasks to emphasize (a) learning spatial reasoning and grounding the orientation information in the environment; (b) learning visual perception and grounding landmark mentions in the environment. These pre-training representations are utilized in the corresponding modules in the navigation model.



(a) Spatial Configuration Scheme



(b) Spatial Configuration Annotation

Figure 3.1 Spatial configuration example.

3. Our experimental results demonstrate that our explicit modeling enhances the VLN agent's grounding ability and improves overall navigation performance.

3.2 EXOR: Explicit Object Relation Alignment

Fig 3.2 shows the EXOR model architecture. The model has four sub-modules, (1) Spatial Configuration (2) Select top-k landmark selection (3) Landmark-Object alignment (4) Landmark-Object Spatial Relation relation alignment. The text highlighted in green and yellow in (1) shows motion indicators and landmarks, respectively. The red arrow in (4) is the initial agent heading (i.e. orientation).

3.2.1 Spatial Configuration

A spatial configuration is the smallest linguistic unit that describes the location/trans-location of an object with respect to a reference or a path that can be perceived in the environment. It contains fine-grained spatial roles, such as motion indicator, landmark, spatial indicator, trajector. Essentially, each spatial configuration forms a sub-instruction in our setting. Fig. 3.1 shows an example of splitting an instruction into its corresponding spatial configurations and the extracted spatial roles. The instruction "Move to the table with chair, and stop." can be split into two spatial configurations: "move to the table with chair" and "stop". In configuration1, "move" is the motion indicator; "to" is a spatial indicator; "table" is the landmark. "table with a chair" is a nested spatial configuration of configuration1. The role of "table" is trajector; "with" is a spatial indicator; and "chair" is a landmark. In configuration 2, "stop" is a motion indicator. A spatial configuration is the smallest linguistic unit that describes the location/trans-location of an object with respect to a reference or a path that can be perceived in the environment. It contains fine-grained spatial roles, such as motion indicator, landmark, spatial indicator, trajector. Essentially, each spatial configuration forms a sub-instruction in our setting. Previous research argues representing the semantic structure of the language could improve the reasoning capabilities of deep learning models [17, 143]. There are relevant works modeling the meaning of spatial semantics in probabilistic models [49, 107] and neural models [93, 28]. However, its impact on deep learning models for navigation remains an open research problem.

To obtain the configurations in a navigation instruction, we first split the instructions into sentences. Then we design a parser with rules applied on an off-the-shelf dependency parser[1] to extract all the verb phrases and noun phrases in each sentence. In general, each configuration contains at most one motion indicator. Since we aim to process instructions and look for motions, we split the sentences with the extracted verb phrases as motion indicators to obtain spatial configurations. We do not separate the nested configurations with no motion indicator and keep them attached to the dynamic configurations (i.e. the ones with motion-indicator). As shown in Figure 3.1, "table with chair" is the nested spatial configuration of "move to the table with chair". Here, we only consider the prepositions that are attached to verbs, and merge the spatial indicators and motion indicators such as "move to" and use them together as the motion indicator. After that, we insert a pseudo delimiter token after each configuration and identify their contained noun phrases as landmarks.

3.2.2 Landmark Selection

Landmark phrases in instructions are split into groups according to the spatial configuration. We assign the attention weights of each spatial configuration to all its included landmarks. The attention weights of landmarks are the same once they appear in the same configuration. Then we sort all weighted landmarks and select the top-k important ones for the agent to focus on at each navigation

[1]https://spacy.io/

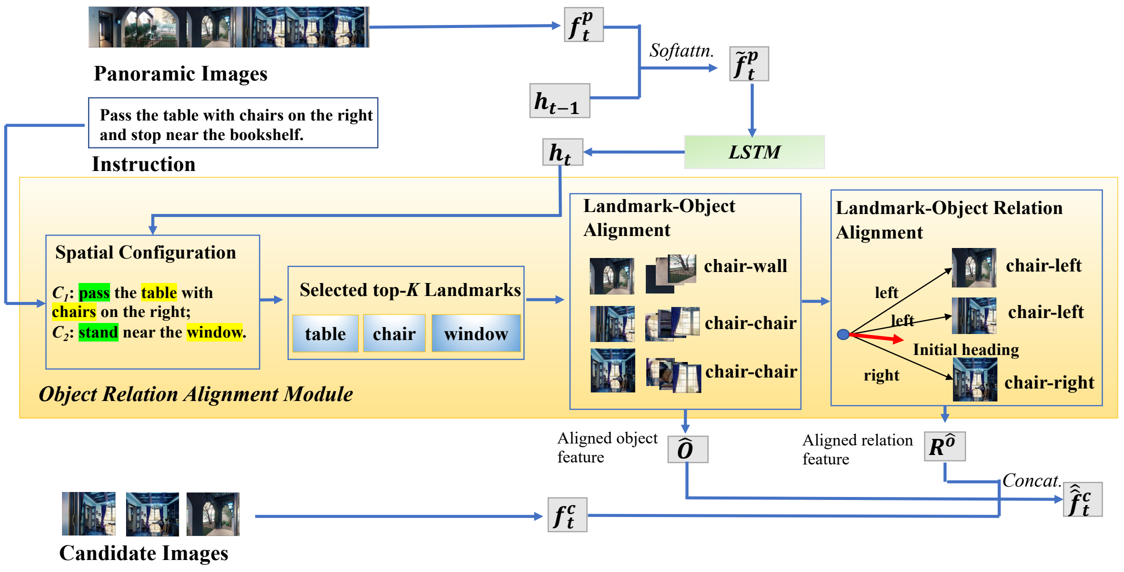


Figure 3.2 Model architecture of EXOR.

step. Formally, each configuration contains n landmarks, denoted as $L =< L_1, L_2, \cdots, L_n >$. The total number of landmarks is $m * n$ in m spatial configurations. After sorting all landmarks based on the spatial configuration weights β, we can obtain top-k selected landmark representations, as $\tilde{L} =< \tilde{L}_1, \tilde{L}_2, \cdots, \tilde{L}_k >$. We obtain the best result when k is 3.

3.2.3 Landmark-Object Alignment

After selecting the top landmarks, the next step is to align them with the corresponding objects in the image. We use Faster-RCNN to detect 36 objects in each image, and the object representation of the i-th image is $O_i = [o_{i,1}, o_{i,2}, \cdots, o_{i,36}]$. We compute the cosine similarity scores between the j-th landmark in top-k landmarks and all objects in the i-th image, and select the object with the highest similarity score as the most relevant object to the j-th landmark, as $\hat{O}_{i,L_j} = max(cos\_sim(\tilde{L}_j, O_i))$. The aligned objects in the i-th image are denoted as $\hat{O}_i = [\hat{O}_{i,L_1}, \hat{O}_{i,L_2}, \cdots, \hat{O}_{i,L_k}]$. We get k aligned objects since we have top-k landmarks. Finally, we concatenate the aligned object representations with the candidate image features f^c. The ith candidate image is represented as $f_i^p = [ResNet(v_i^c); d_i]$. After aligned with the corresponding objects, its representation is updated as $\hat{f}_i^c = [f_i^c; \hat{O}_i^c]$.

3.2.4 Landmark-Object Spatial Relation Alignment

We model both textual spatial relations and visual spatial relations. On the text side, there are mainly three different cases of spatial relations described in the navigation instructions.

- Case 1. Motions verbs, such as “turn left to the table";
- Case 2. Relative spatial relationships between agent and landmarks, such as “table on your left";
- Case 3. Spatial relationships between landmarks, such as “vase on the table".

This work mainly investigates the spatial relations from the agent’s perspective, and we only model the first two cases. We extract "landmark-relation" pairs for each landmark in the instructions (based on syntactic rules). For Case 1, we pair the spatial relation with all landmarks in the configuration. For example, “turn left to the table with the chair", the extracted pairs are {table-left} and {chair-left}. For Case 2, we pair the relation with the related landmark. For example, “go to the sofa on the right.”, the extracted pair is {sofa-right}.

We encode the spatial relations for the landmarks in six bits $[left, right, front, back, up, down]$ as the **textual spatial relation encoding**. Each bit is set to 1 for the landmark if its paired relation has the corresponding relation. On the image side, we encode the same six spatial relations as the **visual spatial relation encoding**. We obtain the spatial relations of objects in the visual environment based on the relative angle, the differences between the agent’s initial direction and the navigable direction. The spatial relations are the same for all objects if they are in the same image.

Formally, for the obtained top-k landmarks, we denote their spatial encoding as $R^{\hat{L}} = [R^{\hat{L}}_1, R^{\hat{L}}_2, \cdots, R^{\hat{L}}_k]$. For the top-$k$ objects aligned with those landmarks, the spatial relations in i-th navigable image are represented as $R^{\hat{O}}_i = [R^{\hat{O}}_{i,1}, R^{\hat{O}}_{i,2}, \cdots, R^{\hat{O}}_{i,k}]$. We compute the inner product of the spatial encoding between top-k landmarks and the top-k aligned objects to obtain the spatial similarity score between the instruction and the i-th image, that is, $sim^R_i = R^{\hat{L}} \cdot R^{\hat{O}}_i$. Then we concatenate each aligned object spatial encoding with the corresponding similarity score, denoted as $\hat{O}_{i,R} = [[R^{\hat{O}}_{i,1}; sim^R_{i,1}], [R^{\hat{O}}_{i,2}; sim^R_{i,2}], \cdots, [R^{\hat{O}}_{i,k}; sim^R_{i,k}]]$. Finally, we further concatenate $\hat{O}_{i,R}$ with

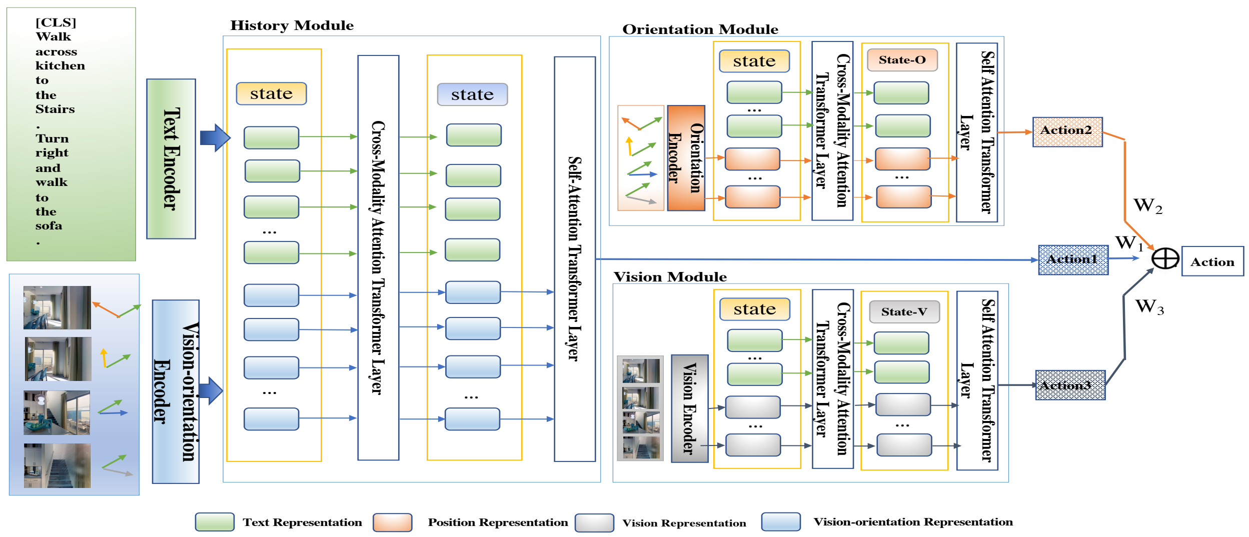

[CLS] Walk across kitchen to the Stairs . Turn right and walk to the sofa .
Text Encoder
Vision-orientation Encoder
History Module
state
Cross-Modality Attention Transformer Layer
state
Self-Attention Transformer Layer
Orientation Module
Orientation Encoder
state
Cross-Modality Attention Transformer Layer
State-O
Self Attention Transformer Layer
Action2
W2
W1
Action1
Action
W3
Action3
Vision Module
Vision Encoder
state
Cross-Modality Attention Transformer Layer
State-V
Self Attention Transformer Layer
Text Representation
Position Representation
Vision Representation
Vision-orientation Representation

Figure 3.3 Model architecture of LOViS.

the candidate image features $\hat{f}_i^c$ which is concatenated with the aligned object features , and i-th candidate images features is updated as $\hat{\hat{f}}_i^c = [\hat{f}_i^c; \hat{O}_{i,g}]$. The updated image representations are then used to make action decisions for the agent.

3.2.5 Action Prediction

After modeling alignment between landmark tokens in the instruction and visual objects, the panoramic image feature is enriched with the aligned visual objects, and candidate image feature is enriched with both visual objects and their spatial relations. Then based on the backbone sequence to sequence agent, the probability of moving to the k-th navigable viewpoint $p_t(a_{t,k})$ is calculated as softmax of the alignment between the navigable viewpoint features and a context-aware hidden output $\tilde{h}_t$, which can be calculate as

$$\tilde{h}_t = tanh(W_{\tilde{c}h}[\tilde{C}; h_t]) \tag{3.1}$$

$$p_t(a_{t,k}) = softmax(\hat{\hat{f}}_i^c W_{\hat{c}} \tilde{h}_t) \tag{3.2}$$

where $W_{\tilde{c}h}$ and $W_{\hat{c}}$ are learnt weights.

3.3 LOViS: Learning Orientation and Visual Signals

LOViS has three main modules: history module, orientation module, and vision module, as depicted in Figure 3.3.

3.3.1 History Module

History Module receives three types of inputs: state representation s_t (see “state” in Figure 3.3), text representation X, and “vision-orientation” representations. To obtain “vision-orientation” representation, we feed the concatenation of vision and orientation representations to a “vision-orientation encoder" (see Figure 3.3). We denote “vision-orientation” representation as $\tilde{VO} = \{\tilde{vo}_1, \tilde{vo}_2, \cdots, \tilde{vo}_k\}$. Then we use cross-modal attention layers and self-attention layers to obtain the cross representation. In cross-modality attention Transformer layers, one modality is used as a query and the other as the key to exchange information as follows,

$$\hat{X}, \hat{s}_t, \hat{VO}_t = Cross\_Attn(X, [s_t; VO_t]), \tag{3.3}$$

where $\hat{X}$, $\hat{s}_t$, and $\hat{VO}_t$ are respectively updated state, text and “vision-orientation” representations after cross modality attention layers. Then state and “vision-orientation” representations are fed into self-attention Transformer layers:

$$s_{t+1}, p_t^h = Self\_Attn([\hat{s}_t; \hat{VO}_t]) \tag{3.4}$$

where s_{t+1} is the updated state after self-attention layers. p_t^h is the self attention score between state representations and “vision-orientation” representations. Note that the refinement of the state representation only happens in the history module.

3.3.2 Orientation Module

Orientation information is vital for the navigation task. For example, the instruction, “*turn left*" can assist the agent to ignore the navigable viewpoints on the right side. In our work, we build an orientation module specifically to encourage the agent to learn the spatial information from the instructions and ground it in the visual environment. Specifically, we linearly project the orientation features O via the “Orientation Encoder” (see Figure 3.3) to obtain its projected representation, denoted as $\tilde{O}$. Then we input the state representation s_t, text representation X, and the projected orientation representation $\tilde{O}$ to the cross-modality attention Transformer layer. The orientation module learns a new state representation, denoted as s_t^o, for orientation information (see “State-O’ in Figure 3.3). For cross-model attention layers, we have:

$$\hat{X}^o, \hat{s}_t^o, \hat{O}_t = Cross\_Attn(X, [s_t^o; \tilde{O}_t]) \tag{3.5}$$

where $\hat{X}^o, \hat{s}_t^o, \hat{O}_t$ are updated state, text, orientation representations after cross modality attention layers in the orientation module. Then we use the state representation enriched with the orientation information to perform self-attention with orientation representations as follows.

$$p_t^o = Self\_Attn([\hat{s}_t^o; \hat{O}_t]) \tag{3.6}$$

where p_t^o is the attention score between state representation and orientation feature.

3.3.3 Vision Module

Connecting mentioned landmarks in the instruction to the scene and objects in the visual environment is also important to the navigation task. In the instruction, "*enter into the bedroom and move close to TV.*", The mentioned landmarks, such as "*bedroom*" and "*TV*", provide apparent clues for the navigation actions. Like the orientation module, we build a vision module to ground the text landmarks in the visual scene and objects. Specifically, we first project vision representations V (refer to the notations in Section 2.3.3) using "Vision Encoder" (see Figure 3.3) to obtain the projected visual representation, denoted as $\tilde{V}$. Then we input the state representation s_t, text representation X, and projected vision representation $\tilde{V}$ to the cross-modal attention and self-attention layers as follows,

$$\hat{X}^v, \hat{s}_t^v, \hat{V}_t = Cross\_Attn(X, [s_t^v; \tilde{V}_t]), \tag{3.7}$$

$$p_t^v = Self\_Attn([\hat{s}_t^v; \hat{V}_t]), \tag{3.8}$$

where s_t^v is the new state representation considering visual information (see "State-V" in Figure 3.3). $\hat{X}^v, \hat{s}_t^v, \hat{V}_t$ are updated state, text, vision representations after cross modality attention layers in the vision module. p_t^v is the attention score between state representation and vision representations.

3.3.4 Action Selection

For each navigable viewpoint, we obtain the self-attention scores from 1) orientation state representation to its orientation representation (orientation module), 2) vision state representation to

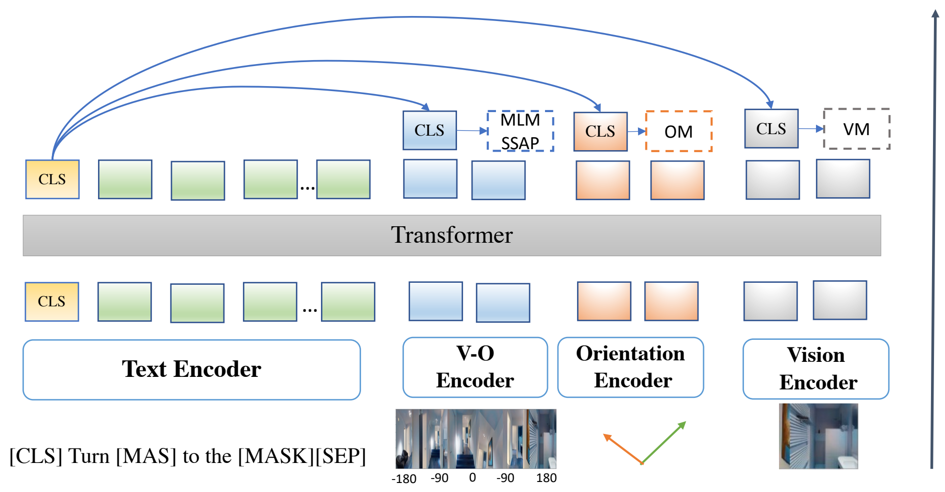


Figure 3.4 Pre-training model with specific pre-training tasks.

the vision representation (vision module), 3) state representation to the combined orientation and visual representations (history module). We combine these scores as follows:

$$p_t = Softmax(W_a[p_t^h; p_t^o; p_t^v]) \tag{3.9}$$

where w_a is the trainable parameter, and p_t denotes the action probability that weights different module scores.

3.3.5 Pre-training Tasks

We follow the model architecture of PREVALENT [34] to obtain the joint cross representations trained on text-image-action triplets, as shown in Figure 3.4. However, the novelty of our pre-training is that we design new tasks named Vision Matching (VM) and Orientation Matching (OM) to pretrain for the vision module and orientation module designed in our navigation agent, as shown in Figure 3.3. Moreover, we improve the existing pre-training tasks of the PREVALENT, Masked Language Modeling (MLM) and Single Step Action Prediction (SSAP), to obtain a more effective initialization of our new architecture. Here, we describe the details of all the pre-training tasks. In the following tasks, we denote each instruction-trajectory pair in training set D as $< w, \tau >$.

Masked Language Modeling (MLM) Different from PREVALENT [34] masking of random tokens, we mask direction and landmark tokens with 8% probability and replace them with special token $[MASK]$. The goal is to recover landmark or orientation tokens w_m by reasoning over the surrounding words $w_{\backslash m}$, and the orientation and visual observation at the each navigation step. We denote the combination of orientation and vision features of panorama views as VO_p. Landmark tokens are usually the token related to scene or objects in the visual environment, such as " *table*", "*sofa*", and "*bedroom*". We extract nouns as landmark tokens based on their pos-tag. The direction tokens usually convey spatial information, such as "*left*", "*right*", and "*forward*". We obtain direction tokens using a direction dictionary built upon R2R training dataset. The loss of MLM is calculated as follows,

$$\mathcal{L}_{MLM} = -\mathbb{E}_{VO_p \tilde{} P(\tau),(w,\tau)\tilde{} D} \log P(w_m | w_{\backslash m}, VO_p), \tag{3.10}$$

Single Step Action Prediction (SSAP) PREVALENT [34] selects actions by mapping the $[CLS]$ representations to the 36 classes directly, which may cause the loose connection between cross-modal representations of the viewpoints and the action space. To address this issue, we use the cross attention distribution from the $[CLS]$ representation to the images in the panoramic view to select an action. We use the cross-entropy loss to compute the loss of SSAP, as follows,

$$\mathcal{L}_{SSAP} = -\mathbb{E}_{OV_p \tilde{} P(\tau),(w,\tau)\tilde{} D} \log P(a | w_{[CLS]}, VO_p), \tag{3.11}$$

where a is the ground-truth action.

Vision Matching (VM) is our novel pre-training specific for initializing our vision module. It predicts whether the current vision information can match with the instruction. In this task, to encourage the agent to focus on learning the connection between landmarks in the instruction and the scene objects in the visual environment, we only use the vision representation (i.e. excluding the heading and elevation) of viewpoint as the input, denoted as v_p. We generate the negative samples by replacing the ground-truth images with an images from another environment. We use the output representation of the $[CLS]$ as the joint representation of textual and visual features to feed to a fully connected layer with a sigmoid function. This layer predicts the matching score $s(w, v_p)$. The loss of SSAP is computed as follows,

$$\mathcal{L}_{VM} = -\mathbb{E}_{v_p \tilde{} \tau,(w,\tau)\tilde{} D} [y \log P + (1-y) \log P)], \tag{3.12}$$

		Val Seen			Val Unseen			Test(Unseen)	
	Method	**SR ↑**	**SPL ↑**	**SDTW ↑**	**SR ↑**	**SPL↑**	**SDTW↑**	**SR ↑**	**SPL ↑**
1	Speaker-Follower [25]	0.54	-	-	0.27	-	-	-	-
2	Env-Drop [106]	0.55	0.53	-	0.47	0.43	-	-	-
3	Env-Drop* [106]	0.63	0.60	0.53	0.50	0.48	0.37	0.50	0.47
4	OAAM* [86]	**0.65**	**0.62**	0.53	**0.54**	0.50	0.39	**0.53**	**0.50**
5	Entity-Relation [36]	0.62	0.60	**0.54**	0.52	0.50	**0.46**	0.51	0.48
6	SpC-NAV [134]	**0.65**	0.61	-	0.45	0.42	-	0.46	0.44
7	EXOR	0.60	0.58	0.53	0.52	0.49	**0.46**	0.49	0.46

Table 3.1 Experimental results for EXOR compared to LSTM-based VLN agents.

where $P = s(w, v_p)$, and $y \in \{0, 1\}$ indicates whether the sampled viewpoint-instruction pair is matching.

Orientation Matching (OM) is the second novel pre-training task designed to learn the orientation representations. We propose to predict the current orientation based on the instruction and the initial orientation. As described before, the orientation feature O_p is the combination of the heading α and elevation β. We use the output representation of $[CLS]$ as the joint representation of instruction and orientation. Then we feed this to a fully connected layer to predict 4-bits of orientation features. The loss of OM is computed as follows,

$$\mathcal{L}_{OM} = -\mathbb{E}_{o_p \tilde{} \tau,(w,\tau) \tilde{} D} \log p(O'|w_{[CLS]}, O_p), \tag{3.13}$$

where O' is the ground-truth orientation feature. The full pre-training objective is

$$\mathcal{L}_{pre-train} = \mathcal{L}_{MLM} + \mathcal{L}_{SSAP} + \mathcal{L}_{VM} + \mathcal{L}_{OM}. \tag{3.14}$$

3.4 Experiments

3.4.1 Experimental Results

Table 3.1 shows the performance of EXOR compared with LSTM-based VLN agents on unseen validation and test set. Notably, EXOR achieves significantly improved navigation performance compared to the baseline SpC-NAV, which also models explicit grounding between text and vision modalities. This demonstrates that EXOR not only enhances navigation capabilities but also strengthens grounding abilities, leading to more effective alignment between linguistic and visual

		Val seen			Val Unseen			Test(Unseen)		
	Method	**NE ↓**	**SR ↑**	**SPL↑**	**NE ↓**	**SR ↑**	**SPL↑**	**NE ↓**	**SR ↑**	**SPL ↑**
1	PRESS [67]	4.39	0.58	0.55	5.28	0.49	0.45	5.49	0.49	0.45
2	PREVALENT [34]	3.67	0.69	0.65	4.71	0.58	0.53	5.30	0.54	0.51
3	AirBERT [30]	2.68	0.75	0.70	4.01	0.62	0.56	4.13	0.62	0.57
4	RecBERT [40]	2.90	0.72	0.68	3.93	0.63	0.57	4.09	**0.63**	0.57
5	HAMT [10]	-	0.69	0.65	-	0.64	0.58	-	-	-
6	RecBERT	2.99	0.71	0.66	4.03	0.61	0.56	4.35	0.61	0.57
7	Our pretrain + RecBERT	2.90	0.74	0.69	3.75	0.63	0.58	4.20	**0.63**	0.57
8	Our pretrain + LOViS (our model)	**2.40**	**0.77**	**0.72**	**3.71**	**0.65**	**0.59**	**4.07**	**0.63**	**0.58**

Table 3.2 Experimental results for LOViS compare to Transformer-based VLN agents.

	Val Seen						**Val Unseen**					
Method	**NE↑**	**SR↑**	**SPL↑**	**CLS↑**	**nDTW↑**	**sDTW↑**	**NE↓**	**SR↑**	**SPL↑**	**CLS↑**	**nDTW↑**	**sDTW↑**
EnvDrop* [106]	-	0.52	0.41	0.53	-	0.27	-	0.29	0.18	0.34	-	0.09
OAAM [86]	-	0.56	0.49	0.54	-	0.32	-	0.29	0.18	0.34	-	0.11
NvEM [1]	5.38	0.54	0.47	0.51	0.48	0.35	6.80	0.38	0.28	0.41	0.36	0.20
RecBERT [40]	4.82	0.56	0.46	0.50	0.56	0.38	6.48	0.43	0.32	0.41	0.42	0.21
LOViS (our model)	**4.16**	**0.67**	**0.58**	**0.56**	**0.58**	**0.43**	**6.07**	**0.45**	**0.35**	**0.45**	**0.43**	**0.23**

Table 3.3 Experimental Results for comparing LOViS with the baseline Models on R4R dataset.

inputs. EXOR is better than the baseline (Env-Drop) even with their augmented data [106] (Row#3), showing our improved generalizability. Compared with OAAM (row#4), which learns object-vision matching with the augmented data, EXOR gets better SDTW, indicating that our agent can genuinely follow the instructions to the destination. However, Ent-Rel achieves better results than our method.

Table 3.2 shows the experimental results of LOViS compared to other Transformer-based VLN methods on R2R benchmark. In this table, From row#1 to row#5 are Transformer-based navigation agents that largely have improved the performance of the LSTM-based agents, as shown in Table **??**. PREVALENT [34] pre-trains the cross-modal representations with text-image-action triplets and replaces the encoder of Env-Drop [106] to improve its performance. AirBERT [30] is one of the SOTA methods that train a model on a large scale and diverse in-domain detests. RecBERT[40] , our baseline, is also a SOTA method that uses the attention distribution of the history information on navigation candidates to determine the next action. Row#4 is their own reported results in their paper, and row#6 shows our best reproduced results which is consistent with the reported results in [71]. Row#7 and row#8 are the performance of our LOViS model. We first show the effectiveness

		Val Seen			Val Unseen		
	Method	**SR↑**	**SPL↑**	**SDTW↑**	**SR↑**	**SPL↑**	**SDTW↑**
1	Env-Dropout [106]	0.55	0.53	0.49	0.47	0.43	0.37
2	Lan-Obj	0.59	0.55	0.52	0.50	0.48	0.43
3	Lan-Obj+Rel	**0.60**	**0.58**	**0.53**	**0.52**	**0.49**	**0.46**
4	Lan-Obj+Rel_v	0.59	0.56	0.52	**0.52**	0.47	0.44

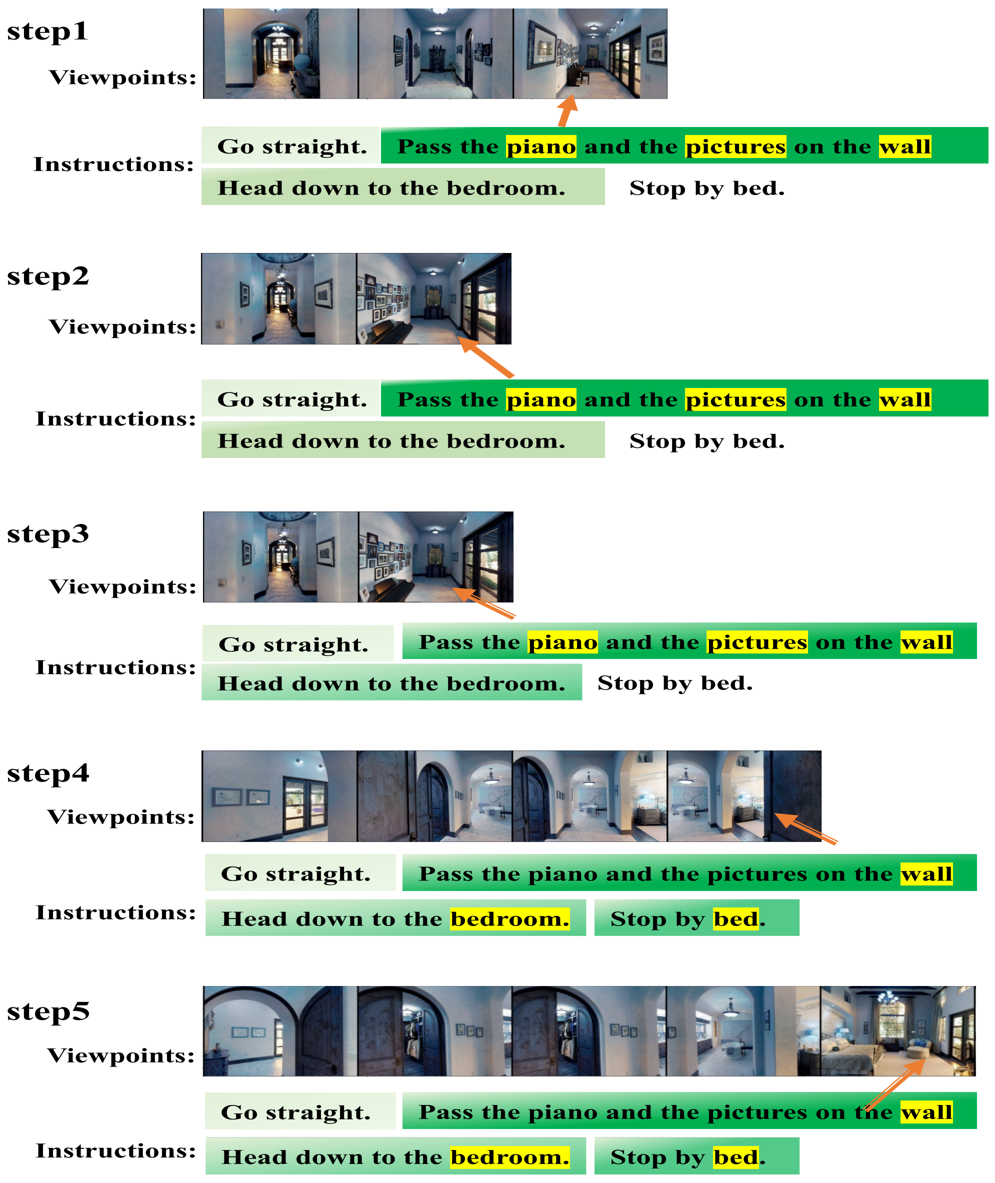
Table 3.4 Ablation Study for EXOR.

of our pre-training on the baseline model. Our pre-training setting can improve the SR and SPL of baseline by about 2% in the unseen validation environment. Moreover, we further improve the performance of the baseline with our designed navigation model and the pre-training setting. The improvement is about 3% of SR and SPL in the seen environment and 2% of SR in the unseen validation and test environment. This result indicates our pre-training tasks are more suitable for our designed navigation model. We also obtain a lower NE showing that our agent navigates closer to the destination. For HAMT [10], we report their results with ResNet-152 as the vision encoder for a fair comparison. Table 3.3 shows the performance of LOViS compared various models on R4R benchmark. Same as R2R, we can better perform in all evaluation metrics. Compared to the our reproduced results of the RecBERT [36], we can improve 4% of CLS, 1% of nDTW, and 2% of sDTW in the unseen validation environment, which indicates the better fidelity of our model.

3.4.2 Ablation Study

EXOR. Table 3.4 shows the ablation study results. Row#1 is the baseline model. Row#2 (*Lan-Obj*) shows that explicitly modeling important landmarks and aligned objects improves the performance compared to the baseline. *Rel* (row#3) is the result after modeling the spatial relation tokens describing the relative relation between agent and landmark. *Rel_v* (row#4) is the result after modeling the spatial relations in motions. The improved SDTW shows the modeling of spatial relations can help the agent to follow the instructions. However, the spatial terms directly describing the landmark are more helpful than the spatial terms in motions.

Different Pre-training Tasks for LOViS. In Table 3.5, we show the influence of each pre-training task on both RecBERT and LOViS. For RecBERT baseline model, SSAP shows about 2% of improvement on both seen and unseen environments. Although the tasks of VM and OM


step1
Viewpoints:
Instructions:
Go straight.
Pass the piano and the pictures on the wall
Head down to the bedroom.
Stop by bed.
step2
Viewpoints:
Instructions:
Go straight.
Pass the piano and the pictures on the wall
Head down to the bedroom.
Stop by bed.
step3
Viewpoints:
Instructions:
Go straight.
Pass the piano and the pictures on the wall
Head down to the bedroom.
Stop by bed.
step4
Viewpoints:
Instructions:
Go straight.
Pass the piano and the pictures on the wall
Head down to the bedroom.
Stop by bed.
step5
Viewpoints:
Instructions:
Go straight.
Pass the piano and the pictures on the wall
Head down to the bedroom.
Stop by bed.
Figure 3.5 An example visualization of the navigation process in EXOR. The green boxes are spatial configurations; the darker green means higher weights; the yellow boxes are the selected landmarks; the orange arrows are the path.

independently do not change the performance of MLM+SSAP, the combination of two tasks improves the performance by about 1%. The same phenomenon happens in LOViS. SSAP improves the performance by a large margin. Although VM and OM do not show significant improvement when used separately in the unseen environment, they improve both SR and SPL in the seen environment. The combination of VM and OM improves the performance significantly, especially in the seen environment.

		Baseline Model				LOViS (Our Model)			
		Val Seen		Val Unseen		Val Seen		Val Unseen	
	Tasks	**SR↑**	**SPL↑**	**SR↑**	**SPL↑**	**SR↑**	**SPL↑**	**SR↑**	**SPL↑**
1	MLM	0.712	0.662	0.613	0.562	0.724	0.673	0.621	0.564
2	MLM+SSAP	0.731	0.675	0.619	0.575	0.747	0.695	0.649	0.585
3	MLM+SSAP+VM	0.737	0.683	0.622	0.577	0.755	0.711	0.637	0.581
4	MLM+SSAP+OM	0.730	0.672	0.617	0.574	0.766	0.724	0.629	0.579
5	MLM+SSAP+VM+OM	**0.743**	**0.691**	**0.632**	**0.583**	**0.774**	**0.722**	**0.653**	**0.592**

Table 3.5 Ablation study for different pre-training tasks for LOViS.

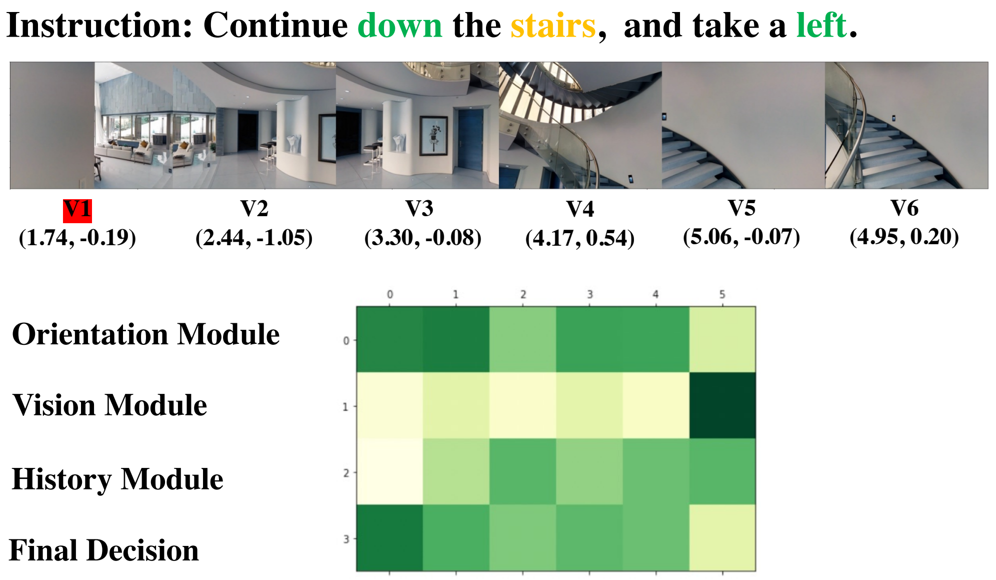


Figure 3.6 A qualitative example to show each module of LOViS.

3.4.3 Qualitative Examples

EXOR. Figure 3.5 illustrates an example of the navigation process, visualized using selected landmarks based on spatial configurations. The darker green spans in the instruction indicate the spatial configuration which the agentThe darker green spans in the instruction highlight the spatial configurations that the agent pays greater attention. Notably, the model's attention transitions from the beginning of the instruction to the end as navigation progress. The yellow boxes highlight the selected landmarks, demonstrating that as the agent navigates, its focus on landmarks dynamically shifts in response to the navigation progress.

LOViS. Figure 5.10 shows a qualitative example that demonstrates the performance of each module of LOViS navigation agent. The ground-truth viewpoint is v1. The word "*down*" and "*left*" are the orientation signals. The word "*stairs*" is the vision signal. The attention map shows the score of different candidate viewpoints in each module. The darker color means the higher score. The numbers below each viewpoint show the orientation information with the format of <relative heading, relative elevation>. The lower value of each number means the orientation is more towards left and down respectively. It is evident that the *orientation module* gives a higher score to the viewpoints that are left, and their elevation is down. The *vision module* gives a higher score to the viewpoints that "stairs" can be seen. The history module also gives a relatively higher score to the viewpoints on the right side. The final decision is $v1$ with its weights of $[0.02, -0.03, -0.04]$ to the three modules. The example shows that our designed orientation and vision modules can attend to the viewpoint with the corresponding information.

3.5 Conclusion

We propose two neural agents that integrate the semantic elements of motions and landmarks for navigation. For EXOR, we first identify key landmarks based on spatial configurations and then guide the agent to focus on relevant objects in the visual environment. Additionally, we explicitly model spatial relations between the agent and the landmarks from the agent's perspective. For LOViS, we introduce vision and orientation modules in the agent's neural architecture. These modules effectively ground landmark mentions and spatial information related to the agent's orientation, as expressed in natural language instructions, into the visual environment. To further enhance their effectiveness, we design new pre-training tasks that equip the agent with spatial reasoning and visual perception abilities before navigation. We evaluate our models on the R2R and R4R datasets, achieving SOTA results. Our findings demonstrate that modeling explicit spatial semantics not only improves navigation accuracy but also enhances the interpretability of navigation agents.

CHAPTER 4

ADDRESSING AMBIGUOUS INSTRUCTIONS AND IMPROVING EXPLAINABILITY

4.1 Introduction

Although grounding methods that connect textual and visual modalities—by aligning semantic information help improve navigation performance [36, 86, 1, 136, 137], we observe that two types of instructions make the grounding in the VLN task quite challenging. **First**, the instruction contains landmarks that are not recognizable by the navigation agent. For example, Figure 4.1(a), the agent can only see the "sofa", "table" and "chair" in the target viewpoint, based on the learned vision representations [35, 95, 20]. However, the instructor mentions landmarks of the "living room" and "kitchen" in the instruction, based on their prior knowledge about the environment, such as relating "sofa" to "living room". Given the small size of the dataset designed for learning navigation, it is hard to expect the agent to gain the same prior knowledge as the instructor. **Second**, the instructions contain the landmarks that can be applied to multiple targets, which causes ambiguity for the navigating agent. In Figure 4.1(b), the instruction "enter the door" does not help distinguish the target viewpoint from other candidate viewpoints since there are multiple "doors" and "walls" in the visual environment. As a result, we hypothesize that these types of instructions cause the explicit and fine-grained grounding to be less effective for the VLN task, as appears in [37, 134] that use sub-instructions and in [36, 45, 86, 136] that use object-level representations.

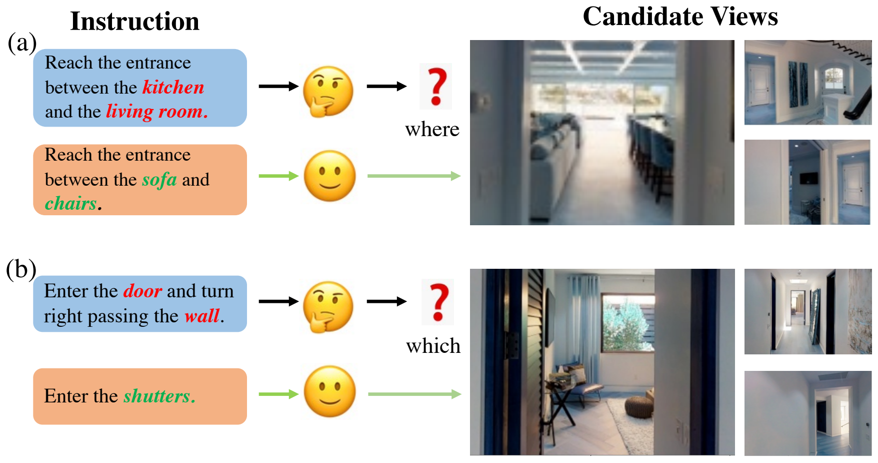

Instruction
Candidate Views
(a)
Reach the entrance between the kitchen and the living room.
where
Reach the entrance between the sofa and chairs.
(b)
Enter the door and turn right passing the wall.
which
Enter the shutters.

Figure 4.1 Instructions that make the grounding in the VLN task challenging.

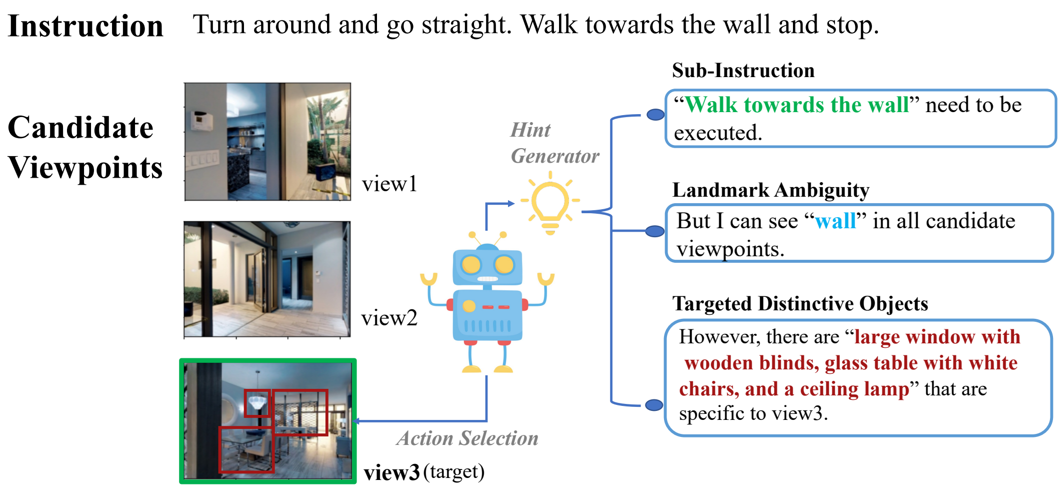

Instruction
Turn around and go straight. Walk towards the wall and stop.
Candidate Viewpoints
view1
view2
view3 (target)
Hint Generator
Action Selection
Sub-Instruction
"Walk towards the wall" need to be executed.
Landmark Ambiguity
But I can see "wall" in all candidate viewpoints.
Targeted Distinctive Objects
However, there are "large window with wooden blinds, glass table with white chairs, and a ceiling lamp" that are specific to view3.

Figure 4.2 VLN Agent with our hint generator.

To address the aforementioned issues, in this chapter, we first introduce a translator module in the VLN agent, named VLN-trans [138], which takes the given instruction and visual environment as inputs and then converts them to easy-to-follow sub-instructions focusing on two aspects: 1) *recognizable* landmarks based on the navigation agent's visualization ability. 2) *distinctive* landmarks that help the navigation agent distinguish the targeted viewpoint from the candidate viewpoints. Consequently, by focusing on those two aspects, the translator can enhance the connections between the given instructions and the agent's observed visual environment and improve the agent's navigation performance.

Furthermore, we introduce a hint generator for the VLN agent (NavHint) [135], aiming to generate visual descriptions that serve as indirect supervision to help the navigation agent obtain a better understanding of the visual environment (as depicted in Fig. 4.2). When the agent navigates at each step, the hint generator concurrently produces visual descriptions that are consistent with the agent's action decision. The hints are designed based on the rationale underlying the navigation process, including three aspects: *Sub-instruction*, *Landmark Ambiguity* and *Targeted Distinctive Objects*. Specifically, at each navigation step, **first**, the hint generator encourages the agent to report its navigation progress by specifying which part of the sub-instruction it is executing based on the current visual environment. As depicted in Fig. 4.2, the sub-instruction "walk towards the wall" needs to be executed. **Second**, the hint generator directs the agent to have a global view of the entire environment and recognize the landmarks mentioned in the instruction from all candidate viewpoints. The agent is tasked with identifying potential challenges by assessing the visibility of the landmarks and comparing the landmarks shared among viewpoints. For instance, in the given example, the landmark "wall" is ambiguous as it appears in multiple views. **Third**, in scenarios where challenges exist, the hint generator guides the agent in describing the distinctive visual objects that only appear in the targeted viewpoint, such as "*large window with wooden blinds*" in view3 in Fig 4.2. This aids the agent in deeply looking into the details of its selected viewpoint while globally comparing it to other candidates.

In summary, our **contributions** are as follows:

1. We propose a translator module that helps the navigation agent generate easy-to-follow sub-instructions considering recognizable and distinctive landmarks based on the agent's visual ability. We construct a high-quality synthetic sub-instruction dataset and design specific tasks for training the translator and the navigation agent.

2. We leverage a language model conditioned on the VLN models to design a hint generator that can be plugged into any VLN agent. This hint generator helps the agent develop a comprehensive understanding of the visual environment. We construct a synthetic hint dataset to provide the agent with visual descriptions at each navigation step. The dataset serves as an indirect supervision for jointly training the navigation agent and the hint generator.

3. We evaluate our method on R2R and R4R datasets, and our method achieves the SOTA results on all benchmarks. We also provide a detailed analysis of the agent's grounding ability by analyzing the translator and the quality of the generated hints, thereby improving the interpretability of the agent's decisions.

4.2 VLN-Trans: VLN Agent with a Translator

Fig. 4.3 (a) provides an overall picture of our proposed architecture for the navigation agent. We use VLN↻BERT [40] (in Sec. 2.3.3) as the backbone of our navigation agent and equip it with a novel *translator* module that is trained to convert the full instruction representation into the most relevant sub-instruction representation based on the current visual environment. Another key

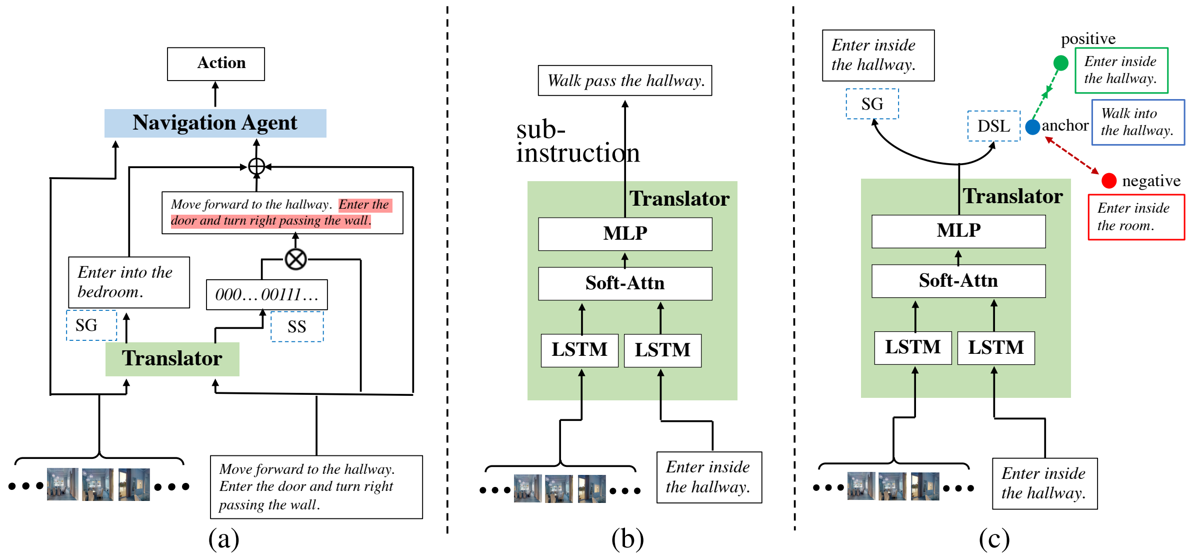

Action
Navigation Agent
Move forward to the hallway. Enter the door and turn right passing the wall.
Enter into the bedroom.
SG
000…00111…
SS
Translator
Move forward to the hallway. Enter the door and turn right passing the wall.
(a)
Walk pass the hallway.
sub-instruction
Translator
MLP
Soft-Attn
LSTM
LSTM
Enter inside the hallway.
(b)
Enter inside the hallway.
SG
DSL
positive
Enter inside the hallway.
anchor
Walk into the hallway.
negative
Enter inside the room.
Translator
MLP
Soft-Attn
LSTM
LSTM
Enter inside the hallway.
(c)

Figure 4.3 The overview of the proposed VLN-Trans. (a) Navigation agent with VLN-Trans. (b) The translator architecture (c) Pre-training the translator. SG:Sub-instruction Generation; DSL: Distinctive Sub-instruction Learning; SS: Sub-instruction Split.

point of our method is to create a synthetic sub-instruction dataset and design the pre-training tasks to encourage the translator to generate effective sub-instruction representations. We describe the details of our method in the following sections.

4.2.1 Synthetic Sub-instruction Dataset (SyFiS)

This section introduces our novel approach to automatically generate a synthetic fine-grained sub-instruction dataset, SyFiS, which is used to pre-train the *translator* (described in Sec. 4.2.2) in a contrastive manner. To this aim, for each viewpoint, we generate one positive sub-instruction and three negative sub-instructions. The viewpoints are taken from the R2R dataset [4], and the sub-instructions are generated based on our designed template. Fig. 4.5 shows an example describing our methodology for constructing the dataset.

The sub-instruction template includes two components: a motion indicator and a landmark. For example, in the sub-instruction “turn left to the kitchen”, the motion indicator is “turn left”, and the landmark is “kitchen”. The sub-instruction template is designed based on the semantics of *Spatial Configurations* explained in [17].

Motion Indicator Selection. First, we generate the motion indicator for the synthesized sub-instructions. Following [134], we use pos-tagging information to extract the verbs from

Motions	Vocab
FORWARD	go forward to; go forward past; pass; walk pass, walk forward, etc.
DOWN	go down; walk straight down; walk down; move forward down, etc.
UP	go up; walk up; climb; leading upwards; travel up, etc.
RIGHT	turn right to; make a right turn to, go right to, veer right to, etc.
LEFT	walk left to, turn left to, go left to, make a left turn to, make a left to, etc.
STOP	stay at, stand by, stop by, wait by, stop at, wait on, etc.

Figure 4.4 Motion indicator vocabulary.

instructions in the R2R training dataset and form our motion-indicators dictionary. We divide the motion indicators to 6 categories of: "FORWARD", "LEFT", "RIGHT", "UP", "DOWN", and "STOP". Each category has a set of corresponding verb phrases. We refer the Figure 4.4 for more details about motion indicator dictionary. Given a viewpoint, to select a motion indicator for each sub-instruction, we calculate the differences between the elevation and headings of the current and the target viewpoints. Based on the orientation difference and a threshold, *e.g.* 30 degrees, we decide the motion-indicator category. Then we randomly pick a motion verb from the corresponding category to be used in both generated positive and negative sub-instructions.

Landmark Selection. For generating the landmarks for the sub-instructions, we use the candidate viewpoints at each navigation step and select the most *recognizable* and *distinctive* landmarks that are easy for the navigation agent to follow. In our approach, the most recognizable landmarks are the objects that can be detected by CLIP. Using CLIP [89], given a viewpoint image, we predict a label token with the prompt "a photo of label" from an object label vocabulary. The probability that the image with representation b contains a label c is calculated as follows,

$$p(c) = \frac{exp(sim(b, w_c)/\tau_1)}{\sum_{i=1}^{M}(exp(sim(b, w_i))/\tau_1)}, \tag{4.1}$$

where τ_1 is the temperature parameter, sim is the cosine similarity between image representation and phrase representation w_c which are generated by CLIP [89], M is the vocabulary size. The top-k objects that have the maximum similarity with the image are selected to form the set of recognizable landmarks for each viewpoint. We filter out the distinctive landmarks from the recognizable landmarks. The distinctive landmarks are the ones that appear in the target viewpoint and not in

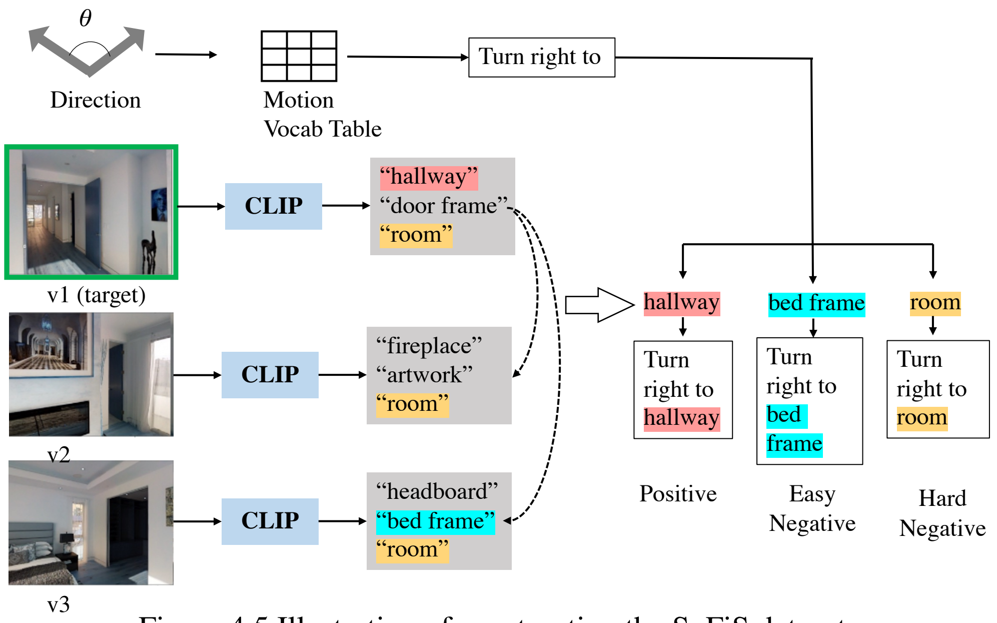


Figure 4.5 Illustration of constructing the SyFiS dataset.

any other candidate viewpoints. For instance, in the example of Fig. 4.5, "hallway" is a distinctive landmark because it only appears in the v1 (target viewpoint).

Forming Sub-instructions. We use the motion verbs and landmarks to construct sub-instructions based on our template. To form contrastive learning examples, we create positive and negative sub-instructions for each viewpoint. A positive sub-instruction is a sub-instruction that includes a distinctive landmark. The negative sub-instructions include easy negatives and hard negatives. An easy negative sub-instruction contains irrelevant landmarks that appear in any candidate viewpoint except the target viewpoint, *e.g.,* in Fig. 4.5, "bed frame" appears in v3 and is not observed in the target viewpoint. A hard negative sub-instruction includes the nondistinctive landmarks that appear in both the target viewpoint and other candidate viewpoints. For example, in Fig. 4.5, "room" can be observed in all candidate viewpoints; therefore, it is difficult to distinguish the target from other candidate viewpoints based on this landmark.

Statistic of the SyFiS dataset. We construct SyFiS dataset using $1,076,818$ trajectories, where 7198 trajectories are from the R2R dataset, and $1,069,620$ trajectories are from the augmented data [34]. Then we pair those trajectories with our synthetic instructions to construct the SyFiS dataset based on our pre-defined motion verb vocabulary and CLIP-generated landmarks (in Sec4.2.1). When we pre-train the translator, we use the sub-instruction of each viewpoint in a trajectory. There are usually 5 to 7 viewpoints in a trajectory; each viewpoint is with one positive sub-instruction and three negative sub-instructions.

4.2.2 Translator Architecture

The translator takes a set of candidate viewpoints and the corresponding sub-instruction as the inputs and generates new sub-instructions. The architecture of our translator is shown in Fig. 4.3(b). This architecture is similar to the LSTM-based Speaker in the previous works [106, 25]. However, they generate full instructions from the whole trajectories and use them as offline augmented data for training the navigation agent, while our translator adaptively generates sub-instruction during the agent's navigation process based on its observations at each step.

Formally, we feed text representations of sub-instruction X and the visual representations of candidate viewpoints V into the corresponding LSTM to obtain deeper representation $\tilde{X}$ and $\tilde{V}$. Then, we apply the soft attention between them to obtain the visually attended text representation $\tilde{X}'$, as:

$$\tilde{X}' = SoftAttn(\tilde{X}; \tilde{V}; \tilde{V}) = softmax(\tilde{X}^T W \tilde{V})\tilde{V}, \tag{4.2}$$

where W is the learned weights. Lastly, we use an MLP layer to generate sub-instruction X' from the hidden representation $\tilde{X}'$, as follows,

$$X' = softmax(MLP(\tilde{X}')) \tag{4.3}$$

We use the SyFiS dataset to pre-train this translator. We also design two pre-training tasks: Sub-instruction Generation and Distinctive sub-instruction Learning.

Sub-instruction Generation (SG). We first train the translator to generate a sub-instruction, given the positive instructions paired with the viewpoints in the SyfiS dataset as the ground-truth. We apply a cross-entropy loss between the generated sub-instruction X' and the positive sub-instruction X_p. The loss function for the SG task is as follows,

$$L_{SG} = -\frac{1}{L}\sum_{L} X_p log P(X') \tag{4.4}$$

Distinctive Sub-instruction Learning (DSL). To encourage the translator to learn sub-instruction representations that are close to the positive sub-instructions with recognizable and distinctive landmarks, and are far from the negative sub-instructions with irrelevant and nondistinctive landmarks, we use triplet loss to train the translator in a contrastive way. To this aim, we first design triplets of sub-instructions in the form of <anchor, positive, negative>. For each viewpoint, we select one positive and three negative sub-instructions forming three triplets per viewpoint. We obtain the anchor sub-instruction by replacing the motion indicator in the positive sub-instruction with a different motion verb in the same motion indicator category. We denote the text representation of anchor sub-instruction as X_a, positive sub-instruction as X_p, and negative sub-instruction as X_n. Then we feed them to the translator to obtain the corresponding hidden representations $\tilde{X}'_a$, $\tilde{X}'_p$, and $\tilde{X}'_n$ using Eq. 4.2. The triplet loss function for the DSL task is computed as follows,

$$L_{DSL} = max(D(\tilde{X}'_a, \tilde{X}'_p) - D(X', \tilde{X}'_n) + m, 0), \tag{4.5}$$

where m is a margin value to keep negative samples far apart, D is the pair-wise distance between representations. In summary, the total objective to pre-train the translator is:

$$L_{pre-train} = \alpha_1 L_{SG} + \alpha_2 L_{DSL} \tag{4.6}$$

where α_1 and α_2 are hyper-parameters for balancing the importance of the two losses.

4.2.3 VLN Agent with Translator

We place the pre-trained translator module on top of the backbone navigation agent to perform the navigation task. Fig.4.3(a) shows the architecture of our navigation agent. At each navigation step, the translator takes the given instruction and the current candidate viewpoints as input and generates new sub-instruction representations, which are then used as additional input to the navigation agent. Since the given instructions describe the full trajectory, we enable the translator module to focus on the part of the instruction that is in effect at each step. To this aim, we design another MLP layer in the translator to map the hidden states to a scalar attention representation. Then we do the element-wise multiplication between the attention representation and the instruction representation to obtain the attended instruction representation.

In summary, we first input the text representation of given instruction X and visual representation of candidate viewpoints V to the translator to obtain the translated sub-instruction representation

$\tilde{X}'$ using Eq. 4.2. Then we input $\tilde{X}'$ to another MLP layer to obtain the attention representation X'_m, $X'_m = MLP(\tilde{X}')$. Then we obtain the attended sub-instruction representation as $X'' = X'_m \odot X$, where $\odot$ is the element-wise multiplication. Lastly, we input text representation X along with translated sub-instruction representation $\tilde{X}'$ and the attended instruction representation X'' into the navigation agent. In such a case, we update the text representation X of VLN↻BERT as $[X; \tilde{X}'; X'']$, where ; is the concatenation operation.

4.3 NavHint: VLN Agent with a Hint Generator

The hint generator is designed as a Transformer-based decoder that leverages visual output from the navigation agent to produce corresponding hints. This hint generator can be plugged into any VLN agent as a language model conditioned on the VLN models. To train the hint generator, we propose a synthetic navigation hint dataset based on Room2Room (R2R) [4] dataset. Our dataset provides hints for each step of the trajectory in the R2R dataset. Each hint description includes sub-instruction, landmark ambiguity, and targeted distinctive objects introduced above. The dataset serves as an extra supervision to train the navigation agent and the hint generator jointly. Besides, our constructed dataset can be utilized to explicitly analyze the navigation agent's grounding ability by assessing the quality of generated hints. In the following section, we first present our constructed navigation hint dataset. Then, we introduce the hint generator. The navigation hint dataset is used to train the navigation agent and the hint generator jointly.

4.3.1 Navigation Hint Dataset

The purpose of constructing the navigation hint dataset is to provide supervision for the hint generator to generate detailed visual description. The navigation hint dataset is automatically generated based on instruction and trajectory pairs from the R2R dataset [4]. For every step of the trajectory, we provide hints that mainly include three key elements, as described below.

Sub-instruction is the first part of the hint that pinpoints to the relevant part of the instruction (sub-instruction) to be processed at the current step. We obtain the sub-instructions and their corresponding viewpoints from the FGR2R [37] dataset, which provides human annotations of sub-instructions and the aligned viewpoints. After obtaining the sub-instruction at each step, we

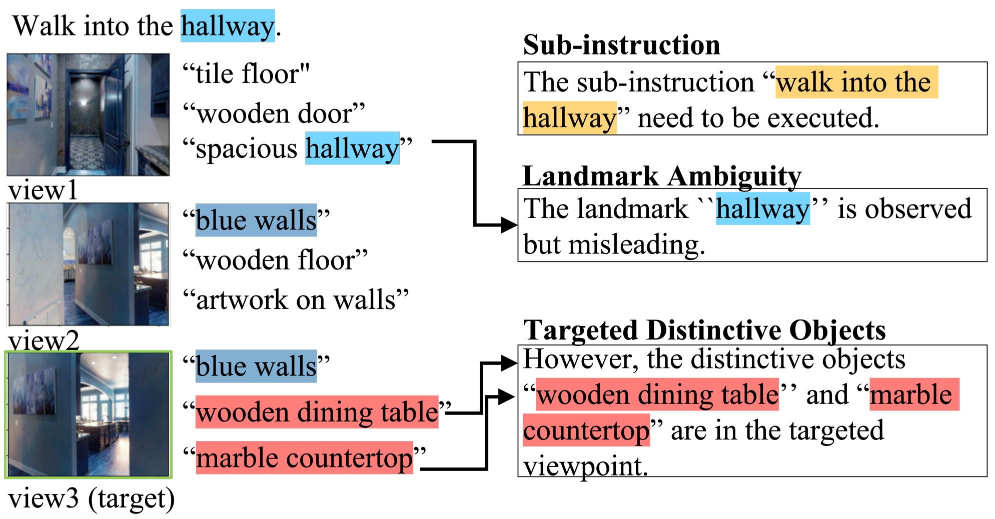

Walk into the hallway.
"tile floor"
"wooden door"
"spacious hallway"
view1
"blue walls"
"wooden floor"
"artwork on walls"
view2
"blue walls"
"wooden dining table"
"marble countertop"
view3 (target)
Sub-instruction
The sub-instruction "walk into the hallway" need to be executed.
Landmark Ambiguity
The landmark ``hallway'' is observed but misleading.
Targeted Distinctive Objects
However, the distinctive objects "wooden dining table'' and "marble countertop" are in the targeted viewpoint.

Figure 4.6 Navigation hint dataset. An example of a navigation hints with the landmark ambiguity of "*Missing Landmarks*". The sub-instruction is"walk into the hallway", and the landmark "hallway" in the instruction is observed in the view1 rather than target view3, which can potentially mislead the navigation agent. The target distinctive objects "wooden dining table" and "marble countertop." are then provided. "Blue walls" is non-distinctive as it appears in both view2 and view3.

Ambiguity Category	**Description**	**Hints**
Target Landmarks	Landmarks only appear in the target.	The {landmarks} are observed.
Multiple Landmarks	Landmarks are visible in multiple viewpoints including the target viewpoint.	The {landmarks} are observed in multiple viewpoints.
Missing Landmarks	Landmarks are visible in other viewpoints except for the target viewpoint.	The{landmarks} are misleading.
Invisible Landmark	Landmarks are not visible in all viewpoints	The{landmarks} are not observed.
No Landmarks	No landmarks in sub-instruction. (*e.g.* "*make a right turn*", "*turn left*", and "*go straight*")	∅

Table 4.1 Landmark ambiguity. The col#1 and col#2 show the categories of landmark ambiguity and the corresponding descriptions. The col#3 shows the template for generating the hint for each category.

insert it into our hint template, which is "*The {sub-instruction} needs to be executed.*". Guiding the navigation agent to detect the related sub-instruction at each step is crucial since it effectively assists the agent in tracking its navigation progress.

Landmark Ambiguity is the second part of the hint that describes the commonalities across multiple views that can result in ambiguity during navigation. This part of hint is achieved by examining the shared landmarks mentioned in the instruction among the candidate viewpoints.

To automatically generate this part of the hint for building the dataset, we first use spaCy[1] to extract noun phrases from sub-instruction and use them as landmarks. Then, we extract visual objects in each candidate viewpoint using MiniGPT-4 [149] with a two-step textual prompting. We choose visual objects generated by MiniGPT-4 instead of Matterport3D object annotations because

[1]https://spacy.io/

Matterport3D objects are pretty limited, with only 40 object categories like "*doors*", "*walls*", and "*floors*". These generic objects are not sufficient for resolving landmark ambiguity. Moreover, the absence of attribute annotations in Matterport3D poses a challenge for landmark disambiguation, such as the differences between "*wooden table*" and "*glass table*". In contrast, MiniGPT-4 can generate such detailed attribute descriptions. Specifically, for each candidate viewpoint, we feed MiniGPT-4 with the viewpoint image, asking "*Describe the details of the image.*" and then "*List the objects in the image*". The generated text is in free form, and we post-process it to retrieve a list of extracted object descriptions. After obtaining textual landmark names and visual objects, we examine the shared landmarks among the candidate viewpoints. The presence of shared landmarks can pose ambiguity for the navigation agent. We categorize the ambiguity into: *Target Landmarks*, *Multiple Landmarks*, *Missing Landmarks*, *Invisible Landmarks* and *No Landmark*, and their descriptions are in Table 4.1. After identifying the category of landmark ambiguity, we construct this part of the hint using the corresponding templates in col #3 of Table 4.1. Identifying landmark ambiguity requires the navigation agent to ground the mentioned landmark names in the instruction to the visual objects in all candidate viewpoints. Guiding the navigation agent to identify such detailed ambiguities can help enhance its understanding of the connection between the instruction and the entire visual environment.

Targeted Distinctive Objects is the third part of the hint that describes the distinctive visual objects specific to the targeted view. The agent should be able to justify its decision by describing the distinction of the targeted view. We follow the approach of obtaining distinctive objects in the VLN-Trans [138] that compares the visual objects in the targeted and other candidate viewpoints. The distinctive objects are the ones that exclusively appear in the targeted viewpoint and do not appear in other views. The hint template for targeted distinctive objects is "*However, {the comma-separated list of distinctive object names} are in the targeted view.*". We use 3 distinctive objects at most. If the cases belong to the challenge of "*Target Landmark*", there is no need to provide extra distinctive objects since the landmark is already exclusive to the targeted viewpoint. Describing distinctive objects is important to obtain a global understanding of the visual environment by highlighting the

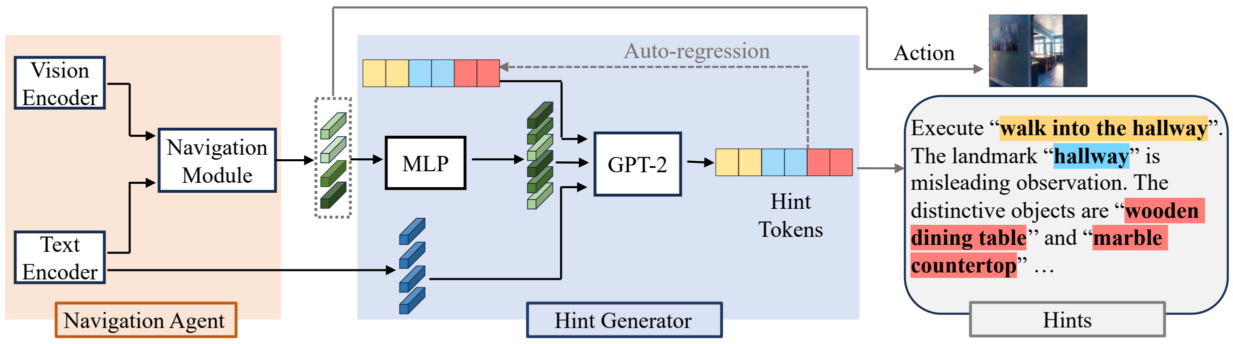

Vision Encoder
Text Encoder
Navigation Module
Navigation Agent
Auto-regression
MLP
GPT-2
Hint Tokens
Hint Generator
Action
Execute "walk into the hallway". The landmark "hallway" is misleading observation. The distinctive objects are "wooden dining table" and "marble countertop" …
Hints

Figure 4.7 Model architecture of NavHint.

differences between the targeted viewpoint and other candidate viewpoints.

4.3.2 VLN Agent with a Hint Generator

We propose a hint generator that can be plugged into any navigation agent easily. We use VLN↻BERT [39] as the base model to illustrate our method but noted that the hint generator is compatible with most of the current agents. Fig. 4.7 shows the model architecture.

Text Encoder. We use BERT [110] to obtain initial text representation of instruction, denoted as $X = [x_1, x_2, \cdots, x_l]$.

Vision Encoder. We follow previous works to concatenate image and relative orientation features as vision features for each candidate viewpoint. Specifically, we extract the image features from ResNet-152 [35] pre-trained on the Places365 dataset [144]. The orientation features are derived from the relative heading denoted as α and the elevation denoted as β. The orientation features are represented as $[\sin\alpha; \cos\alpha; \sin\beta; \cos\beta]$. The vision features are then passed through an MLP (Multilayer Perception) of Vision Encoder to obtain vision representation for each candidate viewpoint, denoted as $[v_1, v_2, \cdots, v_n]$.

Navigation Agent. VLN↻BERT is a cross-modal Transformer model. Besides text and vision representations, a state representation is introduced in the model to store history information recurrently, which is denoted as S. At the t-th navigation step, the text representation X, the visual representation V_t and state representation S_t are input into cross-modal Transformer layers, as follows,

$$\hat{X}, \hat{S}_t, \hat{V}_t = Cross\_Attn(X, [S_t; V_t]), \tag{4.7}$$

where $\hat{X}$, $\hat{S}_t$, and $\hat{V}_t$ are the learnt contextual text, state representation, and visual representations, respectively. Then we apply attention layer between state representation $\hat{S}_t$ contextual vision representations $\hat{V}_t$ as follows,

$$S_{t+1}, a_t = Attn(k = \hat{V}_t, q = \hat{S}_t, v = \hat{V}_t), \tag{4.8}$$

where S_{t+1} is the updated state representation that is passed to the next steps to show the history. a_t is the attention score over the navigable views to show action probability of the current step.

Hint Generator. Inspired by the idea of prefix engineering [77] that uses the image representation as the prefix of the text for the image captioning task, we employ a decoder language model (LM) and use the contextual visual representation of the navigation agent and the original instruction as the prefix. However, unlike the previous work, rather than just using one image as the prefix, we input all images of candidate viewpoints to encourage the hint generator to learn the global relations among views.

Formally, we denote the hint at the i-th navigation step as $C^i = \{c^i_1, c^i_2, \cdots, c^i_j\}$, where j is the length of the hint. Different from LANA [113] that generates route description after navigation, our hint generator provides a more in-depth visual description at each step. Our approach requires the agent to possess a global and deep visual understanding, which can be learnt through the supervision from our navigation hint dataset explained in Section 4.3.1. We obtain the LM representation of the original instruction W and the hint C as $X' = \{x'_1, x'_2, \cdots, x'_l\}$ and $c = \{c_1, c_2, \cdots, c_j\}$ respectively. Since the semantic structure of our auto-generated dataset can be easily captured, we use a 1.5B-parameters decoder LM (GPT-2 large) in the hint generator. Note that any larger decoder language model in the GPT series can be employed.

We use the instruction text representation X' as the instruction prefix representation. We use the weighted vision representations output from the navigation agent as the image prefix representation. The weighted vision representation is obtained using action probability and the contextual vision representations as $\hat{\hat{V}}_t = a_t * \hat{V}_t$. Then we simply employ an MLP to map $\hat{\hat{V}}_t$ to LM token space. We denote such MLP as F. We obtain prefix embedding that is mapped from visual representation $\hat{V}$ as

		Val seen			**Val Unseen**			**Test Unseen**		
	Method	**NE ↓**	**SR ↑**	**SPL↑**	**NE ↓**	**SR ↑**	**SPL↑**	**NE ↓**	**SR ↑**	**SPL ↑**
1	Env-Drop [106]	3.99	0.62	0.59	5.22	0.47	0.43	5.23	0.51	0.47
2	RelGraph [36]	3.47	0.67	0.65	4.73	0.57	0.53	4.75	0.55	0.52
3	NvEM [1]	3.44	0.69	0.65	4.27	0.60	0.55	4.37	0.58	0.54
4	PREVALENT [34]	3.67	0.69	0.65	4.71	0.58	0.53	5.30	0.54	0.51
5	HAMT (ResNet) [10]	–	0.69	0.65	–	0.64	0.58	–	–	–
6	HAMT (ViT) [10]	2.51	0.76	0.72	–	0.66	0.61	**3.93**	0.65	**0.60**
7	CITL [69]	2.65	0.75	0.70	3.87	0.63	0.58	3.94	0.64	0.59
8	ADAPT [70]	2.70	0.74	0.69	3.66	0.66	0.59	4.11	0.63	0.57
9	LOViS [137]	**2.40**	0.77	0.72	3.71	0.65	0.59	4.07	0.63	0.58
10	VLN↻BERT [40]	2.90	0.72	0.68	3.93	0.63	0.57	4.09	0.63	0.57
11	VLN↻BERT^{+}*(ours)*	2.72	0.75	0.70	3.65	0.65	0.60	4.09	0.63	0.57
12	VLN↻BERT^{++} *(ours)*	2.51	0.77	0.72	3.40	0.67	0.61	4.02	0.63	0.58
13	VLN-Trans-R2R *(ours)*	**2.40**	**0.78**	**0.73**	3.37	0.67	**0.63**	3.94	0.65	0.59
14	VLN-Trans-FG-R2R *(ours)*	2.45	0.77	0.72	**3.34**	**0.69**	**0.63**	3.94	**0.66**	**0.60**

Table 4.2 Experimental results on R2R Benchmarks in a single-run setting. The best results are in bold font. + means we add RXR [55] and Marky-mT5 dataset [112] as the extra data to pre-train the navigation agent. ++ means we further add the SyFiS dataset to pre-train the navigation agent. ViT means Vision Transformer representations.

follows,

$$p_1, \cdots, p_k = F(\hat{\hat{V}}_t), \tag{4.9}$$

where k is the prefix length, and p is the image prefix representation. We concatenate the representation of image prefix p and instruction prefix X', and combine them with the text representation of hint C. The hint generator only decodes the hint in an auto-regressive manner at each step. During training, the parameters of both of MLP and the LM in the hint generator and the navigator are updated. The training objective is to maximize the likelihood of the next hint token. The following equation shows the loss of generating the j-th token of the hint at the i-th step.

$$\begin{aligned} L_{hint} = -\sum_{i,j} \log p_\theta(c^i_j | p^i_1, \cdots, p^i_k, \\ x'_1, \cdots, x'_l, c^i_j, \cdots, c^i_{j-1}). \end{aligned} \tag{4.10}$$

		Validation Unseen					Test Unseen		
	Method	NE ↓	SR ↑	SPL↑	sDTW↑	nDTW↑	NE ↓	SR ↑	SPL ↑
1	Seq-to-Seq [4]	7.81	0.22	–	–	–	7.85	0.20	0.18
2	Self-Monitor [73]	5.52	0.45	0.32	–	–	5.67	0.48	0.35
3	AuxRN [73]	5.63	0.51	0.46	–	–	–	–	–
4	VLN↻BERT [39]	3.93	0.63	0.57	–	–	4.09	0.63	0.57
5	HAMT (ViT) [10]	3.97	0.66	0.61	–	–	3.93	0.65	**0.60**
6	LANA [113]	–	0.66	0.60	–	–	–	0.64	0.59
7	VLN-SIG (ViT) [56]	3.37	0.68	0.62	0.59	0.70	–	0.65	**0.60**
8	VLN-trans [138]	3.34	**0.69**	0.63	0.60	0.70	3.94	**0.66**	**0.60**
9	EDrop* [106]	5.49	0.55	0.47	0.42	0.58	5.60	0.51	0.49
10	EDrop + Hint. (NavHint)	5.44	0.55	0.47	0.44	0.60	5.47	0.53	0.49
11	VLN↻BERT^{++} [138]	3.40	0.67	0.61	0.58	0.69	4.02	0.63	0.58
12	VLN↻BERT^{++} + Hint. (NavHint)	**3.23**	**0.69**	**0.65**	**0.61**	**0.72**	4.00	0.65	**0.60**

Table 4.3 Experimental results on R2R dataset. ViT: uses Vision Transformer representations. Hint.: uses our hint generator.

4.4 Experiments

4.4.1 Experimental Results

Table 4.2 shows the model performance of VLN-Trans on the R2R benchmarks. Row #4 to row#9 are Transformer-based navigation baseline with pre-trained cross-modality representations, and such representations greatly improve performance of LSTM-based VLN models (row #1 to row#3). It is impressive that our VLN-Trans model's performance (row #13 and row #14) on both validation seen and unseen performs 2%-3% better than HAMT [10] when it even uses more advanced ViT [20] visual representations compared with ResNet. Our performance on both SR and SPL are still 3%-4% better than the VLN agent using contrastive learning: CITL [69] (row #7) and ADAPT [70] (row #8). LOViS [137] (row #9) is another very recent SOTA improving the pre-training representations of the navigation agent, but we can significantly surpass their performance. Lastly, compared to the baseline (row #10), we first significantly improve the performance (row #11) by using extra augmented data, Room-across-Room dataset (RXR) [55] and the Marky-mT5 [112], in the pre-training of navigation agent. The performance continues to improve when we further include the SyFiS dataset in the pre-training, as shown in row #12, proving the effectiveness of our synthetic data. Row #13 and row #14 are the experimental results after incorporating our pre-trained translator into the navigation model. First, for a fair comparison with

Model	**Val Seen**		**Val Unseen**	
	`Bleu-1`	`Bleu-4`	`Bleu-1`	`Bleu-4`
EDrop + Hint. (ours)	0.74	0.62	0.72	0.60
VLN↻BERT^{++}+ Hint. (ours)	**0.76**	**0.64**	**0.74**	**0.62**

Table 4.4 `Bleu` score for the generated sub-instruction on the R2R dataset.

Dataset	**Method**	**Tasks**			**Val Seen**		**Val Unseen**	
		SG	DSL	SS	SR↑	SPL↑	SR↑	SPL↑
R2R	VLN↻BERT^{++}				0.767	0.722	0.672	0.611
	1	✔			0.764	0.721	0.673	0.623
	2	✔	✔		**0.780**	**0.728**	0.674	0.627
	3	✔	✔	✔	0.772	0.720	**0.690**	**0.633**

Table 4.5 Ablation study for training tasks for the translator.

other models, we follow the baseline [40] to train the navigation agent using the R2R [4] dataset and the augmented data from PREVALENT [34]. As shown in row #13, our translator helps the navigation agent obtain the best results on the seen environment and improves SPL by 2% on the unseen validation environment, proving that the generated sub-instruction representation enhances the model's generalizability. However, FG-R2R [37] provides human-annotated alignments between sub-instructions and viewpoints for the R2R dataset, and our SyFiS dataset also provides synthetic sub-instructions for each viewpoint. Then we conduct another experiment using FG-R2R and SyFiS datasets to train the navigation agent. Simultaneously, we optimize the translator using the alignment information with our designed SG and SS losses during the navigation process. As shown in row #13, we further improve the SR and SPL on the unseen validation environment. This result indicates our designed losses can better utilize the alignment information.

Table 4.3 shows the performance of NavHint on validation unseen and test of the R2R dataset. To verify the adaptability of our approach, we evaluate it using both LSTM-based and Transformer-based navigation agents. Since Transformer-based methods are pre-trained on large vision-language datasets and have a more complex model architecture, they achieve a higher performance than LSTM-based methods. For the LSTM-based model, we use EDrop [106] which uses CLIP [89] visual representations without augmented data during training. For the Transformer-based model, we use the VLN↻BERT^{++} (row#11) as the baseline.

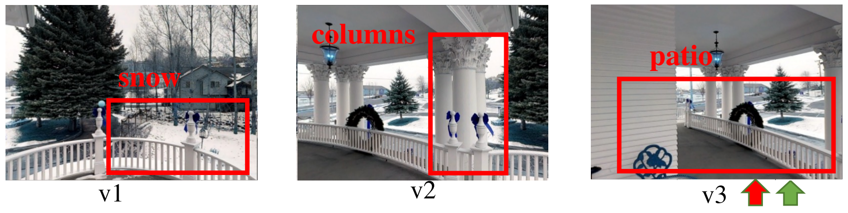
Method	Hints				Val Unseen		
	Sub.	L-A.	TD-Obj.	Obj.	SR↑	SPL↑	nDTW↑
VLN↻BERT^{++}					0.665	0.607	0.685
1	✔				0.671	0.612	0.690
2		✔			0.673	0.613	0.687
3			✔		0.677	0.624	0.702
4				✔	0.676	0.621	0.698
5	✔	✔			0.674	0.614	0.709
6	✔	✔		✔	0.681	0.632	0.694
7	✔	✔	✔		**0.692**	**0.647**	**0.724**

Table 4.6 Ablation study for different parts of hint. Sub.:sub-instruction; L-A.:Landmark Ambiguity; TD-Obj: Target Distinctive Objects. Obj:Top-3 objects.

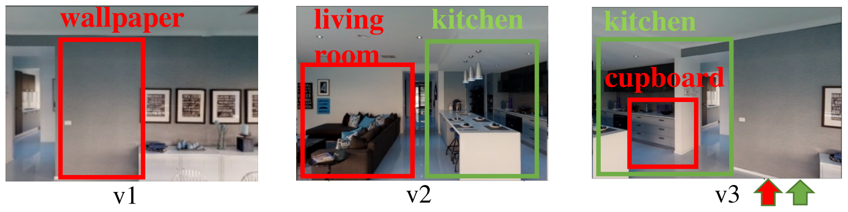
Row#1 to row#3 in Table 4.3 show other LSTM-based methods and row#4 to row#8 are the SOTA Transformer-based methods. Row#9 shows the performance of the LSTM baseline EDrop. Row#10 shows the results after equipping the EDrop with our designed hint generator. The improved sDTW and nDTW on the validation unseen proves that the hint generator helps the navigation agent follow the instructions. Moreover, our hint generator on top of the VLN↻BERT^{++} (row#12) significantly improves both wayfinding metrics (SP and SPL) and fidelity metrics (sDTW and nDTW) of the baseline model, indicating that our hint generator not only assists the agent in reaching the correct destination but also encourages the agent to follow the original instructions. Improving both LSTM-based and Transformer-based navigation agents shows the generalization ability of the navigation agent with our designed hint generator.

We use `Bleu` score [80] as an evaluation metric to assess whether the navigation agent can identify sub-instruction accurately. We conduct experiments on both LSTM-based and Transformer-based navigation agents, as shown in Table 4.4. The generated sub-instruction from the Transformer-based navigation agent can obtain a relatively high `Bleu` score compared to the LSTM-based agent. This result demonstrates that a more robust navigation agent achieves a stronger alignment between the instruction and visual modality for identifying the relevant part of the instruction to track the progress.

(a) Walk left past the table and chairs and through the doorway.



(b) Walk towards the kitchen area. Keep walking along the kitchen area towards the doorway.



(c) Walk through the kitchen and go into the hallway with a marble floor.

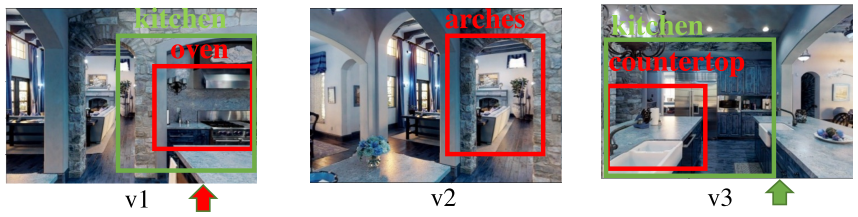


Figure 4.8 Qualitative examples to show how the translator helps the navigation agent. The red boxes and green boxes show the distinctive and the nondistinctive landmarks; the green arrow and red arrow show the target and the predicted viewpoints.

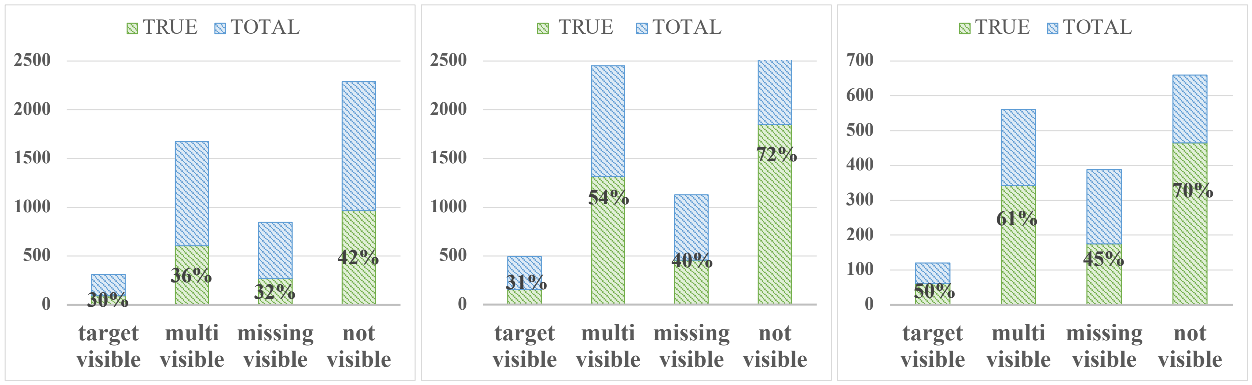
4.4.2 Ablation Study

VLN-Trans. In Table 4.5, we show the performance after ablating different tasks in the baseline model on the R2R and R2R-Last datasets. We compared with VLN↻BERT^{++}, which is our improved baseline after adding extra pre-training data to the navigation agent. First, we pre-train our translator with SG and DSL tasks and incorporate the translator into the navigation agent without further training. For both the R2R dataset and R2R-Last, SG and DSL pre-training tasks can incrementally improve the unseen performance (as shown in method 1 and method 2 for R2R and R2R-Last). Then we evaluate the effectiveness of the SS task when we use it to train the translator together with the navigation agent. For the R2R dataset, the model obtains the best result on the unseen environment after using the SS task. However, the SS task causes the performance drop for the R2R-Last dataset. This is because the R2R-Last dataset merely has the last single sub-instruction in each example and there is no other sub-instructions our model can identify and learn from.

NavHint. Table 5.4 reports the ablation analysis. From row#1 to row#3, we individually include sub-instruction, landmark ambiguity, and targeted distinctive objects to the hint. All navigation performance metrics improve gradually compared to the baseline. In another experiment (row#4), we attempt to describe the visual environment by identifying only top-3 recognized objects (using MiniGPT-4) in the targeted viewpoint without differing them from other viewpoints. The navigation results still improve, indicating that visual descriptions of the objects benefit the overall navigation performance. Row#5 shows that combining sub-instruction and landmark ambiguity further improves the baseline, particularly in the nDTW metric. In row#6, when we combine sub-instruction, landmark ambiguity and top-3 objects, we observe improvement in the goal-related metrics (SR and SPL), but the model's ability to faithfully follow the instruction is somewhat compromised (lower nDTW). The best result is obtained when we replace the above top-3 objects with distinctive ones (row#7), indicating our designed hint's effectiveness in describing the targeted view from a global perspective.

4.4.3 Translator Analysis

Our translator can relate the mentioned landmarks in the instructions to the visible and distinctive landmarks in the visual environment. In Fig. 4.8 (a), "tables" and "chairs" are not visible in three candidate viewpoints (v1-v3). However, our navigation agent can correctly recognize the target viewpoint using the implicit instruction representations generated by the translator. We assume the most recognizable and distinctive landmark, that is, the "patio" here in the viewpoint v3 has a higher chance to be connected to a "table" and a "chair" based on our pre-training, compared to the landmarks in the other viewpoints. In Fig. 4.8 (b), both candidate viewpoints v2 and v3 contain kitchen (green bounding boxes); hence it is hard to distinguish the target between them. However, for the translator, the most distinctive landmark in v3 is the "cupboard" which is more likely to be related to the "kitchen". Fig. 4.8(c) shows a failure case, in which the most distinctive landmark in candidate viewpoint v1 is "oven". It is more likely for the translator relates "oven" to the "kitchen" compared to "countertop", and the agent selects the wrong viewpoints. In fact, we



(a) EDrop+Hint. (b) VLN⟳BERT^{++}+Hint. (c) Correct Sub.

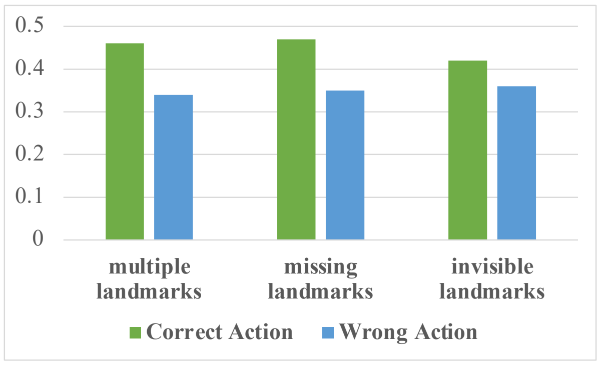
Figure 4.9 Accuracy of the landmark ambiguity in generated hints.

observe that the R2R validation unseen dataset has around 300 instructions containing “kitchen”. For corresponding viewpoints paired with such instructions, our SyFiS dataset generates 23 and 5 sub-instructions containing “oven” and “countertop”, respectively, indicating the trained translator more likely relates “oven” to “kitchen”.

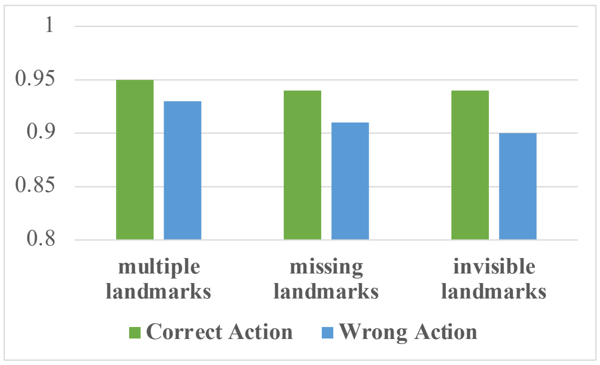
4.4.4 Generated Hints Analysis

Landmark Ambiguity Analysis. We assess the accuracy of four categories of landmark ambiguity in the generated hints. Specifically, We extract the part of the landmark ambiguity from the generated hint and check its accuracy in the visual environment. In Fig. 4.9, the TOTAL in the y-axis shows the total number of navigation steps that include each ambiguity category, shown on the x-axis. The TRUE (green) indicates the percentage of navigation steps when the corresponding ambiguity truly exists. We evaluate both LSTM-based and Transformer-based agents, and the result shows that Transformer-based agents can achieve higher accuracy of landmark ambiguity. We conclude that accurate landmark ambiguity detection is positively correlated with better navigation performance. In Fig. 4.9 (c), we evaluate the generated hint for the examples in which the sub-instruction is generated correctly, as indicated by a `Bleu-4` score of 1.0. In those examples, the accuracy of identifying each category of landmark ambiguity is also higher. This result shows accurately locating the sub-instruction positively impacts landmark ambiguity detection.

Targeted Distinctive Objects Analysis. We report the accuracy of identifying the targeted distinctive objects in the generated hints when landmark ambiguity exists, as shown in Fig. 4.10. The generated hints are from the model of VLN⟳BERT^{++} with our designed hint generator. We provide two



(a) Exact Matching



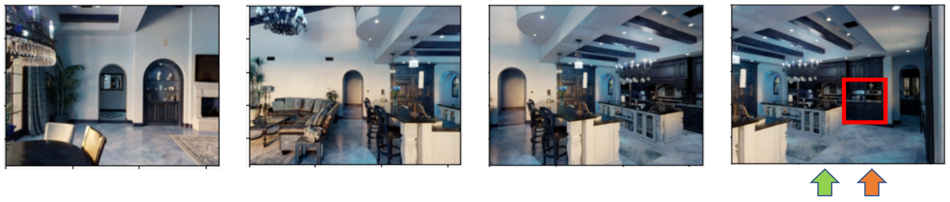
(b) Object Matching

Figure 4.10 Accuracy of the distinctive objects for each landmark ambiguity in the targeted viewpoint.

types of comparisons, exact phrase matching and object token matching while performing both wrong and right actions. Exact matching evaluates the detection of distinctive object tokens and the attribute descriptions in the whole referring phrase. Object matching only evaluates the detection of distinctive object tokens. The result shows that the accuracy in generating distinctive objects is generally higher when the action is correct than when it is wrong. Also, the agent tends to generate distinctive objects that align with its targeted viewpoint, as indicated by an accuracy exceeding 90%, even when the action is incorrect. The lower accuracy of exact matching also aligns with the fact that generating the whole referring expression, including the correct attributes, is more challenging.

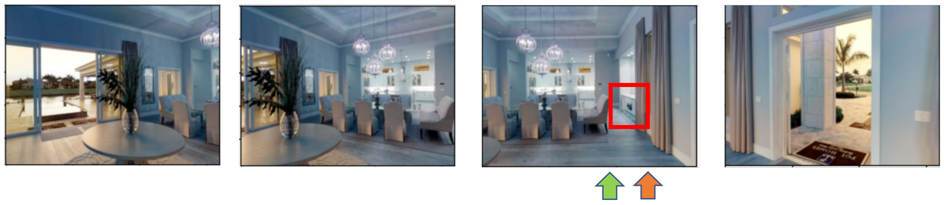
Generated Hints. Fig. 4.11 demonstrates a few examples of the generated descriptions. The first two examples show successful cases where the agent makes a correct decision. The first example shows the agent can accurately identify the sub-instruction and notice the ambiguous landmark "kitchen". Then, it correctly pinpoints the distinctive object "stove", which only appears in the target viewpoint. In fact, our *targeted distinctive object* design can help connect the specific object (*e.g.* stove, refrigerator, counter table) to more general scene objects (*e.g.* kitchen). Also, the second example shows the agent accurately points out the "table" in the instruction that appears in multiple viewpoints and refers to the "sideboard" in the target viewpoint. The third example shows a failure case in which the agent makes a wrong decision. The sub-instruction is correctly identified, but the agent should turn around towards the counter table and proceed to the sofa rather than walk to the sofa directly. This further indicates that our descriptor pushes the model to focus on

Instruction: Turn right and walk past the kitchen. Continue straight past the sink and turn left.



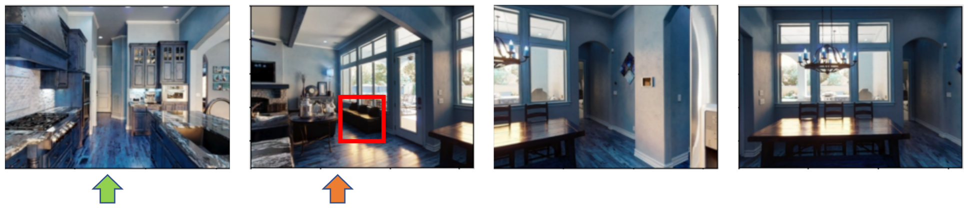
Hint: The instruction "turn right and walk past the kitchen" need to be executed. The landmark "kitchen" is observed in multiple views. The distinctive objects "stove" in the target viewpoint maybe helpful.

Instruction: With the couch behind you and the round table ahead and to the left, move forward towards the kitchen. Stop after you've passed the bar on your right.



Hint: The sub-instruction "with the couch behind you and the round table ahead and to the left, move forward towards the kitchen" need to be executed. The landmarks "table" is observed in multiple viewpoints. However, the distinctive object "sideboard" is in the targeted viewpoint."

Instruction: Turn around and walk towards the sofas. Turn left and walk past the first archway.



Hint: The instruction "turn around and walk towards the sofas." need to be executed. The landmark "sofa" is observed.

Figure 4.11 Qualitative examples for NavHint. The green and orange arrows show the ground truth and the predicted viewpoints, respectively.

landmarks directly and ignore the directions and motions in the instruction. Despite this, our model can generate a description consistent with its selection.

4.5 Conclusion

In the VLN task, instructions often include landmarks that are not recognizable to the agent or are not distinctive enough to specify the target based on the agent's vision perception. Our novel idea to solve these issues is to include a translator module in the navigation agent that converts the given instruction representations into effective sub-instruction representations at each navigation step. To train the translator, we construct a synthetic dataset and design pre-training tasks to encourage the translator to generate the sub-instruction with the most recognizable and distinctive landmarks. Our method achieves the SOTA results on navigation dataset. We also provide a comprehensive analysis to show the effectiveness of our translator. Furthermore, we enhance the navigation agent with a hint generator that provides explicit explanations for its actions, grounded in its visual perception. During navigation, the agent generates natural language descriptions about its visual environment at each step, including comparing various views and explaining ambiguities in recognizing the target destination. Empirical results show that visual description generation improves both navigation performance and the interpretability of actions taken by the navigation agent.

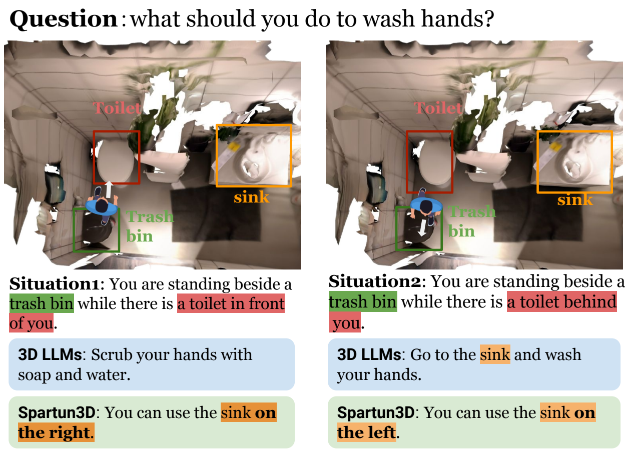
CHAPTER 5

ADVANCING VLN FOR REAL-WORLD CHALLENGES: NAVIGATION IN CONTINUOUS AND 3D ENVIRONMENTS

5.1 Introduction

The ultimate goal of VLN agent agent is to be deployed on a real robot for practical, everyday use in realistic environments. However, most existing experimental setups focus on navigation within discrete, graph-based environments, which are often far from real-world scenarios. In practical applications, a VLN agent must operate in a continuous, unstructured environment, where it relies on low-level control commands to navigate instead of predefined waypoints. Moreover, real-world navigation requires a robust understanding of 3D spatial relationships. Unlike simulated environments that often rely on 2D image-based navigation, a real-world VLN agent must perceive and reason about depth, obstacles, and spatial relationships to make action decisions. Bridging the gap between simulation and real-world challenges is crucial for advancing VLN research toward practical deployment. In this chapter, we address these challenges from two key perspectives: transitioning from discrete to continuous navigation and enhancing spatial reasoning in 3D environment.

For VLN-CE (continuous environment), recent research has made significant progress in improving navigation within continuous environments. Current VLN-CE navigation agents are typically equipped with a waypoint predictor, which is primarily trained to focus on high-level viewpoint selection while relying on an offline controller to execute low-level action execution within the environment. However, this approach overlooks including low-level actions as part of the training signal. Consequently, the navigation agent misses spatial information embedded in low-level actions, thereby affecting the grounding of different modalities of textual instructions, visual images, and physical spatial motions. Furthermore, existing waypoint predictors mainly use raw RGB and depth images, overlooking a thorough exploration of object semantic attributes, which are important for assessing the feasibility of the physical actions, such as recognizing that *walls are impassable*. To address the above-mentioned issues in the VLN-CE agents with a waypoint predictor, we first introduce a dual-action module in which the agent selects high-level viewpoints



Question: what should you do to wash hands?

Situation1: You are standing beside a trash bin while there is a toilet in front of you.

3D LLMs: Scrub your hands with soap and water.

Spartun3D: You can use the sink **on the right**.

Situation2: You are standing beside a trash bin while there is a toilet behind you.

3D LLMs: Go to the sink and wash your hands.

Spartun3D: You can use the sink **on the left**.

Figure 5.1 Illustration of situated scene understanding of Spartun3D-LLM compared to other 3D-based LLMs.

while generating low-level action sequences simultaneously [139]. The high-level actions serve as guidance, facilitating the agent's understanding of the relationships between low-level actions and navigable areas indicated by high-level actions. This enhances the agent's spatial grounding ability to connect action with visual perception and language understanding. Second, to address the issue of the waypoint predictor neglecting object semantic information, we incorporate visual representations with rich object semantics and explicit obstacle masking based on prior knowledge of object possibility.

For spatial understanding in 3D environment, existing studies mainly focus on integrating various 3D scene representations into LLMs, enabling the models to perform 3D grounding and spatial reasoning through natural language. For example, 3D-LLM [42] utilizes multi-view images to represent 3D scenes, pioneering a new direction in this field, while LEO [46] further pushes the boundary by directly injecting 3D point clouds into LLMs, aiming to develop a generalist embodied agent capable of 3D grounding, embodied reasoning, and action planning. Despite the promising progress, current 3D-based LLMs still fall short in ***situated understanding***, a fundamental capability for completing embodied tasks, such as Embodied Question Answering [18], Vision and Language Navigation [5], robotic manipulation [99], and many others. Situated understanding refers to the ability to interpret and reason about a 3D scene from a ***dynamic egocentric perspective***, where the agent must continuously adjust understanding based on its changing position and evolving environment around it. This capability is crucial because an agent's reasoning and response to the same question can vary depending on its current situation [83]. For example, as shown in Fig 5.1, given the same question *"What should you do to wash hands?"*, the agent might need to answer *"use the sink on the right"* or *"use the sink on the left"*, depending on the agent's current perspective and location relative to the *"sink"*. To address the aforementioned issues, we propose two key innovations: we first introduce a scalable, LLM-generated dataset named ***Spartun3D*** [141], consisting of approximately 133k examples. Different from datasets used by previous 3D-based LLMs [153, 46, 42], Spartun3D incorporates various situated spatial information conditioned on the agent's standing point and orientation within the environment, consisting of two situated tasks: situated captioning and situated QA. Situated captioning is our newly proposed task that requires generating descriptions of the surrounding objects and their spatial direction based on the agent's situation. Situated QA is designed with different types of questions targeting various levels of spatial reasoning ability for embodied agents. Furthermore, based on Spartun3D, we propose a new 3D-based LLM, ***Spartun3D-LLM***, which is built on the most recent state-of-the-art 3D-based LLM, LEO [46], but integrated with a novel *situated spatial alignment* module that explicitly aligns 3D visual objects, their attributes and spatial relationship to surrounding objects with corresponding textual descriptions, with the goal of better bridging the gap between the 3D and text spaces.

In summary, our **contributions** is summarized as follows.

1. We introduce a dual-action module for VLN-CE agents to ground high-level visual perception into low-level spatial actions. This design empowers the agent with the flexibility to select high-level viewpoints and generate low-level action sequences. We enhance the waypoint predictor with visual representations containing rich object semantics and explicit prior knowledge about objects' passability attributes. We adapt our method to several VLN-CE agents. The experimental results show the effectiveness of our approach in waypoint predictor, as well as high-level and low-level navigation performance.

2. We propose a method to address the limitation of situated understanding of the 3D-based LLMs from two perspectives. We construct an LLM-generated dataset based on our designed situated scene

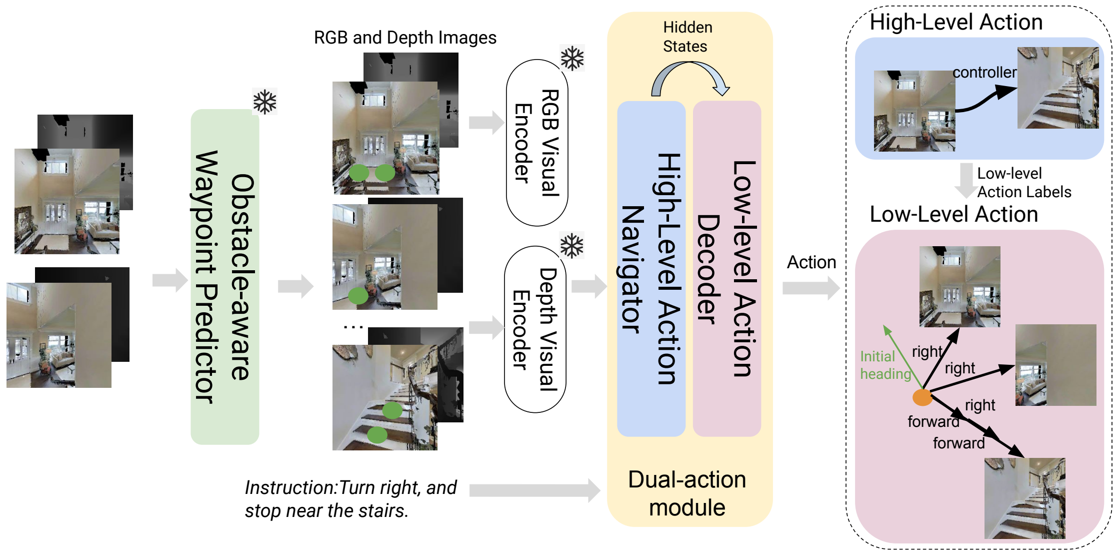


Figure 5.2 Main Architecture for the VLN-CE Agent. The waypoint predictor first provides navigable viewpoints (green circle). Then, the corresponding RGB, depth images, and textual instructions are input to our dual-action module. The freezing sign means no parameters are updated. Please refer to Fig. 5.3 for a detailed architecture of the low-level action decoder.

graph. Then, we propose an explicit situated spatial alignment on the 3D-LLM to encourage the model to learn alignment between 3D object and their textual representations directly. We provide comprehensive experiments to show our proposed Spartun3D improve situated understanding of SQA3D and navigation. We also provide analysis to show our proposed explicit alignment module helps spatial understanding.

5.2 Narrowing the Gap between Vision and Action for the VLN-CE Agent

We enhance the existing VLN-CE agent by introducing an obstacle-aware waypoint predictor and a dual-action module for navigation. The waypoint predictor enables the agent to generate waypoints in open areas rather than obstacles, while the dual-action module allows for flexible execution of both high-level and low-level navigation actions simultaneously. In the following sections, we first introduce the Transformer-based VLN-CE agent, which serves as our baseline. We then detail our improvements, beginning with the obstacle-aware waypoint predictor, followed by the dual-action module. Fig. 5.2 shows the main architecture.

5.2.1 VLN-CE Agent

The backbone comprises two primary components: the waypoint predictor and the navigator. The waypoint predictor is trained offline to generate navigable viewpoints, which are applied to the navigator for view selection.

Text Encoder. We use BERT [110] to obtain initial text representation of instruction w, denoted as $X = [x_1, x_2, \cdots, x_l]$.

Vision Encoder. In the baseline, different RGB vision encoders are used for the waypoint predictor and navigator. The waypoint predictor utilizes ResNet-152 [35] as its vision encoder, pre-trained on the ImageNet dataset [96], Meanwhile, the navigator uses the InternVideo [115] as the vision encoder, pre-trained on large video-text datasets. Formally, the obtained visual representations of RGB images are denoted as $v^{rgb} = \{v_1^{rgb}, v_2^{rgb}, \cdots, v_{12}^{rgb}\}$. The depth images are fed into DD-PPO ResNet-50 [120], which is trained for point-goal navigation, represented as $v^d = \{v_1^d, v_2^d, \cdots, v_{12}^d\}$.

Waypoint Predictor. We follow the waypoint predictor designed in [38], which is a multi-layer Transformer with a non-linear classifier. All 12 RGB image representations v^{rgb} and depth image representations v^d are concatenated and then input into the waypoint predictor to predict a heatmap of 120 angles-by-12 distances. Each angle is 3 degrees, and distances range from 0.25 to 3.00 meters with an interval of 0.25 meters. The heatmap is represented as a Gaussian distribution with a variance of 1.75m and 15° to expand the prediction range. The waypoint predictor is pre-trained based on the navigable connectivity graph from MP3D [7]. During inference, non-maximum suppression (NMS) is used to sample K neighboring waypoints, which are utilized as the candidate views for the following navigator.

Navigator. Our method is designed to be model-agnostic, allowing it to be applied to any VLN-CE navigation agent based on a waypoint predictor. In this paper, we utilize HAMT [10] as the navigator backbone. HAMT is a multimodal Transformer-based navigator that can memorize history information. Specifically, HAMT uses a sequence of panorama images as the navigation history during the navigation trajectory; then it applies Transformers to encode the observations on the trajectory to memorize temporal information. Formally, at each navigation step t, the waypoint predictor generates K candidate views, and their observation representations are denoted as O_t. Also, the history representation is denoted as H_t. HAMT concatenates history and observation as the vision modality and uses a cross-modal transformer to learn the connection between text presentation X and visual representation $[H_t; O_t]$. The actions are predicted by selecting the highest similarity score between observation encoding O_t and <CLS> token containing instruction-trajectory information.

5.2.2 Obstacle-Aware Waypoint Predictor

Object semantics in the visual environment is expected to play an important role in predicting navigable viewpoints, especially in the sense of open and obstacle areas. Objects have attributes that determine whether they should be labeled as passable or impassable. For example, the agent is not supposed to traverse beneath a *"table"* or on the *"bed"*. However, the current methods mainly leverage visual information from RGB and depth images and neglect further exploration of the object semantics and their attributes related to passibility.

To overcome this limitation, we first enhance the current waypoint predictor with vision representation from the VLPMs [89, 91, 115], which contain much more comprehensive object semantics than ResNet in the baseline's waypoint predictor. We employ vision representations from different VLPMs to assess their influence on the waypoint predictor's performance, and please see Table 5.5 for our detailed analysis. Second, we introduce an obstacle mask mechanism based on semantic segmentation within the visual environment and our prior knowledge about impassable objects. We utilize semantic segmentation provided by MP3D with approximately 40 object categories. To identify open areas, we define a vocabulary with open areas objects, such as *"floor"*, *"stairs"*, and *"door"*, and we mask semantic segments that are not within this vocabulary. Table 5.6 in the Appendix shows how different open-area vocabularies affect the performance of the waypoint predictor.

Formally, we denote the visual representation from VL pre-trained models of panoramic images as $v_c^{rgb} = \{v_{c1}^{rgb}, v_{c2}^{rgb}, \cdots, v_{c12}^{rgb}\}$. For each image, we obtain the corresponding obstacle mask based on semantic segmentation. We assign a label of 1 to object areas in the open vocabulary and 0 otherwise. The resulting obstacle masks are represented as $m = \{m_1, m_2, \cdots, m_{12}\}$. Subsequently, we apply obstacle masks to the RGB images and obtain the masked RGB representation, $v_{cm} = v_c * m$, which is then concatenated with depth visual representation v^d. This combined representation is input to the waypoint predictor to generate views at each navigation step. We train the waypoint predictor with enhanced visual representation and obstacle-masked image. Then, we employ it in the navigator for offline usage to generate navigable views.

5.2.3 Dual-Action Prediction for the Navigator

High-level Action is to select a view based on the similarity between the observation O_t and the hidden states from the cross-modal Transformer, which is represented as follows,

$$p_t^h = \texttt{Softmax}([H_t; O_t] * h_t^{\texttt{cls}}), \tag{5.1}$$

where $h^{\texttt{cls}}$ is the <CLS> token representation from navigator at step t, and p_t^h is the probability of high-level action. Once the most similar viewpoint is selected, the agent employs an offline controller to navigate to the corresponding position. While high-level actions effectively boost navigation performance, the training mechanism primarily focuses on view selection, neglecting the spatial information in the low-level action sequence. Additionally, it is challenging for a real-world robot to navigate to a precise angle and distance in a realistic environment. Real-world robots typically operate with very limited action sets, such as `FORWARD` 0.25m.

Low-level Action Challenges. The previous approach uses a non-linear classifier to predict an action class at each navigation step as follows,

$$p_t^l = \texttt{Softmax}(h_t^{\texttt{cls}} W_c), \tag{5.2}$$

where W_c projects the <CLS> token representation to four low-level actions, and p_t^l is the probability of low-level actions. While low-level actions are closer to real-world robotic behavior, directly modeling the agent to generate such actions results in a cost-training process. This is because the episodes for low-level actions are around 10 times longer than high-level actions (around 56 steps for low-level action steps, compared to $4 - 6$ for high-level episodes). Additionally, the performance drops substantially when training the agent with a low-level action classifier [38, 41].

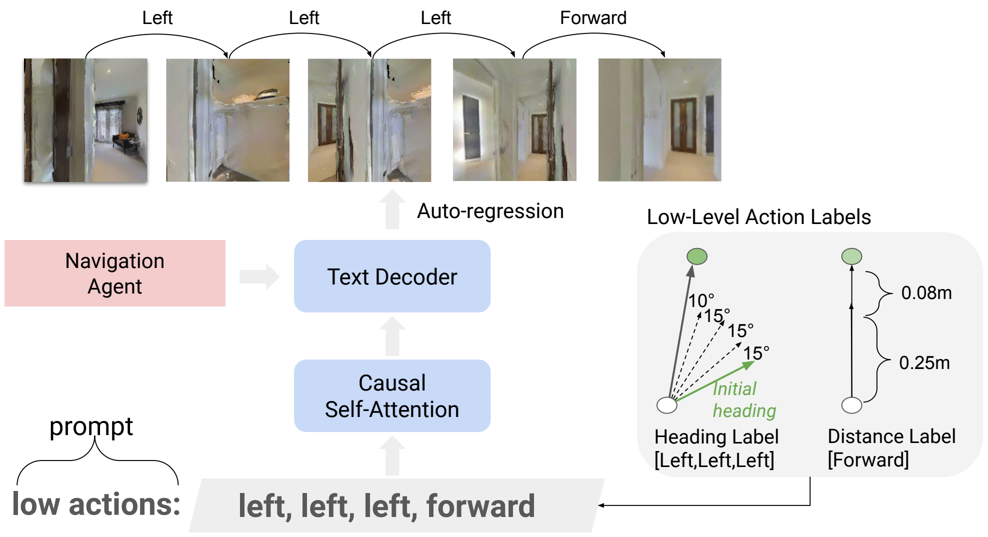


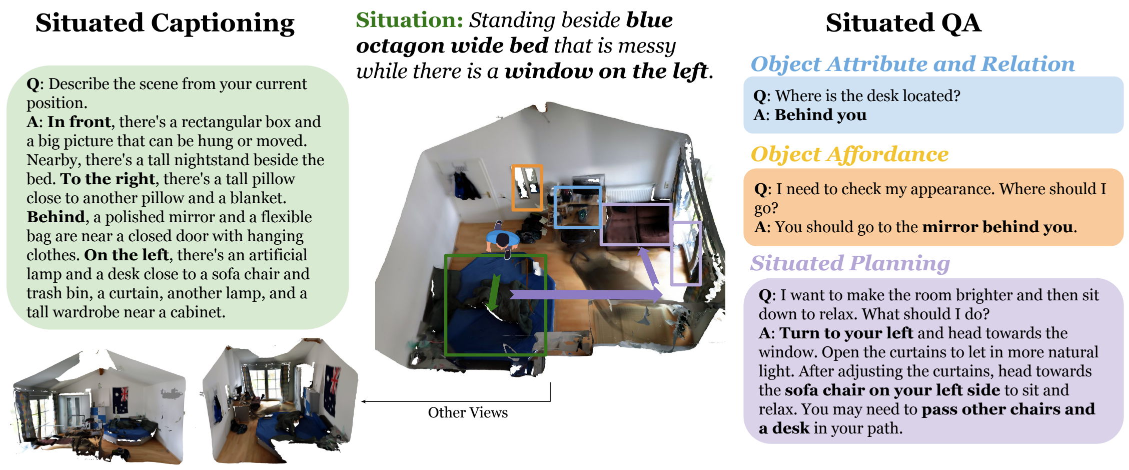
Figure 5.3 Low-Level Action Decoder.

Dual-Actions. We enable the VLN-CE agent to navigate simultaneously using high-level and low-level actions. Built upon the existing VLN-CE agent that predicts high-level actions, we introduce a decoder to simultaneously generate the corresponding low-level action sequence. Specifically, instead of generating one low-level action using an action classifier at each navigation step, our agent is trained to generate a low-level action sequence. We formulate low-level action prediction as the textual sequence generation task. As shown in Fig. 5.3, we introduce a Transformer-based text decoder to generate the low-level action sequences with the prompt of "*low actions:*". At each navigation step, the agent selects a high-level view, and at the same time, it is trained to generate the corresponding low-level sequence of action tokens auto-repressively. We input the text decoder with <CLS> representation from the navigator and textual prompt representation. The training objective is to maximize the likelihood of the next low-level action token. The following equation shows the loss of generating j-th token in the action sequence.

$$\mathcal{L}_{low} = -\sum_{j} \log p_{\theta}(a^{l}_{j}|a^{l}_{1}, \cdots, a^{i}_{m}), \tag{5.3}$$

where m is the length of the sequence.

We jointly train low-level action decoders and high-level selection. The labels for the low-level action sequence are obtained from the heading and distance differences between the initial view and the selected views in the high-level action. When the agent selects a high-level view at each



Situated Captioning
Situation: Standing beside blue octagon wide bed that is messy while there is a window on the left.
Situated QA
Q: Describe the scene from your current position.
A: In front, there's a rectangular box and a big picture that can be hung or moved. Nearby, there's a tall nightstand beside the bed. To the right, there's a tall pillow close to another pillow and a blanket. Behind, a polished mirror and a flexible bag are near a closed door with hanging clothes. On the left, there's an artificial lamp and a desk close to a sofa chair and trash bin, a curtain, another lamp, and a tall wardrobe near a cabinet.
Object Attribute and Relation
Q: Where is the desk located?
A: Behind you
Object Affordance
Q: I need to check my appearance. Where should I go?
A: You should go to the mirror behind you.
Situated Planning
Q: I want to make the room brighter and then sit down to relax. What should I do?
A: Turn to your left and head towards the window. Open the curtains to let in more natural light. After adjusting the curtains, head towards the sofa chair on your left side to sit and relax. You may need to pass other chairs and a desk in your path.
Other Views

Figure 5.4 Examples of two tasks in Spartun3D Benchmark. The green bounding box and arrow in the 3D scene demonstrate the standing point and orientation.

navigation step, the corresponding low-level sequence label is created. Specifically, we initially calculate the degree difference in headings and divide it by 15 to determine the number of rotation steps the agent takes. The direction (`LEFT` or `RIGHT`) is determined based on the smaller rotations. A similar process is applied to distance: we calculate the distance between the start point and the selected view and divide it by 0.25. We ignore the remaining if the heading or distance is not perfectly divisible. For instance, as shown in Fig. 5.3, the distance between the start point and the target view is 0.33m, and the low action is just one `FORWARD` (0.25m).

5.3 Spartun3D: Situated Spatial Understanding in 3D World

We first introduce a scalable, LLM-generated dataset named ***Spartun3D***, incorporates various situated spatial information. Furthermore, based on Spartun3D, we propose a new 3D-based LLM, ***Spartun3D-LLM***, which is built on the most recent state-of-the-art 3D-based LLM, LEO [46], but integrated with a novel *situated spatial alignment* module that explicitly aligns 3D visual objects, their attributes and spatial relationship to surrounding objects with corresponding textual descriptions, with the goal of better bridging the gap between the 3D and text spaces.

5.3.1 Spartun3D Dataset Construction

To better equip 3D-based LLMs with the capability of understanding situated 3D scenes, we introduce Spartun3D, a diverse and scalable situated 3D dataset. To ensure the scalability of Spartun3D, we carefully design an automatic pipeline that leverages the strong capabilities of

GPT-4o [78], with three key stages: (1) Designing diverse situations that specify the agent's standing point and orientation given a 3D scene as input (Sec. 5.3.1.1); (2) Constructing situated scene graphs to describe the spatial relationships between the agent and objects in the environment conditioned on the agent's situations (Sec. 5.3.1.2) ; and (3) Prompting LLMs to generate dataset based on situated scene graphs (Sec. 5.3.1.3).

5.3.1.1 Situation Design

The 3D scenes in Spartun3D are taken from 3RScan [121], which provides a diverse set of realistic 3D environments. Given a particular 3D scene with all the objects labeled by humans from 3RScan, our first step is to generate diverse situations for the agent. To construct the situation, we begin by identifying the standing point and orientation and then complete a situation description accordingly using the following template: "*You are standing beside {pivot object name}, and there is {referent object name} on the {left/right/front/backward}*." The elements within *{}* specify the key components that together define the situation. Below, we define the agent's standing point and orientation and explain how these elements are obtained to construct diverse and reliable situations.

Standing Point and Orientation. We begin with determining the agent's standing point and orientation within the 3D scene. Our approach is to place the agent beside an object, ensuring a clear reference for orientation when interacting with the environment. Specifically, we project all objects from 3D space onto a 2D plane, focusing only on the x and y coordinates to construct a bird-eye-view of the scene. From this projected 2D scene, we randomly select an object from the set of segmented objects within the 3D scene. To ensure the constructed situation remains realistic, we exclude objects that are positioned too high to avoid unnatural situations like *"standing beside the lamp on the ceiling"*. As a result, we limit the selection to objects whose z-axis is below the average height of all objects in the scene. Then, we choose a midpoint from two sides of the selected object's bounding box that are closest to the center of the scene, as shown in Fig. 5.5. By prioritizing the side closest to the center, we minimize this risk and keep the agent within the scene's boundaries. Finally, the selected midpoint will be used as the agent's standing point. In addition, we need to determine the agent's orientation. We assume the agent's orientation is always facing forward to the center of

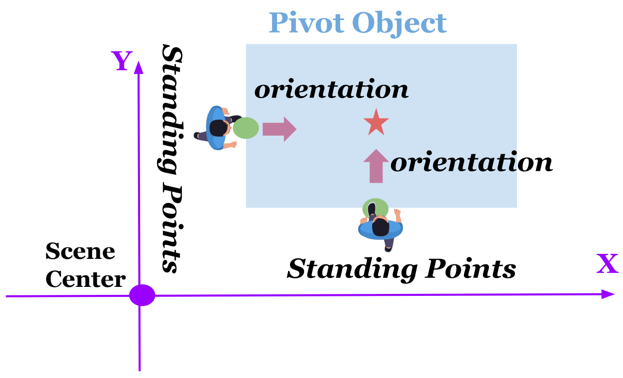


Figure 5.5 Standing Point and Orientation Selection.

the selected object. This guarantees that the selected object remains within the agent's field of view.

Pivot and Referent Object. Once the agent's standing point and orientation are determined, we refer to the object that the agent stands beside as the *"pivot object"*, and other objects surrounding the pivot object are potential referent objects. A referent object is then randomly selected, and its relative position (left/right/front/backward), with respect to the agent's standing point and orientation, is used to generate the description of the situation.

5.3.1.2 Situated Scene Graph Construction

Building on the agent's situation, we further construct a situated scene graph that captures the comprehensive spatial relationships between the agent and its surrounding objects. Existing 3D-based LLMs [153, 46] represent scenes in a structured manner using JSON-formatted scene graphs, including detailed scene context of object attributes and relative spatial relationships between objects. However, their spatial relations are based on a global view, such as a bird-view-eye perspective (as shown in Fig. 5.6). To enable situated understanding, we introduce a *situated-scene-graph* adapted from the original global scene graph to capture all relative spatial relationships between the agent's standing point and surrounding objects as follows:

Rotation Angles. We calculate rotation angles that reorient the agent from its orientation to the surrounding objects. Specifically, we first calculate the horizontal angle between the standing point and the center of the pivot object. Next, we calculate the horizontal angle between the standing point and a surrounding object. The rotation angle is determined by the difference between these two angles. We further normalize the rotation angles such that larger values correspond to a greater

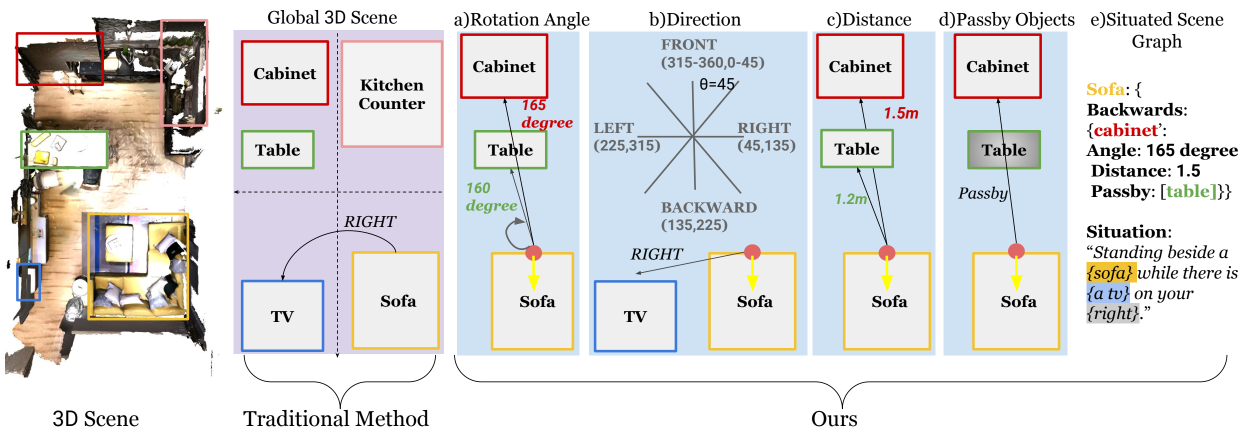


Figure 5.6 Spatial information in Situated Scene Graph. The red dot and green arrow show the standing point and orientation, respectively. In this example, the pivot object is the "sofa", the referent object is the "TV", and the surrounding objects include the "table" and "cabinet".

degree of rightward rotation.

Direction. We classify the object's rotation angles to the agent into four directional categories according to a predefined standard: [*front, right, backward, left*] (see Fig. 5.6 (b)). For instance, an object is categorized as "*right*" if the turn angle falls within the range of [45-135] degrees relative to the agent's forward-facing orientation.

Distance. We compute the Euclidean distance between the agent's standing point and the center of the bounding boxes of surrounding objects.

Passby Objects. We assess whether the agent can move freely from its standing point to other objects. We draw a straight line from the agent's standing point to the center of the referenced object. If this line intersects any other objects in the scene, those objects are considered "passby objects". For example, as illustrated in Fig 5.6 (d), the "*table*" is a passby object between the agent and the "*kitchen cabinet*". We explictly include the information of passby objects to help the agent build awareness of objects that might influence its path while navigating.

After gathering the spatial information described above, we organize it into a JSON format (shown in Fig. 5.6 (e)), which is then used as input to prompt LLMs to generate our datasets.

5.3.1.3 LLMs prompting

We design specific instructions to prompt **GPT-4o** [78] for two situated tasks: **Situated Captioning** and **Situated QA**. For both tasks, we ask GPT-4o to provide responses considering situated spatial information, and examples of the generated dataset are shown in Fig. 5.4.

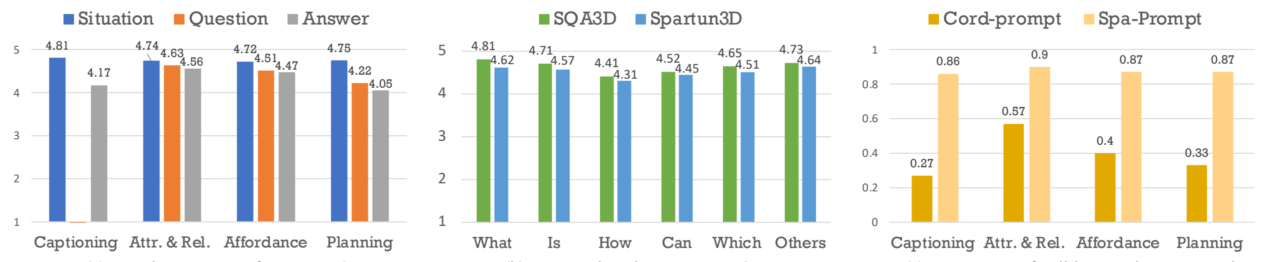
Situated Captioning is our newly introduced task, aiming to generate brief situated descriptions of the surrounding objects as the agent performs a 360° clockwise rotation starting from its standing point and orientation. The motivation for introducing this task stems from its crucial role in embodied tasks, such as navigation, where the agent must interpret and reason about its environment from 360° panoramic views to make decisions about movement and interaction [146]. Therefore, we guide GPT-4o to generate descriptions progressively, starting from lower rotation angles and moving toward higher angles in each direction.

Situated QA. We design three types of questions for the Situated QA task, each targeting a different aspect of spatial reasoning for embodied agents. Unlike previous works that rely on a single generic prompt for all question types, we develop tailored prompting strategies for each question type, encouraging LLM to generate QA pairs focusing on different levels of reasoning.

Object Attribute and Relations include questions about objects attributes, such as *color*, *shape*, and *size*, while also incorporating situated spatial information. For instance, the questions to identify *"the color of the table positioned to the left"*, and determine *"how many pictures are hanging on the wall to the right"*.

Object Affordance focuses on the function utility of the objects, often based on common sense knowledge about how objects are used. Similarly, we require situated spatial information to be part of the answer. For example, when asked "*Where can you check your appearance?*", the correct answer should be *"mirror on your left"*, specifying both the object name (*mirror*) and its spatial location from the agent.

Situated Planning is the most challenging task, as it requires the agent to perform multi-hop situated spatial reasoning. The agent must not only recognize its surroundings but also plan and execute a series of actions across multiple steps, where each subsequent action depends on the outcome of the previous one. In our dataset, we implement 2-hop reasoning, which requires the agent to perform a sequence of two continuous actions. For example, given the example in Fig. 5.4, *"make the room brighter and then sit down to relax."*, the agent needs to first turn left from its



(a) Avg. human score for Spartun3D.

(b) Comparison between SQA3D and Spartun3D.

(c) Percentage of valid examples generated from two different prompts.

Figure 5.7 Human Evaluation of Spartun3D.

orientation to face and move toward the window, open it to brighten the room, then based on its new position, the agent continues turning left toward the sofa chairs and sits down.

5.3.1.4 Dataset Statistics and Quality Control

In total, we collect approximately **10k** situated captions and **123k** QA pairs. For the tasks of object attribute and relation and the tasks of affordance, we sampled around 10 situations per scene. For captioning and planning tasks, we sample around 5 situations per scene due to the increasing cost of longer token sequences required for these tasks. For each task, we split the data instances into a Training and Test set. Table 5.1 shows the statistics of our dataset.

We conduct a comprehensive human evaluation to manually assess the quality of Spartun3D, introducing human scores based on two key criteria: *language naturalness*, which evaluates whether the text reads as if it were naturally written by a human, and *spatial fidelity*, which ensures that the data accurately reflects the 3D scene with correct spatial relationships. Each criterion is rated on a scale from 1 to 5, and the average of these two scores is the overall human score. We randomly select 50 examples from each task and compute human scores of situation, question, and answer, respectively. As shown in Fig. 5.7 (a), the average scores align with the complexity of each task, with relatively lower scores for captioning and planning tasks. To assess how our generated data compares to human-annotated data, we sampled 50 examples from SQA3D and mix them with our dataset. We focus on the human score of different types of questions, as shown in Fig. 5.7 (b). We also evaluate how different prompting strategies influence the quality of the data. We experiment with two types of prompts for representing spatial information to prompt GPT-4o: **Cord-prompt**, which consists of object center coordinates, standing point, orientation, and instructions for calculating

Tasks	# of Examples	Train/Test
Captioning	$\sim$ 10K	$8,367/1,350$
Attr. & Rel.	$\sim$ 62K	$61,254/8,168$
Affordance	$\sim$ 40K	$35,070/5,017$
Planning	$\sim$ 21K	$19,434/2,819$

Table 5.1 Dataset statistics of Spartun3D and human validation results.

distances and rotation angles, and **Spa-prompt**, consisting of the calculated angles and distance based on the approaches we described in Sec. 5.3.1.3. Fig.5.7 (c) shows the percentage of examples with high human scores (≥ 4) for each prompt across tasks. The results indicate that Cord-prompt yields unsatisfactory results, revealing that LLMs lack strong 3D spatial reasoning capabilities when interpreting raw spatial coordinates. Our Spa-prompt significantly improves the quality of the generated dataset by providing qualitative spatial relations (*e.g.* distance, direction).

5.3.2 Model Architecture

In addition to enhancing the situated understanding of 3D-based LLMs with Spartun3D, we also propose a new 3D-based LLM, named ***Spartun3D-LLM***, which integrates a novel *Situated Spatial Alignment* module to strengthen the alignment between the situated 3D visual features and their corresponding textual descriptions. Spartun3D-LLM is built upon LEO [46], which represents the most recent and state-of-the-art 3D-based LLM, and directly takes 3D point cloud data as input, making it well-suited for spatial reasoning tasks in 3D environments. Fig. 5.8 illustrates the overview architecture of Spartun3D-LLM.

5.3.2.1 Background

Problem Formulation. We formally define the input as a triple $< C, S, Q >$, where C is the 3D scene context, S is the situation, and Q is a question. The situation S can be further denoted as $S =< S^t, S^p, S^r >$, where S^t is a textual situation description, and S^p and S^r are the standing points and orientation, respectively. Specifically, S^p is a 3D coordinate in the form $< x, y, z >$ and S^r is the quaternion $< qx, qy, qz, w >$, where $< qx, qy, qz >$ is the rotation axis and w is the rotation angle. For simplicity, we define $z = 0$ to calculate the rotation angle on a 2D plane. The task is to generate a textual answer, denoted as A, given scene context C, situation S, and question Q. During training,

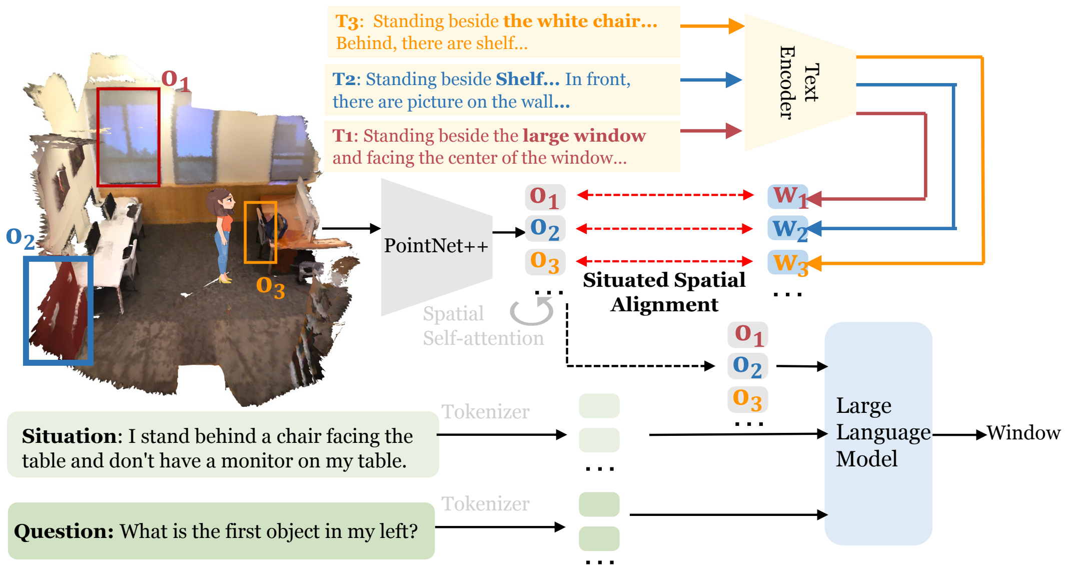

T3: Standing beside the white chair... Behind, there are shelf...
T2: Standing beside Shelf... In front, there are picture on the wall...
T1: Standing beside the large window and facing the center of the window...
Text Encoder
o1
o2
o3
PointNet++
w1
w2
w3
Situated Spatial Alignment
Spatial Self-attention
Large Language Model
Window
Tokenizer
Situation: I stand behind a chair facing the table and don't have a monitor on my table.
Question: What is the first object in my left?

Figure 5.8 Spartun3D-LLM Model Architecture.

S^p and S^r are provided to the agent to rotate and translate the environment, while during testing, only questions and situations are provided.

Backbone. LEO [46] takes the text, 2D image (optional), and 3D point clouds as input and formulate comprehensive 3D tasks as autoregressive sequence generation. Specifically, data from different modalities are converted into a sequence of tokens as input to the LLM. The text tokens include system messages (*e.g., "You are an AI visual assistant situated in a 3D scene."*), situations, and questions. These tokens are then embedded into vector representations using an embedding look-up table. For 3D point clouds, LEO first applies segmentation masks to extract the point clouds of individual objects in the 3D scenes. Then, the sampled points of each object are input into a object-centric point cloud encoder, PointNet++ [85] pre-trained on ScanNet [16], to obtain the object-level representations.

Formally, we denote the representation of input text tokens as $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M] \in \mathbb{R}^{M \times D}$, where M denotes the number of input tokens, and D represents the dimensionality of each token's embedding. Additionally, the input object visual representations are expressed as $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_K] \in \mathbb{R}^{K \times D}$, where K is the number of extracted objects from the scene. Finally, the output answer are represented as $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_N]$, where N is the number of tokens in the response. The model's objective is to generate the answer given these combined inputs. The loss for generating the i-th token of the output answer is formulated as follows:

$$\mathcal{L}_{\texttt{LM}}(\theta) = \sum_{i} logp_{\theta}(\mathbf{a}_i|\mathbf{a}_{i-1}, \mathbf{W}_{\mathbf{S}}, \mathbf{o}). \tag{5.4}$$

LEO can integrate various LLM backbones, including OPT1.3B [132] and Vicuna-7B [15]. In our experiments, we fine-tune LEO with different LLM backbones on our proposed dataset via LoRA [44].

5.3.2.2 Situated Spatial Alignment Module

Situated tasks require robust spatial reasoning abilities to comprehend the position, orientation, and spatial relationships of objects within a 3D environment. Existing 3D-based LLMs typically process inputs by concatenating output representations from various modality encoders. While this method facilitates the integration of data across different modalities, it does not inherently ensure that the 3D visual representations encode situated spatial information or effectively align with textual descriptions, which potentially limits the model's ability to perform tasks that require precise spatial understanding. To tackle this challenge, we introduce a novel *Situated Spatial Alignment Module* to improve the alignment between the object-centric 3D visual representations and their situated textual descriptions. The process begins by generating detailed situated textual descriptions for each object. Subsequently, an alignment loss is introduced, which directs the model in effectively learning the 3D visual representations based on these situated textual descriptions.

Situated Textual Descriptions. For each object, we construct a comprehensive situated textual description based on a template that captures the object's name, attributes, and spatial relations with nearby objects, as *"Stand besides {object name} and facing the center of the {object name}, in front, there are {a list of nearby objects}; on the right, ...; behind ...; and on the left..."*. The object's attributes are also considered (e.g., *"white chair"*). We consider up to five objects per direction. If no object is present in a specific direction, the description explicitly states this, ensuring to provide complete information about the 3D environment.

3D Object-Text Alignment. Inspired by the success of 2D Visual-Language models, which effectively leverage semantically aligned text and visual features to excel in downstream tasks [89, 63, 62, 115], we aim to enhance the 3D visual representations so that they can better encode the situated spatial information and effectively align with the textual descriptions. Specifically, we introduce a 3D object-text alignment loss to guide the learning process of point cloud encoders within 3D-based LLMs, leveraging the robust language representations captured by pre-trained text encoders. We experiment with various text encoders, and CLIP achieved the best performance.

More formally, we obtain the text representations of situated textual description for each object from pre-trained text encoders, denoted as $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_k] \in \mathbb{R}^{K\times D}$. For object visual representations $\mathbf{O}$, we employ spatial self-attention layers [11] to learn spatial-aware object representations. Specifically, a pairwise spatial feature matrix $\mathbf{F} \in \mathbb{R}^{K\times K\times 5}$ is introduced to represent relative spatial relations between objects. For example, for each object pairs $\mathbf{o}_i$ and $\mathbf{o}_j$, we construct pairwise spatial feature as $\mathbf{f}_{ij} = [d_{ij}, sin(\theta_h), cos(\theta_h), sin(\theta_v), cos(\theta_v)] \in \mathbb{R}^{1\times 5}$, where d_{ij} is Euclidean distance between two objects, and θ_h and θ_v are horizontal and vertical angles connecting bounding box centers of $\mathbf{o}_i$ and $\mathbf{o}_j$, respectively. Then, we inject $\mathbf{F}$ into the self-attention of the object as,

$$\mathbf{O}' = \texttt{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_h}} + \texttt{MLP}(\mathbf{F}))\mathbf{V}, \text{where } Q = W_Q O; K = W_K O, V = W_V O, \tag{5.5}$$

where $\mathbf{O}' \in \mathbb{R}^{K\times D}$ denotes the spatial-aware object representation; Then we use a Mean Squared Error (*i.e.*, `MSE`) as the objective function to minimize the distance between the object representation $\mathbf{O}'$ and the corresponding situated textual embedding $\mathbf{W}$, denoted as $\mathcal{L}_{\texttt{align}} = \texttt{MSE}(\mathbf{o}', \mathbf{w}_t)$. The model is trained to jointly optimize both the alignment loss and the language modeling loss (Eq. 5.4) as $\mathcal{L} = \mathcal{L}_{LM} + \mathcal{L}_{\texttt{align}}$.

5.4 Experiments for VLN-CE

5.4.1 Experimental Results on High-Level and Low-Level Action Performance

We evaluate our method on top of three VLN-CE agents: WP-VLN-BERT [38], WP-HAMT [115], and ETPNav [2]. These VLN-CE agents are all Transformer-based models and employ a waypoint predictor to discretize the visual environment for navigation in the continuous setting. WP-VLN-BERT represents historical information using implicit state representations, whereas WP-HAMT uses explicit panoramic images in the traversed path. ETPNav is a graph-based VLN agent, which

		Validation Unseen			Test Unseen	
	Model	nDTW↑	SR↑	SPL↑	SR↑	SPL ↑
1	Waypoint Models [52]	-	0.36	0.34	0.32	0.30
2	CWP-CMA [38]	0.55	0.41	0.36	0.38	0.33
3	Sim2Sim [53]	-	0.43	0.36	0.44	0.37
4	VLN-BERT+Ego2-Map [41]	0.60	0.52	0.46	0.47	0.41
5	WP-VLN-BERT [38]	0.54	0.44	0.39	0.42	0.36
6	**WP-VLN-BERT+Ours**	0.55	0.46	0.41	0.44	0.38
7	WP-HAMT [115]	0.60	0.52	0.47	0.49	0.45
8	**WP-HAMT+Ours**	**0.62**	0.54	**0.49**	0.52	0.47
9	ETPNav [2]	-	0.57	**0.49**	0.55	0.48
10	**ETPNav+Ours**	**0.62**	**0.58**	**0.49**	**0.56**	**0.48**

Table 5.2 Experimental results on **high-level** action evaluated on the R2R-CE validation unseen and test dataset.

differs slightly from the standard navigation setting. The other two agents can only select local navigable viewpoints connected to the current viewpoint. However, graph-based agents like ETPNav can jump back to the previously explored viewpoints, often resulting in a higher success rate. We integrate our dual-action module and enhanced waypoint predictor into the three baseline backbones introduced above and evaluate navigation performance on both high-level and low-level actions as follows.

High-Level Action Performance. Table 5.2 presents the navigation performance using high-level actions on the validation unseen and test unseen sets. All VLN-CE agents in Table 5.2 first employ a waypoint predictor to generate navigable viewpoints and then select a view from these viewpoints. We improve high-level action performance for all baselines by incorporating our obstacle-aware waypoint predictor and dual-action modules. Specifically, we improve WP-VLN-BERT almost 2% on all navigation metrics on both validation and test unseen sets, as shown in row#6. WP-HAMT utilizes visual representation from InternVideo [115] to strengthen the model's performance. We mainly compare with InternVideo base weights because of the computation cost limitation. In terms of the navigation performance of this baseline, we especilly improve 3% of the success rate on the test unseen (row#8). In addition to enhancing the standard Transformer-based navigation agent, our method can also increase the success rate of the graph-based agent ETPNav, as shown in row#10.

Low-Level Action performance. Table 5.3 shows the navigation performance with low-level

	Methods	nDTW↑	SR ↑	SPL ↑
1	CMA+PM+DA+Aug [54]	0.51	0.32	0.30
2	LAW [92]	0.54	0.35	0.31
3	WS-MGMap [9]	-	0.39	0.34
4	CWP-CMA [38]	0.49	0.27	0.25
5	VLN-BERT+Ego2-Map [41]	0.52	0.30	0.29
6	WP-VLN-BERT [38]	0.48	0.23	0.22
7	**WP-VLN-BERT+Ours**	0.54	0.28	0.27
7	WP-HAMT* [115]	0.54	0.35	0.32
9	**WP-HAMT+Ours**	0.55	0.44	0.38
10	**ETPNav+Ours**	**0.58**	**0.48**	**0.42**

Table 5.3 Experimental results of **low-level** actions on the R2R-CE validation unseen set. * means our implementation for low-level action prediction, as most VLN-CE agents do not report their low-level performance. We train the VLN agent with a low-level action classifier for fair comparison.

movement actions. The existing methods mainly compare the results of the validation unseen for low-level actions. The models in row#1 to row#3 are based on LSTM architecture to frame the navigation as a sequence-to-sequence task and predict low-level actions directly. The models from row#4 to row#6 are models using a waypoint predictor. They add an action classifier to the navigator and train the model to select one low-level action at each navigation step. Another noticeable difference between the models from row#1 to row#3 and others is that the agent in these methods can only observe the current view rather than the whole panoramic view at each navigation step. As shown in Table 5.3, we observe that some Transformer-based navigators with much more powerful pre-trained visual representations and complex model architecture, such as the approaches in row#5 and row#6, their low-level action navigation performance could not compete with LSTM-based models (row#1 to row#3). Specifically, for the baseline model WP-VLN-BERT, although our method can significantly improve it (row#7), it is still far behind LSTM-based models. However, we can achieve SOTA after applying our method on WP-HAMT and ETPNav, as shown in row#9 and row#10, respectively. It is worth noting that the majority of VLN-CE agents do not report their low-level action performance. To ensure a fair comparison, we follow the method of low-level action prediction in WP-VLN-BERT to add a non-linear classifier on top of WP-HAMT [115] to adapt it to low-level action prediction. In general, our method's performance is aligned with the VLN-CE

Method				High			Low		
	CLIP	Ob-Mask	Dual-Action	nDTW↑	SR↑	SPL↑	nDTW↑	SR↑	SPL↑
Baseline				0.60	0.52	0.47	0.54	0.35	0.32
1			✔	0.60	0.52	0.47	**0.55**	0.43	0.36
2	✔			0.61	0.53	0.47	-	-	-
3	✔	✔		0.61	0.53	0.48	-	-	-
4	✔	✔	✔	**0.62**	**0.54**	**0.49**	**0.55**	**0.44**	**0.38**

Table 5.4 Ablation study on different components of our method. The baseline is WP-HAMT, and the Ob-mask is the obstacle mask.

navigator's performance. This correlation is expected, as the low-level action sequence is trained using the hidden state representation from the corresponding baseline, and stronger representations yield better performance when training low-level actions.

5.4.2 Ablation Study

In this section, we conduct an ablation analysis of the waypoint predictor and for waypoint predictor and the effectiveness of different components in our method.

Waypoint Predictor Performance. The waypoint predictor is trained offline to generate navigable viewpoints. We enhance the baseline waypoint predictor from the aspects of stronger visual representations and explicit object masks, and Table 5.5 shows our results on R2R-CE validation unseen set. The main metrics to evaluate the waypoint predictor's performance are as follows: $|\Delta|$ measures the difference in the number of target waypoints and predicted waypoints. `%Open` is the ratio of predicted waypoints in open space. d_c and d_H are the Chamfer and Hausdorff distances, respectively, to measure the distance between point clouds.

We experiment with visual representations from different Vision and Language Pre-trained Models (VLMs) and test their influence on the performance of the waypoint predictor. As shown in Table 5.5, the waypoint predictor achieves the best performance when utilizing CLIP vision representations. However, we cannot conclude that more powerful vision representations lead to better waypoint-predicting performance since the representations from InterVideo seem to hurt the waypoint predictor. This result suggests that different pre-trained visual encoders possess varying capacities to influence the agent's ability to recognize open and obstacle areas. The better result is achieved when both CLIP representation and our designed obstacle mask are applied. Compared

		Waypoint Predictor			
	Visual Encoder	$\|\Delta\|$	%Open↑	$d_C \downarrow$	$d_H \downarrow$
1	ResNet [38]	1.40	0.80	1.07	2.00
2	InternVideo [115]	1.44	0.65	1.15	2.04
3	DenseCLIP [91]	1.41	0.81	1.05	2.01
4	CLIP [89]	**1.38**	0.83	**1.04**	2.00
5	CLIP+ Obstacle Mask	**1.38**	**0.85**	**1.04**	**1.94**

Table 5.5 Evaluation of different visual encoders.

to ResNet (row#1), CLIP improves the open area prediction by about 3%, demonstrating that rich semantics in visual representation in CLIP aids the waypoint predictor in learning the open and obstacle objects. After applying our designed obstacle mask, the accuracy of open area prediction gained an additional improvement of 2%, emphasizing the effectiveness of the prior knowledge in encouraging the waypoint predictor to better focus on open area spaces.

Different Components. The results of the ablation study in Table 5.4 demonstrate the influence of each component of our proposed method on both high-level and low-level navigation. The components in our method include dual-action for the navigator and the enhanced waypoint predictor with CLIP visual representations and the obstacle mask. The analyzed navigator is WP-HAMT, which uses visual representation obtained from InternVideo base weights. We report results on the R2R-CE validation unseen dataset. In row#1, we integrate the low-level action decoder with the baseline navigator and jointly train it with high-level actions, and the low-level action navigation performance is significantly improved (about 4% on SPL). In row#2, we train the waypoint predictor with only CLIP representations and apply it to the baseline navigator without dual-action training. Notably, the enhanced waypoint predictor already contributes to better high-level navigation performance, indicating that CLIP's rich object semantic representation boosts overall navigation. Row#3 shows the effectiveness of the obstacle mask in enhancing the SPL for high-level action. In row#4, we train the waypoint predictor with both CLIP and obstacle mask, and we apply this enhanced predictor to the navigator with the dual-action module. We achieve the best results in this setting. Compared to row#3, we conclude that the spatial information incorporated into the low-level action benefits the high-level viewpoint selection. Similarly, compared to row#1, enhanced waypoint prediction not

		Waypoint Predictor			
	Ob-Mask Vocab	$\|\Delta\|$	%Open↑	$d_C \downarrow$	$d_H \downarrow$
	No Mask	**1.38**	0.83	**1.04**	2.00
1	Floor	**1.38**	0.84	1.07	2.00
2	Stairs	1.40	0.82	1.04	2.00
3	Doors	1.40	0.80	1.04	1.94
4	Floor+Stairs+Doors	**1.38**	**0.85**	**1.04**	**1.94**

Table 5.6 Analysis of the influence of various open-area vocabularies on the waypoint predictor.

only enhances high-level viewpoint selection but also benefits low-level action generation.

5.4.3 Qualitative Analysis

In this section, we provide a qualitative analysis from the perspectives of low-level actions and obstacle masks.

Low-Level Actions Generation. In Fig. 5.9 (1), we show an example of our generated low-level action sequences that lead the agent to the destination. However, we have observed cases where the agent generates a low-level action sequence that reaches the destination but does not fully follow the instruction, as shown in Fig. 5.9 (2). This issue also occurs in other VLN-CE agents when modeling low-level action predictions. We assume the reason is the inherent challenges in building the VLN-CE dataset. It transfers the instructions and trajectories from VLN-DE. When the simulator executes the low-level actions, especially rotations, there is no human evaluation process to confirm whether the low-level actions align with the instruction. The actions are generated based on minimal required rotations when the selected view is given to the simulator. Training the agent to learn low-level actions in these scenarios is more challenging compared to directly training with view selection.

Open-area Vocabulary. In Table 5.6, we provide an analysis of object semantics and their relation with the performance of the waypoint predictor. We select open-area vocabularies based on our prior knowledge. The baseline we used is the waypoint predictor trained with CLIP visual representations. Given the semantic segmentation from the simulated environment, we mask other object areas except the semantic areas in open-area vocabularies. Then, we input the masked image into the waypoint predictor. The experimental results demonstrate that different object semantics

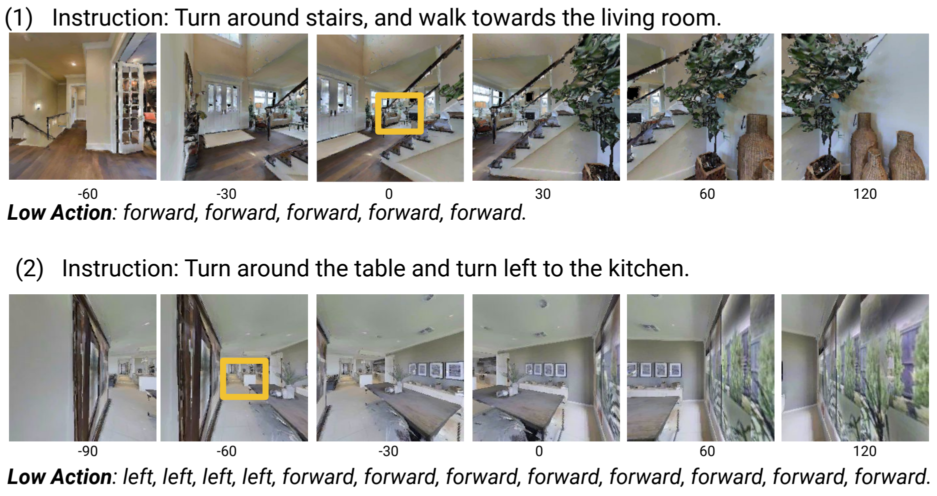

(1) Instruction: Turn around stairs, and walk towards the living room.
-60
-30
0
30
60
120
Low Action: forward, forward, forward, forward, forward.
(2) Instruction: Turn around the table and turn left to the kitchen.
-90
-60
-30
0
60
120
Low Action: left, left, left, left, forward, forward, forward, forward, forward, forward, forward, forward.

Figure 5.9 Examples of generated low-level actions. 0 denotes the current direction, while − means `LEFT` turn. The number represents the rotation degree. The yellow bounding box indicates the target.

show varying influences on the waypoint predictor. For instance, the %open is low when we mask objects other than "*door*", indicating the presence of closed or blocked doors (row#3). We can get the best results when our open-area vocabularies contain "*floor*", "*stairs*", and "*doors*" (row#4).

Qualitative Examples for Obstacle Mask. In Fig. 5.10, we show an example to demonstrate the different generated waypoint heatmaps between an RGB image with (Fig. 5.10 (2)) and without (Fig. 5.10 (1)) obstacle mask. The image shows the corresponding views based on the headings of the highlighted areas in the heatmap. It is evident that the waypoint predictor samples more viewpoints (5 viewpoints) from image (a) and image (b) when an obstacle mask is applied, both of which contain large open areas. In contrast, RGB images without obstacle masks sample relatively fewer viewpoints on images (a) and (b), but they sample viewpoints from (c) and (d), although (c) is not included in the ground truth. This example illustrates that the obstacle mask aids the waypoint predictor in concentrating mainly on large open areas but falls short in narrow open areas. However, based on the final navigation result in Table 5.4, the obstacle mask ultimately contributes to navigation performance.

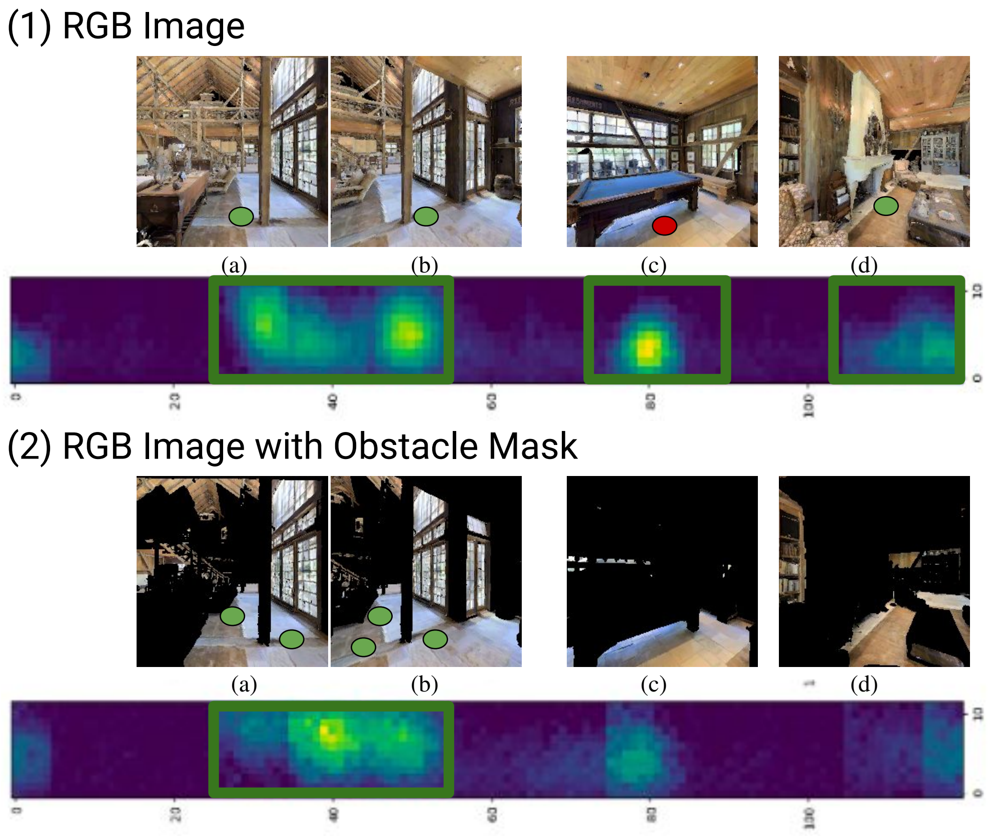


Figure 5.10 An example of a generated waypoint heatmap given an RGB image with and without obstacle mask.

Models	LLMs	Attributes and Relations					Affordance					Situated Planning				
		C	B-4	M	R	EM	C	B-4	M	R	S	C	B-4	M	R	S
LEO	zero-shot	100.3	0.00	17.4	39.1	42.7	13.3	0.00	3.00	5.00	32.3	0.00	0.00	7.00	15.3	59.2
LEO+Spartun3D	OPT1.3B	121.3	7.0	20.1	45.3	47.7	224.6	30.6	24.9	53.2	66.9	229.7	44.8	32.1	60.9	83.8
	Vicuna7B	125.4	10.1	22.1	46.7	52.1	238.9	32.1	24.4	55.0	68.3	242.1	46.5	35.2	63.1	84.3
LEO*+Spartun3D	Vicuna7B	129.2	10.4	23.0	48.1	53.2	211.3	32.1	24.6	55.0	67.8	247.1	47.5	36.2	65.1	85.8
Spartune3D-LLM	OPT1.3B	124.1	9.2	21.0	47.3	49.4	227.2	31.4	26.3	54.1	68.2	232.3	45.2	33.2	62.1	85.4
	Vicuna7B	131.2	10.3	24.3	48.8	53.7	240.4	32.1	25.0	55.3	68.7	244.0	47.1	36.4	64.0	86.8
Spartune3D-LLM*	Vicuna7B	**135.4**	10.7	24.9	51.3	**56.9**	**254.7**	**32.9**	**26.7**	**57.3**	**69.7**	**252.1**	47.6	36.2	**65.4**	**88.7**

Table 5.7 Experimental Results on Spartun3D Situated QA Tasks. $*$ represents the model initialized with LEO instruction-tuned weights. [Keys: C: CIDER; B-4: BLEU-4; M: METEOR; R: ROUGE; Sim: Sentence Similarity; EM: Exact Match; **Bold: best results**].

5.5 Experiments for Spartun3D

5.5.1 Experimental Setup

To demonstrate the effectiveness of our proposed Spartun3D-LLM, we conduct experiments on two situated understanding datasets, including Spartun3D and SQA3D [74]. For SQA3D, we evaluate under two conditions: object proposals in 3D are derived either from Mask3D [98] or ground-truth annotations. Also, we assess the transferability of our method on the navigation task using MP3D ObjNav [97]. Following LEO [46], we report the performance using standard generation metrics, including *CIDEr*, *METEOR*, *BLEU-4*, and *ROUGE_L*, sentence similarity [94] for captioning task. For SQA3D and situated QA tasks of questions about attributes and relations, we also report an additional metric of exact-match accuracy. We leverage LEO as baseline. Since the training stage in LEO has covered most of the evaluation tasks, we experiment with models initialized from scratch to ensure a fair comparison in the zero-shot setting. For other settings, we report the performance of models initialized both from scratch and from the instruction-tuned LEO. To distinguish between the two, models initialized from the instruction-tuned LEO are marked with an asterisk (*).

5.5.2 Experimental Results on Different Tasks

Spartun3D Benchmark We evaluate the performance of both the LEO model and Spartun3D-LLM after fine-tuning them on our proposed Spartun3D dataset. The fine-tuned LEO model is referred to as LEO+Spartun3D. Table 5.7 and Table 5.8 show the experimental results on Situated Captioning and Situated QA tasks, respectively. We experiment with two different LLM backbones: Opt1.3B and Vicuna7B. Our experiments show that Spartun3D-LLM consistently outperforms LEO+Spartun3D across all question types (around $2\% - 3\%$ across all metrics), regardless of the LLM backbone used, indicating the effectiveness of our explicit alignment module. We observe that initializing our model with LEO pre-trained weights improves performance. Notably, without fine-tuning, LEO performs reasonably well on attribute and relation questions in a zero-shot setting but struggles with other situated tasks.

SQA3D Performance. We evaluate our method on the SQA3D dataset, whose scenes are derived

Models	LLMs	C	B-4	M	R	S
LEO	zero-shot	0.00	0.00	9.00	15.3	51.9
LEO+Spartun3D	OPT1.3B	5.9	15.3	17.7	31.2	67.3
	Vicuna7B	6.7	15.8	18.7	32.3	70.4
LEO+Spartun3D*	Vicuna7B	14.1	17.2	22.6	32.1	76.3
Spartun3D-LLM	OPT1.3B	6.4	15.7	18.5	31.2	68.6
	Vicuna7B	8.5	16.4	19.6	32.5	72.5
Spartun3D-LLM*	Vicuna7B	**14.6**	**19.3**	**23.3**	**33.4**	**78.1**

Table 5.8 Experimental Results on Spartun3D Situated Captioning Task.

	#	Methods	Mask3D [98]				GT			
			C	M	R	EM	C	M	R	EM
Zero-shot	1	LEO [46]	14.2	6.4	8.2	12.4	15.3	6.7	8.6	13.9
	2	LEO+Spartun3D	82.3	14.2	32.8	34.7	83.1	15.2	33.7	35.9
	3	Spartun3D-LLM	83.5	15.7	34.7	36.2	85.6	16.6	35.8	37.1
Fine-tune	4	3D-Vista [153]	-	-	-	48.5	-	-	-	-
	5	3D-LLM [42]	-	-	-	50.2	-	-	-	-
	6	LEO [46]	132.0	33.0	49.2	52.4	132.3	34.3	51.4	52.5
	7	LEO*+Spartun3D	134.0	34.6	52.2	53.5	135.3	34.2	52.1	54.2
	8	Spartun3D-LLM*	**138.2**	**35.3**	**53.4**	**54.8**	**138.3**	**35.4**	**53.7**	**55.0**

Table 5.9 Experimental Results on SQA3D given the 3D objects from Mask3D and Ground-truth.

from ScanNet [16]. Their scenes differ from those in Spartun3D, which are sourced from 3RScan. We experiment with two settings: zero-shot and fine-tuning. In the zero-shot setting, we re-trained LEO on their dataset (row#1) only constructed from 3RScan excluding all dataset constructed from ScanNet to ensure a fair comparison with our method. As shown in Table 5.9, LEO performs poorly on SQA3D in the zero-shot setting, suggesting its limitations in learning situated understanding from its original dataset. In contrast, LEO trained on Spartun3D (row#2) shows significant improvement, demonstrating the effectiveness of our dataset. Further comparisons of Spartun3D-LLM with LEO+Spartun3D demonstrate a better zero-shot learning (i.e., generalization) capability of our model. In the fine-tuning setting, Spartun3D-LLM continues to outperform LEO across all metrics.

Navigation Performance. To demonstrate the effectiveness of our approach on downstream embodied tasks, we evaluate it on the object navigation tasks. Specifically, we randomly select 5 scenes that contain around 1000 examples from the MP3D ObjNav dataset. In this task, we additionally input 2D ego-centric images to both LEO and Spartun3D-LLM for comparison. There

	LEO	Spartun3D-LLM
Zero-shot	0	**20.3**

Table 5.10 Navigation Performance (Accuracy%).

Methods	Scan2Cap	ScanQA
LEO + Spartun3D	54.2	46.3
Spartun3D-LLM	**55.7**	**48.6**

Table 5.11 Spatial Alignment Evaluation on Other Benchmarks. The metric is Sentence Similarity.

are four types of navigation actions: turn left, turn right, move forward, and stop. We evaluate whether the model generates correct action at each step. We conduct the experiment in a zero-shot setting, and Table 5.10 shows the accuracy of the model's performance. The baseline model, LEO, struggles to generate the required action-related text to guide navigation steps without fine-tuning specifically for navigation tasks. In contrast, our model demonstrates strong transferability to generate correct actions. Fig 5.13 (e) showcases a qualitative example, illustrating how our model effectively generates accurate navigation actions without task-specific fine-tuning.

5.5.3 Ablation Study and Extra Analysis

Explicit Alignment Enhances General Spatial Understanding. We evaluate the effectiveness of our proposed situated spatial alignment module on general scene understanding tasks, such as Scan2Cap [14] and ScanQA [6]. In line with our approach for situated tasks, we construct textual descriptions for each object based on its attributes and spatial relations to others from a top-view perspective. As shown in Tab. 5.11, by incorporating the explicit spatial alignment module, our model shows better results, indicating that our proposed alignment module not only improves situated understanding but also enhances general 3D scene understanding.

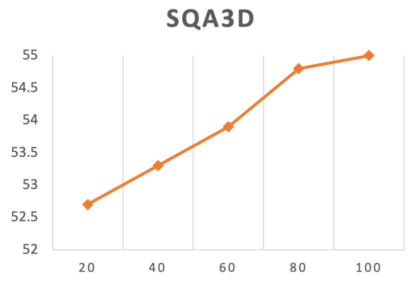


Figure 5.11 Scaling Effects.

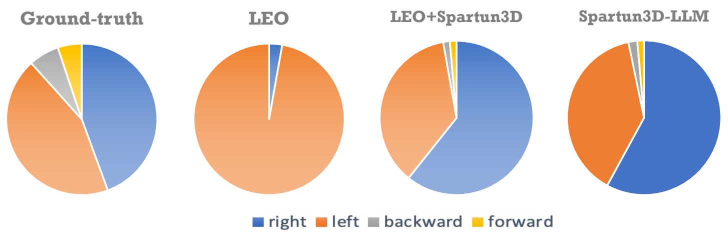


Figure 5.12 SQA3D Labels Distribution.

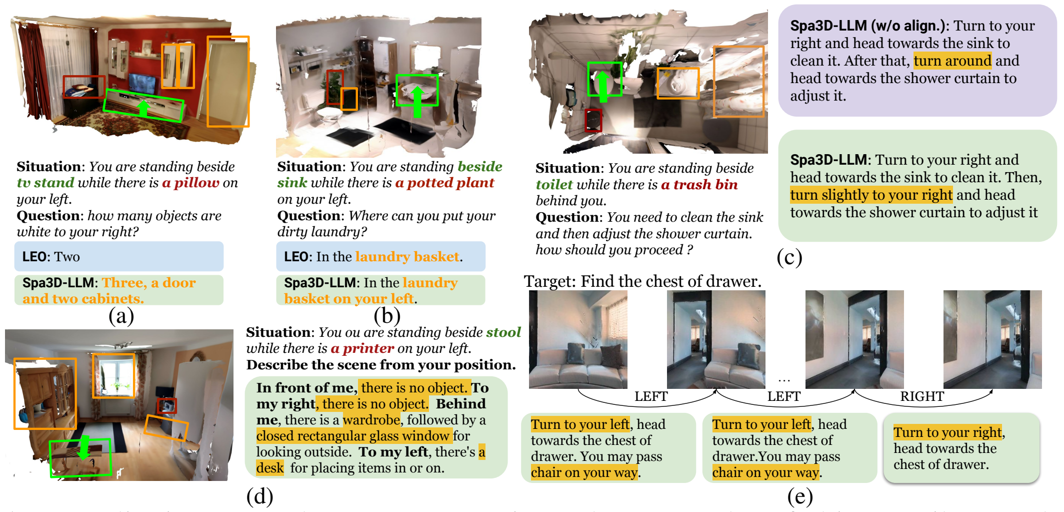
Improved Situated Understanding. To analyze the model's situated understanding ability further, we visualize the distribution of model responses generated for questions requiring strong spatial understanding from SQA3D. Specifically, we extract questions starting with *"which direction"*. Fig. 5.12 illustrates the distribution of generated *"directions"*, including *"left"*, *"right"*, *"forward"* and *"backward"*. We observe that LEO is biased towards generating *"left"* 97% of the time. However, the ground-truth distribution of *"left"* and *"right"* should be balanced, suggesting that LEO may have a limited understanding of situated spatial relationships. The bias is significantly mitigated when LEO is trained on our dataset (LEO+Spartun3D). While adding our alignment loss (Spartun3D-LLM) helps futher, our dataset is the primary factor in addressing the bias.

Scaling Performance. We conduct scaling experiments to demonstrate how model performance improves with the addition of Spartun3D datasets. As shown in Fig. 5.11, we evaluate performance on SQA3D and observe consistent improvement as the dataset scales, highlighting the potential for dataset expansion using our proposed method.

Qualitative Examples. In Fig. 5.13, we showcase several successful examples to demonstrate the effectiveness of Spartun3D-LLM across various situated tasks. Notably, in Fig 5.13 (c), the model without an explicit alignment module tends to generate more general or vague spatial descriptions, such as *"turn around"*. In contrast, with the alignment module, the model produces more specific details, including terms like *"turn slightly right"*. To verify this, we examine 30 examples from both situated planning and situated captioning tasks and observe this phenomenon in 17 of them. This highlights how the proposed spatial alignment module enhances the generation of fine-grained spatial information, leading to more precise and contextually accurate outputs.

5.6 Conclusion

In this chapter, we bridge the gap between simulation and real-world challenges from two key perspectives: transitioning from discrete to continuous navigation and improving spatial reasoning in 3D environments. For VLN-CE agents, we focus on narrowing the gap between visual perception and action grounding. We introduce a dual-action module that enables the current VLN-CE agent, equipped with the waypoint predictor, to train jointly for both high-level and low-level actions.


(a)

Situation: *You are standing beside **tv stand** while there is **a pillow** on your left.*
Question: *how many objects are white to your right?*

LEO: Two

Spa3D-LLM: **Three, a door and two cabinets.**

(b)

Situation: *You are standing **beside sink** while there is **a potted plant** on your left.*
Question: *Where can you put your dirty laundry?*

LEO: In the **laundry basket**.

Spa3D-LLM: In the **laundry basket on your left**.

(c)

Situation: *You are standing beside **toilet** while there is **a trash bin** behind you.*
Question: *You need to clean the sink and then adjust the shower curtain. how should you proceed ?*

Spa3D-LLM (w/o align.): Turn to your right and head towards the sink to clean it. After that, turn around and head towards the shower curtain to adjust it.

Spa3D-LLM: Turn to your right and head towards the sink to clean it. Then, turn slightly to your right and head towards the shower curtain to adjust it

(d)

Situation: *You ou are standing beside **stool** while there is **a printer** on your left.*
Describe the scene from your position.

In front of me, there is no object. **To my right**, there is no object. **Behind me**, there is a wardrobe, followed by a closed rectangular glass window for looking outside. **To my left**, there's a desk for placing items in or on.

(e)

Target: Find the chest of drawer.

Turn to your left, head towards the chest of drawer. You may pass chair on your way.

Turn to your left, head towards the chest of drawer.You may pass chair on your way.

Turn to your right, head towards the chest of drawer.

Figure 5.13 Qualitative Examples. (a), (c), (c) situated qa examples of object attribute and relation, object affordance, and situated planning. (d) situated captioning example. (e) navigation example in a zero-shot setting.

This joint optimization encourages the agent to learn to ground the high-level visual perception and view selection into physical actions and spatial motions. Second, we enhance the existing waypoint predictors by incorporating rich object semantic representations and knowledge about object properties. This helps the model to consider the feasibility of actions in their navigation decisions. For spatial understanding in 3D world, we tackle the limitations of 3D-based LLMs in situated understanding from two perspectives. First, we propose a method to construct an LLM-generated dataset based on our designed situated scene graph. Then, we propose an explicit situated spatial alignment on the 3D-LLM to encourage the model to learn alignment between 3D object and their textual representations directly. Finally, we provide comprehensive experiments to show our own benchmark improve situated understanding of SQA3D and navigation.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we aim to enhance the language grounding capabilities of VLN agents. These agents interpret natural language instructions and align them with their visual observations to make accurate action decisions. Effective language grounding is essential for improving both navigation performance and the interpretability of the agent's decision-making process. The key contributions of this thesis are summarized as follows.

6.1 Summary of Contributions

Enhancing VLN Agent Grounding via Explicit Modulation of Spatial Semantics. Most VLN agents overlook explicit spatial semantics modeling, relying primarily on implicit representation learning to align semantics across different modalities. While these methods significantly enhance navigation performance, they compromise the interpretability of the agent's decision-making process. To address this challenge, we primarily focus on the spatial semantics of motion-related and landmark-related information, and introduce two neural navigation agents with modular designs tailored to effectively learn these key semantics. The first method segments long instructions into spatial-semantic units, each consisting of motions and landmarks. We then identify key landmarks based on navigation progress and align them with the most relevant objects in the environment. Additionally, we model spatial relations between the landmark and the agent across both textual and visual modalities. The second method involves designing two independent modules to separately learn orientation (motion) and vision (landmarks). To achieve this, we introduce novel pre-training tasks tailored for orientation learning and visual perception. These tasks enhance the model's ability to understand different semantics effectively, which are subsequently leveraged within their respective modules to improve overall performance.

Aligning Instructions with Agent Perception and Explaining Decision-Making. Although our explicit grounding methods enhance the navigation agent's language understanding and align corresponding semantics with the visual environment, we observe ambiguities in instructions, particularly when they contain landmarks that are either unrecognized or indistinctive from the agent's vision perception. Such ambiguities negatively impact the language grounding ability, leading to challenges in improving navigation performance. To address this challenge, we first introduce a translator module to convert the original ambiguous instructions into easy-to-follow sub-instruction representations. Our design encourages the agent to interpret instructions in alignment with its visual perception. However, the translator's design relies on implicit learning, making it challenging to explicitly understand the agent's difficulties in interpreting human instructions. To address this, we further design an explainer module for the VLN agent, utilizing a language model to generate explanations that describe the ambiguity in instructions and the rationale behind the agent's action decisions.

Advancing VLN for Real-World Challenges: Navigation in Continuous and 3D Environments. We advance research on enhancing the applicability of navigation agents to real-world robotic systems from two key perspectives. First, we address navigation in continuous and unstructured environments, where agents must operate using low-level control commands rather than predefined high-level actions. Despite recent progress, existing navigation agents in continuous environments often overlook the crucial role of language grounding, particularly in the execution of low-level actions. To address this gap, we introduce a dual-action-perception module that connects linguistic instructions to the agent's low-level action space. Second, we extend navigation capabilities to 3D environments, where agents must reason about spatial relationships in three-dimensional space rather than relying solely on 2D image. While LLMs have demonstrated strong reasoning capabilities in 3D spatial contexts, they lack the essential ability for situated spatial understanding—a key requirement for navigation tasks. To address this limitation, we propose a scalable, LLM-generated dataset enriched with situated spatial information and introduce a spatial alignment module to improve the correspondence between 3D visual representations and their textual descriptions. Our method significantly enhances the LLM's situated spatial understanding ability in the 3D world, ultimately improving navigation performance.

6.2 Future Directions

This section highlights several promising avenues for future research that extend our findings and methodologies. Beyond the work presented in this thesis, we outline potential future directions from the following aspects.

Structured and Interpretable Planning with Foundation Models. Large generative models have demonstrated strong generalization capabilities across various domains. Integrating these models with planning strategies has significantly improved performance across various embodied tasks, such as robotic navigation [146], object manipulation [66], and interactive task execution [116]. However, while foundation models excel at generating natural language outputs, they inherently struggle to produce structured representations such as graphs, decision trees, or task flows. These structured outputs are crucial for downstream tasks, as they facilitate transparent, interpretable, and hierarchically organized reasoning. Therefore, we should focus on equipping large generative models with the ability to generate structured representations, thereby addressing key challenges in decision support for embodied navigation agents. Specifically, developing models that can decompose high-level goals into executable subtasks—represented as hierarchical workflows or decision trees—could significantly improve task planning and execution [129]. Such models can be explored and validated using various embodied benchmarks like ALFRED [100], BEHAVIOR [102], and VirtualHome [84], which offer controlled and diverse environments for evaluating embodied task performance.

Adaptive Usage of Foundation Models. Inspired by [81], which use LLMs to generate plans by dynamically decomposing complex sub-tasks as needed, we emphasize the importance of leveraging foundation models more strategically in embodied agents. An interesting future direction would be using foundation models as auxiliary systems that provide guidance and decision-making support when necessary. Specifically, future efforts could focus on two key aspects: 1) Difficulty Analysis: Developing systematic methods to analyze the challenges faced by the model in specific scenarios, such as ambiguous instructions/descriptions. 2) Query Mechanisms: Designing intelligent mechanisms that enable the agent to “ask for help" from the foundation model only when necessary.

This includes determining when and how to formulate queries and integrating responses seamlessly into the model's decision-making pipeline.

Compositional Learning. Another interesting direction is compositional learning, where the goal is to decompose complex tasks into smaller and reusable skills [43, 128]. Instead of learning complex embodied tasks, the model can be designed to acquire key concepts in both language and vision. Specifically, it converts language into structured representations, such as programming languages, and uses more formal methods to represent visual concepts like objects and their attributes. We can also develop "action concepts" to learn fundamental units of action policies. This line of research holds a potential to bridge high-level reasoning with low-level execution, enabling the model to tackle different tasks with greater adaptability and robustness.

BIBLIOGRAPHY

[1] Dong An, Yuankai Qi, Yan Huang, Qi Wu, Liang Wang, and Tieniu Tan. Neighbor-view enhanced model for vision and language navigation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 5101–5109, 2021.

[2] Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *arXiv preprint arXiv:2304.03047*, 2023.

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

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

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

[6] Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19129–19139, 2022.

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

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

[9] Peihao Chen, Dongyu Ji, Kunyang Lin, Runhao Zeng, Thomas Li, Mingkui Tan, and Chuang Gan. Weakly-supervised multi-granularity map learning for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 35:38149–38161, 2022.

[10] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 34, 2021.

[11] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Language conditioned spatial relation reasoning for 3d object grounding. *Advances in neural information processing systems*, 35:20522–20535, 2022.

[12] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. 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, 2022.

[13] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. 2019.

[14] Zhenyu Chen, Ali Gholami, Matthias Nießner, and Angel X Chang. Scan2cap: Context-aware dense captioning in rgb-d scans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3193–3203, 2021.

[15] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[16] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[17] Soham Dan, Parisa Kordjamshidi, Julia Bonn, Archna Bhatia, Zheng Cai, Martha Palmer, and Dan Roth. From spatial relations to spatial configurations. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5855–5864, Marseille, France, May 2020. European Language Resources Association.

[18] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–10, 2018.

[19] Jian Ding, Nan Xue, Gui-Song Xia, and Dengxin Dai. Decoupling zero-shot semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11583–11592, 2022.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[21] Zi-Yi Dou and Nanyun Peng. Foam: A follower-aware speaker model for vision-and-language navigation. *arXiv preprint arXiv:2206.04294*, 2022.

[22] Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. A survey of vision-language pre-trained models. *arXiv preprint arXiv:2202.10936*, 2022.

[23] Yu Du, Fangyun Wei, Zihe Zhang, Miaojing Shi, Yue Gao, and Guoqi Li. Learning to prompt for open-vocabulary object detection with vision-language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14084–14093, 2022.

[24] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244, 2022.

[25] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, pages 3314–3325, 2018.

[26] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *International Journal of Computer Vision*, 132(2):581–595, 2024.

[27] Georgios Georgakis, Yimeng Li, and Jana Kosecka. Simultaneous mapping and target driven navigation. *arXiv preprint arXiv:1911.07980*, 2019.

[28] Mehdi Ghanimifard and Simon Dobnik. What goes into a word: generating image descriptions with top-down spatial knowledge. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 540–551, 2019.

[29] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. *arXiv preprint arXiv:2104.13921*, 2021.

[30] Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. Airbert: In-domain pretraining for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1634–1643, 2021.

[31] Xiao Guo, Xiaohong Liu, Iacopo Masi, and Xiaoming Liu. Language-guided hierarchical fine-grained image forgery detection and localization. *IJCV*, 2024.

[32] Xiao Guo, Xiufeng Song, Yue Zhang, Xiaohong Liu, and Xiaoming Liu. Rethinking vision-language model in face forensics: Multi-modal interpretable forged face detector. *arXiv preprint arXiv:2503.20188*, 2025.

[33] Xiao Guo, Manh Tran, Jiaxin Cheng, and Xiaoming Liu. Dense-face: Personalized face generation model via dense annotation prediction. *arXiv preprint arXiv:2412.18149*, 2024.

[34] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146, 2020.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[36] Yicong Hong, Cristian Rodriguez, Yuankai Qi, Qi Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation. *Advances in Neural Information Processing Systems*, 33:7685–7696, 2020.

[37] Yicong Hong, Cristian Rodriguez-Opazo, Qi Wu, and Stephen Gould. Sub-instruction aware vision-and-language navigation. *arXiv preprint arXiv:2004.02707*, 2020.

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

[39] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. *arXiv preprint arXiv:2011.13922*, 2020.

[40] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. 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, 2021.

[41] Yicong Hong, Yang Zhou, Ruiyi Zhang, Franck Dernoncourt, Trung Bui, Stephen Gould, and Hao Tan. Learning navigational visual representations with semantic map supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3055–3067, 2023.

[42] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. *Advances in Neural Information Processing Systems*, 36:20482–20494, 2023.

[43] Joy Hsu, Jiayuan Mao, Josh Tenenbaum, and Jiajun Wu. What's left? concept grounding with logic-enhanced foundation models. *Advances in Neural Information Processing Systems*, 36, 2024.

[44] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[45] Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Kate Saenko, et al. Are you looking? grounding to multiple modalities in vision-and-language navigation. *arXiv preprint arXiv:1906.00347*, 2019.

[46] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. *arXiv preprint arXiv:2311.12871*, 2023.

[47] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. *arXiv preprint arXiv:1907.05446*, 2019.

[48] Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. *arXiv preprint arXiv:1905.12255*, 2019.

[49] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 259–266. IEEE, 2010.

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

[51] Parisa Kordjamshidi, Marie-Francine Moens, and Martijn van Otterlo. Spatial Role Labeling: Task definition and annotation scheme. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, pages 413–420. European Language Resources Association (ELRA), 2010.

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

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

[54] Jacob Krantz, Erik Wijmans, Arjun Majundar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision and language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*, 2020.

[55] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *arXiv preprint arXiv:2010.07954*, 2020.

[56] Jialu Li and Mohit Bansal. Improving vision-and-language navigation by generating future-view image semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10803–10812, 2023.

[57] Jialu Li and Mohit Bansal. Panogen: Text-conditioned panoramic environment generation for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 36:21878–21894, 2023.

[58] Jialu Li, Aishwarya Padmakumar, Gaurav Sukhatme, and Mohit Bansal. Vln-video: Utilizing driving videos for outdoor vision-and-language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18517–18526, 2024.

[59] Jialu Li, Hao Tan, and Mohit Bansal. Improving cross-modal alignment in vision language navigation via syntactic information. *arXiv preprint arXiv:2104.09580*, 2021.

[60] Jialu Li, Hao Tan, and Mohit Bansal. Clear: Improving vision-language navigation with cross-lingual, environment-agnostic representations. *arXiv preprint arXiv:2207.02185*, 2022.

[61] Jialu Li, Hao Tan, and Mohit Bansal. Envedit: Environment editing for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15407–15417, 2022.

[62] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[63] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.

[64] Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems*, 34:9694–9705, 2021.

[65] Liunian Harold Li, Mark Yatskar, D Yin, CJ Hsieh, and KW Chang. Visualbert: A simple and performant baseline for vision and language. arxiv 2019. *arXiv preprint arXiv:1908.03557*, 3, 1908.

[66] Xiaoqi Li, Mingxu Zhang, Yiran Geng, Haoran Geng, Yuxing Long, Yan Shen, Renrui Zhang, Jiaming Liu, and Hao Dong. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18061–18070, 2023.

[67] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah Smith, and Yejin Choi. Robust navigation with language pretraining and stochastic sampling. *arXiv preprint arXiv:1909.02244*, 2019.

[68] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020.

[69] Xiwen Liang, Fengda Zhu, Yi Zhu, Bingqian Lin, Bing Wang, and Xiaodan Liang. Contrastive instruction-trajectory learning for vision-language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1592–1600, 2022.

[70] Bingqian Lin, Yi Zhu, Zicong Chen, Xiwen Liang, Jianzhuang Liu, and Xiaodan Liang. Adapt: Vision-language navigation with modality-aligned action prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15396–15406, 2022.

[71] Chong Liu, Fengda Zhu, Xiaojun Chang, Xiaodan Liang, Zongyuan Ge, and Yi-Dong Shen. Vision-language navigation with random environmental mixup. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1644–1654, 2021.

[72] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

[73] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*, 2019.

[74] Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. Sqa3d: Situated question answering in 3d scenes. *arXiv preprint arXiv:2210.07474*, 2022.

[75] Lina Mezghani, Sainbayar Sukhbaatar, Arthur Szlam, Armand Joulin, and Piotr Bojanowski. Learning to visually navigate in photorealistic environments without any supervision. *arXiv preprint arXiv:2004.04954*, 2020.

[76] Dmytro Mishkin, Alexey Dosovitskiy, and Vladlen Koltun. Benchmarking classic and learned navigation in complex 3d environments. *arXiv preprint arXiv:1901.10915*, 2019.

[77] Ron Mokady, Amir Hertz, and Amit H Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.

[78] OpenAI. Hello gpt-4o, 2024.

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

[80] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[81] Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. Adapt: As-needed decomposition and planning with language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4226–4252, 2024.

[82] Tanawan Premsri and Parisa Kordjamshidi. Neuro-symbolic training for reasoning over spatial language. *arXiv preprint arXiv:2406.13828*, 2024.

[83] Tanawan Premsri and Parisa Kordjamshidi. Forest: Frame of reference evaluation in spatial reasoning tasks. *arXiv preprint arXiv:2502.17775*, 2025.

[84] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502, 2018.

[85] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[86] Yuankai Qi, Zizheng Pan, Shengping Zhang, Anton van den Hengel, and Qi Wu. Object-and-action aware model for visual language navigation. In *European Conference on Computer Vision*, pages 303–317. Springer, 2020.

[87] Yanyuan Qiao, Qianyi Liu, Jiajun Liu, Jing Liu, and Qi Wu. Llm as copilot for coarse-grained vision-and-language navigation. In *European Conference on Computer Vision*, pages 459–476. Springer, 2024.

[88] Yanyuan Qiao, Yuankai Qi, Yicong Hong, Zheng Yu, Peng Wang, and Qi Wu. Hop: History-and-order aware pre-training for vision-and-language navigation. *arXiv preprint arXiv:2203.11591*, 2022.

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

[90] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[91] Yongming Rao, Wenliang Zhao, Guangyi Chen, Yansong Tang, Zheng Zhu, Guan Huang, Jie Zhou, and Jiwen Lu. Denseclip: Language-guided dense prediction with context-aware prompting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18082–18091, 2022.

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

[93] Terry Regier. *The human semantic potential: Spatial language and constrained connectionism*. MIT Press, 1996.

[94] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[95] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[96] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[97] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.

[98] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8216–8223. IEEE, 2023.

[99] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pages 894–906. PMLR, 2022.

[100] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 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, 2020.

[101] Xiufeng Song, Xiao Guo, Jiache Zhang, Qirui Li, Lei Bai, Xiaoming Liu, Guangtao Zhai, and Xiaohong Liu. On learning multi-modal forgery representation for diffusion generated video detection. In *NeurIPS*, 2024.

[102] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on robot learning*, pages 477–490. PMLR, 2022.

[103] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

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

[105] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

[106] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. *arXiv preprint arXiv:1904.04195*, 2019.

[107] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Twenty-fifth AAAI conference on artificial intelligence*, 2011.

[108] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406. PMLR, 2020.

[109] Yao-Hung Hubert Tsai, Vansh Dhar, Jialu Li, Bowen Zhang, and Jian Zhang. Multimodal large language model for visual navigation. *arXiv preprint arXiv:2310.08669*, 2023.

[110] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[111] Hanqing Wang, Wenguan Wang, Wei Liang, Caiming Xiong, and Jianbing Shen. Structured scene memory for vision-language navigation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 8455–8464, 2021.

[112] Su Wang, Ceslee Montgomery, Jordi Orbay, Vighnesh Birodkar, Aleksandra Faust, Izzeddin Gur, Natasha Jaques, Austin Waters, Jason Baldridge, and Peter Anderson. Less is more: Generating grounded navigation instructions from landmarks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15428–15438, 2022.

[113] Xiaohan Wang, Wenguan Wang, Jiayi Shao, and Yi Yang. Lana: A language-capable navigator for instruction following and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19048–19058, 2023.

[114] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019.

[115] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, et al. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022.

[116] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *ArXiv*, abs/2302.01560, 2023.

[117] Zun Wang, Jialu Li, Yicong Hong, Songze Li, Kunchang Li, Shoubin Yu, Yi Wang, Yu Qiao, Yali Wang, Mohit Bansal, et al. Bootstrapping language-guided navigation learning with self-refining data flywheel. *arXiv preprint arXiv:2412.08467*, 2024.

[118] Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12009–12020, 2023.

[119] Zun Wang, Jialu Li, Han Lin, Jaehong Yoon, and Mohit Bansal. Dreamrunner: Fine-grained storytelling video generation with retrieval-augmented motion adaptation. *arXiv preprint arXiv:2411.16657*, 2024.

[120] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

[121] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7515–7525, 2021.

[122] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.

[123] Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. Filip: Fine-grained interactive language-image pre-training. *arXiv preprint arXiv:2111.07783*, 2021.

[124] Haonan Yu, Xiaochen Lian, Haichao Zhang, and Wei Xu. Guided feature transformation (gft): A neural language grounding module for embodied agents. *arXiv preprint arXiv:1805.08329*, 2018.

[125] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

[126] Shoubin Yu, Jacob Zhiyuan Fang, Jian Zheng, Gunnar Sigurdsson, Vicente Ordonez, Robinson Piramuthu, and Mohit Bansal. Zero-shot controllable image-to-video animation via motion decomposition. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3332–3341, 2024.

[127] Shoubin Yu, Difan Liu, Ziqiao Ma, Yicong Hong, Yang Zhou, Hao Tan, Joyce Chai, and Mohit Bansal. Veggie: Instructional editing and reasoning video concepts with grounded generation. *arXiv preprint arXiv:2503.14350*, 2025.

[128] Abhay Zala, Jaemin Cho, Han Lin, Jaehong Yoon, and Mohit Bansal. Envgen: Generating and adapting environments via llms for training embodied agents. *COLM*, 2024.

[129] Eric Zelikman, Qian Huang, Gabriel Poesia, Noah Goodman, and Nick Haber. Parsel: Algorithmic reasoning with language models by composing decompositions. *Advances in Neural Information Processing Systems*, 36:31466–31523, 2023.

[130] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[131] Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *arXiv preprint arXiv:2111.03930*, 2021.

[132] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models, 2022. *URL https://arxiv. org/abs/2205.01068*, 3:19–0, 2023.

[133] Yue Zhang, Ben Colman, Xiao Guo, Ali Shahriyari, and Gaurav Bharaj. Common sense reasoning for deepfake detection. In *European Conference on Computer Vision*, pages 399–415. Springer, 2024.

[134] Yue Zhang, Quan Guo, and Parisa Kordjamshidi. Towards navigation by reasoning over spatial configurations. In *Proceedings of Second International Combined Workshop on Spatial Language Understanding and Grounded Communication for Robotics*, pages 42–52, 2021.

[135] Yue Zhang, Quan Guo, and Parisa Kordjamshidi. Navhint: Vision and language navigation agent with a hint generator. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 92–103, 2024.

[136] Yue Zhang and Parisa Kordjamshidi. Explicit object relation alignment for vision and language navigation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 322–331, 2022.

[137] Yue Zhang and Parisa Kordjamshidi. Lovis: Learning orientation and visual signals for vision and language navigation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5745–5754, 2022.

[138] Yue Zhang and Parisa Kordjamshidi. Vln-trans: Translator for the vision and language navigation agent. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[139] Yue Zhang and Parisa Kordjamshidi. Narrowing the gap between vision and action in navigation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 856–865, 2024.

[140] Yue Zhang, Ziqiao Ma, Jialu Li, Yanyuan Qiao, Zun Wang, Joyce Chai, Qi Wu, Mohit Bansal, and Parisa Kordjamshidi. Vision-and-language navigation today and tomorrow: A survey in the era of foundation models. *Transactions on Machine Learning Research*.

[141] Yue Zhang, Zhiyang Xu, Ying Shen, Parisa Kordjamshidi, and Lifu Huang. Spartun3d: Situated spatial understanding of 3d world in large language models. *arXiv preprint arXiv:2410.03878*, 2024.

[142] Ming Zhao, Peter Anderson, Vihan Jain, Su Wang, Alexander Ku, Jason Baldridge, and Eugene Ie. On the evaluation of vision-and-language navigation instructions. *arXiv preprint arXiv:2101.10504*, 2021.

[143] Chen Zheng and Parisa Kordjamshidi. Srlgrn: Semantic role labeling graph reasoning network. *arXiv preprint arXiv:2010.03604*, 2020.

[144] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[145] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. *arXiv preprint arXiv:2305.16986*, 2023.

[146] Gengze Zhou, Yicong Hong, and Qi Wu. 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, 2024.

[147] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16816–16825, 2022.

[148] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[149] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[150] Fengda Zhu, Xiwen Liang, Yi Zhu, Qizhi Yu, Xiaojun Chang, and Xiaodan Liang. Soon: Scenario oriented object navigation with graph-based exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12689–12699, 2021.

[151] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10012–10022, 2020.

[152] Wang Zhu, Hexiang Hu, Jiacheng Chen, Zhiwei Deng, Vihan Jain, Eugene Ie, and Fei Sha. Babywalk: Going farther in vision-and-language navigation by taking baby steps. *arXiv preprint arXiv:2005.04625*, 2020.

[153] Ziyu Zhu, Xiaojian Ma, Yixin Chen, Zhidong Deng, Siyuan Huang, and Qing Li. 3d-vista: Pre-trained transformer for 3d vision and text alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2911–2921, 2023.