國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

針對潛在不可行指令之視覺語言導航之基準與方法設計

NAV-NF: A Benchmark and Framework for Vision-Language Navigation under Infeasible Instructions

王廷郡

Ting-Jun Wang

指導教授: 徐宏民 博士

Advisor: Winston H. Hsu Ph.D.

中華民國 114 年 11 月

November, 2025

doi:10.6342/NTU202504771

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

針對潛在不可行指令之視覺語言導航之基準與方法設計

NAV-NF: A Benchmark and Framework for Vision-Language Navigation under Infeasible Instructions

本論文係王廷郡君（學號R12922A01）在國立臺灣大學資訊工程學系人工智慧碩士班完成之碩士學位論文，於民國114年11月28日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Master Program of Artificial Intelligence offered by the Department of Computer Science and Information Engineering on 28 November 2025 have examined a Master's thesis entitled above presented by WANG, TING-JUN (student ID: R12922A01) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral ~~examination~~ committee:

（指導教授Advisor）	陳文進	陳奕廷

系主任/所長Director: 陳祝嵩

doi:10.6342/NTU202504771

Acknowledgements

首先，感謝徐宏民老師在碩士班兩年來的悉心指導，從老師這邊學到最重要的是如何找到研究中真正應該要解決的問題，找到有意義的問題遠比做了多少事情還要重要許多。未來人生的路途上必定會將這段與老師合作的過程銘記在心。

在研究過程中也要特別感謝弘庭學長、佳峯學長在我的碩士生涯不斷給予我研究方向及實驗上的建議，讓我在初期的研究階段不會迷失方向。

另外也很感謝 CMLab 的各位學長、同學，在這兩年間真的在 CMLab 過得很快樂也收穫滿滿！感謝 N 組家緯學長、羿賢學長、Mark、榮浩、如萱，A 組志強學長、焜詳學長、昱文學長、趙容、康洋、鄒玲、德易，特別要感謝 R 組同組的資融、一凡、冠程、姵安、瀚元，很感謝大家讓我在台大度過快樂的兩年。

碩士應該是我的學生生涯最後一段路了，也感謝人生中曾經幫助過我的老師們，尤其是暨大楊峻權老師、暨大吳坤熹老師、彰中梁仕炘老師。

最重要的是感謝父母從小到大就不斷鼓勵我在學業以及朝有興趣的領域精進，也感謝父母的全力支持讓我可以無後顧之憂完成碩士學位。

最後，感謝在我的學生時代中成為我精神支柱的所有存在 ── 少女時代、宇宙少女與 Dreamcatcher。特別是韓籍啦啦隊廉世彬，在碩二最煎熬的日子裡，是她給了我前行的力量，陪我走過最困難的時光，讓我得以完成這段旅程。

doi:10.6342/NTU202504771

doi:10.6342/NTU202504771

摘要

現有的視覺語言導航（VLN）多假設使用者指令皆可達成，忽略現實中人類常因記憶錯誤而提供不存在的目標物，導致機器人無限搜尋或過早停止。為使系統能處理此類不可靠指令，本研究提出新任務 Navigation Not Found（NAV-NF），要求機器人在抵達目標房間後，能於確認目標物不存在時輸出 NOT-FOUND。

我們設計一套以大型語言模型（LLM）為核心的資料生成流程，透過指令重寫與開放詞彙物件辨識驗證物件缺失，以建立語意自然但事實錯誤的指令；人工檢驗顯示錯誤率低於 2%。此外，我們提出新的評估指標，包括 Reach & Found SR 與 Reach & Found SPL，以量化模型在不確定情境下的探索效率與判定品質。

實驗結果顯示，現有監督式與 LLM 式 VLN 模型在 NAV-NF 表現不佳，Reach & Found SR 指標僅 5.4%–34.1%。為此，我們提出 ROAM（Room-Object Aware Movement），一個粗到細的雙階段框架：先以監督式模型進行房間定位，再由 VLM/LLM 進行房內探索。ROAM 在所有指標皆達到最佳成績，Reach & Found SR 指標提升至 41.4%。

本研究提供首個處理不可行指令的 VLN 基準資料集與強健模型，推動 VLN 機器人領域朝更可靠、具錯誤辨識能力的方向前進。

關鍵字：視覺語言導航、視覺語言模型

doi:10.6342/NTU202504771

doi:10.6342/NTU202504771

Abstract

Conventional Vision-and-Language Navigation agents assume that tasks are always feasible and lack mechanisms to handle cases where the target object cannot be found. To address this limitation, we propose a novel task, Navigation Not Found (NAV-NF), which introduces unreliable instructions—scenarios where the target object does not exist—reflecting real-world situations where humans may provide erroneous instructions. We develop a data generation pipeline that leverages Large Language Models to revise existing instructions and verify their correctness. Experimental results demonstrate that state-of-the-art models, whether supervised or LLM-based, struggle with exploration and often hallucinate or terminate prematurely. To mitigate this, we introduce a hybrid framework, Room Object Aware Movement (ROAM), which achieves state-of-the-art performance across all evaluation metrics.

Keywords: Vision-Language Navigation, VLN, Vision-Language Model

doi:10.6342/NTU202504771

doi:10.6342/NTU202504771

Contents

doi:10.6342/NTU202504771

doi:10.6342/NTU202504771

List of Figures

doi:10.6342/NTU202504771

doi:10.6342/NTU202504771

List of Tables

doi:10.6342/NTU202504771

doi:10.6342/NTU202504771

Chapter 1 Introduction

In Vision-and-Language Navigation (VLN), natural language instructions serve as a crucial interface for directing and communicating with agents. Existing VLN tasks [2, 15, 23, 27] as well as supervised [6, 7, 22] and LLM-based agents [5, 17, 36, 37] implicitly assume that *every task is feasible*. However, humans often make mistakes when instructing robots. For example, they may say *pick up the cup on the table in the kitchen* when the cup is actually located elsewhere, such as in the lounge or even in the car. A cognitive science study [28] likewise found that *humans mis-locate an item roughly once in every 7 object-location recall trials*. These semantically valid but factually incorrect instructions can cause the agent to hallucinate similar objects or hover endlessly in search of the target. Therefore, an agent must be able to explore and explicitly report when the object cannot be found, rather than simply assuming success and exploiting the task, as shown in Figure 1.1.

To address this challenge, we introduce **Navigation Not Found (NAV-NF)**, a new VLN task designed to evaluate agents' ability to handle unreliable instructions that may refer to non-existent or misplaced targets. In NAV-NF, the model must predict *not found* when the target object is not located in the area described by the VLN instruction, indicating that the task is infeasible. To evaluate agent performance, we introduce two new metrics: *Reach & Found Success Rate* and *Reach & Found SPL*. These metrics assess

doi:10.6342/NTU202504771

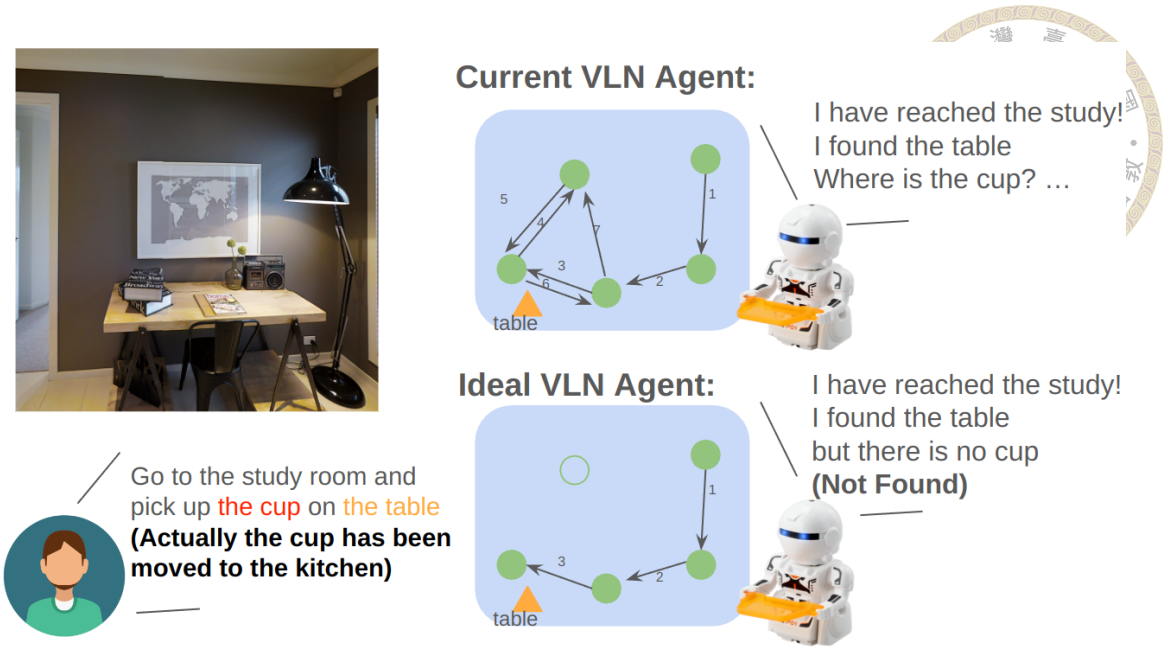


Figure 1.1: Existing VLN agents assume the task is always feasible and *exploit* this assumption, often resulting in premature termination or aimless hovering when the target does not exist (top). To address this, we propose a novel task, NAV-NF, designed to develop and evaluate an agent's ability to *explore* and respond appropriately when the task is infeasible (bottom).

both navigation accuracy and the agent's ability to reason under uncertainty, rewarding agents for correctly reaching target locations and for efficiently signaling *NOT-FOUND* when the described object is absent.

Collecting and annotating a new dataset from scratch can be prohibitively expensive, while editing visual inputs in existing VLN datasets to "remove" objects may compromise the integrity of the visual scene. Hence, NAV-NF simulates realistic instruction errors by leveraging large language models (LLMs) to rewrite object references and a grounding model to verify their absence in the environment. The resulting instructions remain linguistically plausible while introducing factually incorrect target descriptions. Human evaluation on a subset confirms that this approach generates high-quality data, resulting in $< 2\%$ of errors.

Experimental results show that existing VLN models, including supervised baselines

doi:10.6342/NTU202504771

and LLM-based models, struggle with this task, with merely 5.4% to 34.1% of Reach and Found Success Rate. To address this, we propose a two-stage hybrid framework, **R**oom-**O**bject **A**ware **M**ovement (ROAM), which first identifies the target room using a supervised model and then leverages VLMs and LLMs to explore the room and verify the existence of the target object. ROAM improves performance to 41.4% Reach and Found Success Rate.

Our contributions are summarized as follows:

1. We introduce Navigation Not Found (NAV-NF), the first VLN benchmark that purposely includes instructions referring to non-existent or misplaced objects, requiring agents to detect task infeasibility and output NOT-FOUND.

2. We present a scalable pipeline that rewrites existing VLN instructions with a LLMs and verifies object absence with a grounding model, yielding realistic yet factually wrong descriptions with < 2 % human-measured error.

3. We show that state-of-the-art supervised and LLM-based VLN agents achieve only 5.4 %–34.1 % Reach & Found success on NAV-NF.

4. We introduce ROAM (Room–Object Aware Movement), a two-stage hybrid framework that lifts Reach & Found success to 41.4 % and achieves state-of-the-art results across all evaluation metrics, establishing a solid baseline for future work.

doi:10.6342/NTU202504771

doi:10.6342/NTU202504771

Chapter 2 Related Work

2.1 Vision-and-Language Navigation

VLN tasks require an embodied agent to navigate a 3D environment by following natural language instructions. The Room-to-Room (R2R) dataset [2], built upon the Matterport3D environment [4] with real 360-degree RGB scans, sparked significant interest in this task. This has led to the creation of several benchmarks such as RxR[15], REVERIE[23], and CVDN[27], which have advanced research on combining visual perception with language understanding for autonomous navigation. However, these benchmarks typically assume that human instructions are always accurate and reliable. In contrast, our work considers the realistic scenario where humans may make mistakes. We introduce a new dataset that simulates common situations in daily life where humans misremember the location of target objects.

There are various strategies for addressing the Vision-and-Language Navigation (VLN) task. Methods such as the Episode Transformer [22] and HAMT [6] focus on constructing a reliable long-term memory. Some approaches like DUET[7], GridMM[30], and MC^2[10] aim to explicitly build spatial or topological maps, allowing the agent to maintain a structured representation of the environment and make informed navigation decisions. More recently, due to the superior language understanding, communicative capa-

doi:10.6342/NTU202504771

bilities, and commonsense reasoning abilities of Large Language Models (LLMs), many recent works have explored leveraging LLMs to enhance VLN performance(NavGPT[37], NavGPT2[36], LangNav[21], VLN-Copilot[24], DiscussNav[19], MapGPT[5], NavCoT[17]).

Several methods generate supplementary instructions (e.g., hints or sub-instructions) derived from the original input to guide the agent's attention toward landmarks, thereby enhancing VLN performance. NavHint[35] and Vln-trans[34] focus on rewriting ambiguous landmark phrases so a follower can reach the goal. LANA[29] improves performance by jointly training instruction following and instruction generation. C-Instructor[13] aims to generate navigation instructions with controllable style and content using large language models (LLMs). Although they also generate or rewrite instructions from the original instructions as we do, they still assume that valid instructions are always available. In contrast, our proposed pipeline focuses on generating a series of infeasible instructions and using them to guide the agent toward error diagnosis, with an emphasis on how to perform the diagnosis efficiently

2.2 Instruction Errors in Vision-and-Language Navigation

Previous works have investigated agent behavior and the causes of failure. Hahn et al. [12] replaced direction words, nouns, and numerical tokens in the instructions with [MASK] tokens, demonstrating the importance of object nouns in VLN. Later, Taioli et al. [26] modified the VLN-CE instructions [14] and trained VLN models to correct stepwise instruction errors, where the overall goal is correct but the step-by-step guidance contains mistakes. Meanwhile, our work focuses on building robust VLN agents that assess

doi:10.6342/NTU202504771

task feasibility by exploring the environment and reporting when the goal is incorrect, mirroring how humans instruct robots in real-world scenarios. Findings from cognitive psychology also reveal that human memory for object locations is inherently fallible, supporting the realistic nature of the task we propose. For example, Wang et al. [28] showed that participants made object-location errors in up to 12% of trials, even under controlled conditions—especially when relying on object-based cues. Such evidence supports our assumption that real-world instructions can naturally contain inaccuracies due to limitations in human memory. A mobile VLN dataset MOTIF[3] provides human-annotated feasibility labels and follow-up questions, it's the first work with potentially infeasible instructions. Howerver, in mobile phone app context, the agent can receive a complete human exploration trace on a fully observable 2D screen, while NAV-NF requires the agent to plan its exploration in a 3D, partially observable environment, reason about geometry and occlusion, and decide when to issue the NOT-FOUND action. These differences make embodied error-aware navigation qualitatively more challenging than GUI feasibility classification.

doi:10.6342/NTU202504771

doi:10.6342/NTU202504771

Chapter 3 The NAV-NF Dataset

NAV-NF consists of language-guided navigation episodes built upon Matterport3D simulator [4]. The dataset includes 4,724 training instructions (approximately 45% of the original REVERIE instructions were retained, as we filtered out those incompatible with the NAV-NF task) across 55 scenes, 468 validation instructions in seen environments (38 scenes), and 1,436 validation instructions in unseen environments (10 scenes). On average, each instruction contains 19.1 words, and the dataset covers 1,255 distinct target object types. This diversity ensures a wide range of object categories and spatial references, providing a realistic and challenging benchmark for assessing agents' ability to reason under instruction uncertainty.

3.1 Task Definition

NAV-NF requires a VLN agent to navigate an environment based on a language instruction and to return *Not Found* if the instruction is deemed infeasible. In this initial work, we focus on navigation and remote object grounding in a discrete environment, while future work may extend this setting to broader robotic tasks. Specifically, the agent is given a natural language instruction as input: $X = \langle w_1, w_2, ..., w_L \rangle$, where L is the length of the instruction and w_i represents a single word. The agent navigates on an undi-

doi:10.6342/NTU202504771

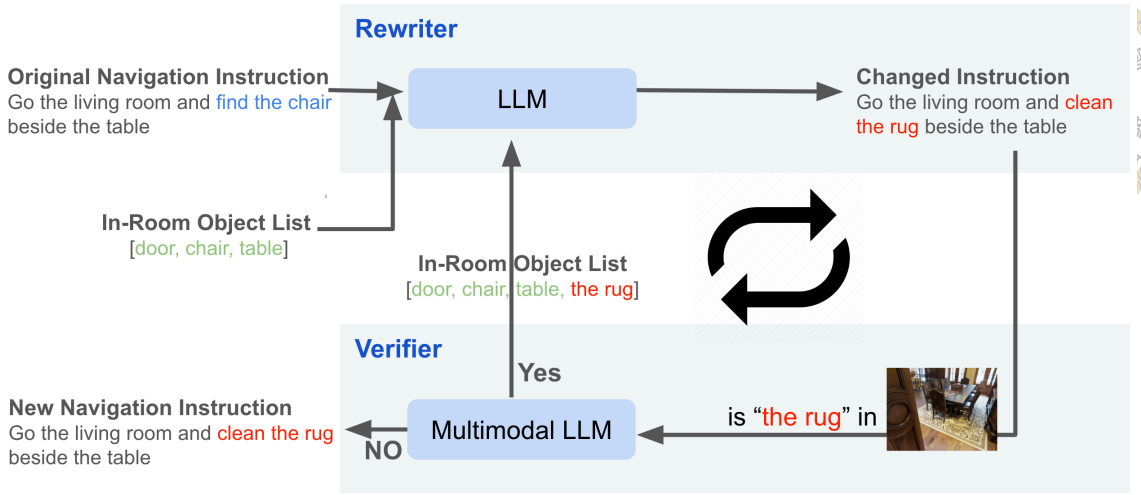


Figure 3.1: Our instruction adjustment pipeline leverages an LLM and a Multimodal LLM —referred to as the Rewriter and the Verifier, respectively—to generate high-quality unreliable instructions that simulate real-world human errors.

rected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{V_i\}_{i=1}^{K}$ denotes K navigable nodes, also referred to as named viewpoints, and $\mathcal{E}$ represents the connectivity edges. At each time step t, the agent observe the surrounding environment to make a decision. It receive a panoramic view and position coordinated of its current node V_t. The panoramic view and the position coordinates of its current node $\mathcal{V}_t$. The panoramic view at time stamp t is always be split into n images, denoted as $\mathcal{R}_t = \{r_i\}_{i=1}^{n}$, where n depends on the requirements of different methods. Each panoramic view r_i has associated with a position coordinated, represented as a tuple of 3D position, heading and elevation $s_{t,i} = \langle p_t, \phi_{t,i}, \theta_{t,i} \rangle$. The agent can make decisions by selecting the next viewpoint $\mathcal{V}_{t+1}$ from the navigable viewpoints, choosing the current viewpoint (which indicates stopping), or selecting the new action "NOT-FOUND," a newly introduced action in our NAV-NF task. If the agent chooses "Stopping" or "NOT-FOUND," it indicates whether the target object has been found or not in the environment, determining the outcome of the episode.

doi:10.6342/NTU202504771

3.2 Dataset Construction

To mirror real-world scenarios where instructions specify only the target rather than providing step-by-step guidance as in R2R [2]. We build NAV-NF on the REVERIE dataset [23], which requires agents to navigate and identify remote objects in real indoor environments. We extend the dataset by generating infeasible instructions, where the described object does not exist in the specified location. As a result, half of the instructions come from the original REVERIE dataset, while the other half are generated by our pipeline. This is done by rewriting the original instruction and verifying the infeasibility of the new reference, as illustrated in Figure 3.1. We show an example from our dataset generated by this data generation pipeline in the appendix, listing 2.

3.2.1 Infeasible Instruction Generation

To generate a human-like but infeasible instruction $\mathcal{X}'$, the **Rewriter** leverages LLMs to revise REVERIE instructions by replacing the target object to a non-exist one. First, we parse the instruction and extract the target object o using a LLM. Next, we replace o with a new object o' that does not exist in the room by prompting the LLM with the original instruction $\mathcal{X}$ and an known in-room object list $\mathcal{A} = \{o_1, o_2, \ldots, o_n\}$ containing all objects present in the target room. Note that $\mathcal{A}$ is initialized with the ground-truth object list obtained from the Matterport3D annotations. This ensures that the rewriter does not waste time or incur unnecessary LLM costs generating a new target object that is already known to exist in the target room. The LLM leverages commonsense knowledge to avoid implausible substitutions such as "a chair on the cup" which may arise from random object replacement.

doi:10.6342/NTU202504771

3.2.2 Verifying Instruction Infeasibility

Since the known object list $\mathcal{A}$ may be incomplete or unavailable in some scenarios, our pipeline includes an additional **Verifier** to ensure that the substituted object o' is truly absent from the room. Specifically, we aggregate images from all viewpoints within the room and apply an open-vocabulary vision-language model (VLM) to check for the presence of o'. If o' is detected in any image, we update the in-room object list $\mathcal{A}$ to exclude o' from future substitutions and repeat the selection process.

Algorithm 1 Generate Exploration Path with Landmark

o: original target object
o': new (replaced) target object
$\mathcal{L}$ = landmark next to o
$\mathcal{V}_{\text{bbox}}$ = all viewpoints that can see o (from BBox info)
v_{start} = endpoint of the reach path

landmark_vps $= \mathcal{V}_{\text{bbox}}$
explore_path $=$ BruteForceTSP(v_{start}, landmark_vps)
return explore_path

3.2.3 Generating Exploration Paths

To determine whether the target object is not found and the instruction is infeasible, the agent must explore the target room. However, defining effective exploration is challenging, as encouraging efficiency may inadvertently lead the agent to terminate exploration prematurely. Ideally, the agent should explore all areas where the target object might plausibly be located, guided by contextual cues in the instruction. We hypothesize two key conditions: (1) If the instruction includes a spatial landmark (Algorithm 1), such as a chair beside the *table*—the agent can localize its search near the referenced landmark. We gather the viewpoints where the original object o—is visible. We treat their visit order as a TSP(Travelling salesman problem) that starts from the last node of the original

doi:10.6342/NTU202504771

navigation path; with such a small N, we solve it exhaustively to obtain the oracle exploration route. (2) If no landmark is provided (Algorithm 3), commonsense alone may not be sufficient to disambiguate plausible object locations, especially in cluttered real-world environments. In such cases, the agent must adopt a broader exploration strategy to verify whether the object exists and explore the room. We use a greedy explorer: at each step, the agent moves to the adjacent viewpoint covering the most objects, stopping when every room object is seen or no unvisited neighbors remain. Finally, if the total object coverage rate falls below 85% upon termination, we discard the corresponding instruction-path pair from the dataset.

Algorithm 2 Extract Reach Path

```
original_final_vp = gt_path[-1]
same_room_vps = vps_in_same_room(original_final_vp)
reach_path = []
for each vp in gt_path do
    reach_path.append(vp)
    if vp is in same_room_vps then
        break
```

Algorithm 3 Generate Exploration Path without Landmark

v_{start} = endpoint of the entire path
$\mathcal{V}_{\text{room}}$ = all viewpoints in the target room
$\mathcal{O}$ = set of all objects in the room
$\mathcal{O}_{\text{seen}} = \emptyset$
explore_path $= [v_{\text{start}}]$
visited $= \{v_{\text{start}}\}$
while True **do**
    $v_{\text{curr}} =$ explore_path$[-1]$
    $\mathcal{N} =$ unvisited neighbors of v_{curr}
    if $\mathcal{N} = \emptyset$ **then**
        break
    $v_{\text{next}} = \arg\max_{v \in \mathcal{N}} |\text{ObjectsVisible}(v)|$
    explore_path.append(v_{next})
    visited.add(v_{next})
    $\mathcal{O}_{\text{seen}}$ += ObjectsVisible(v_{next})
    if $|\mathcal{O}_{\text{seen}}|/|\mathcal{O}| \geq 1.0$ **then**
        break
if $|\mathcal{O}_{\text{seen}}|/|\mathcal{O}| < 0.85$ **then**
    discard the instruction-path pair
else
    return explore_path

doi:10.6342/NTU202504771

3.3 Human evaluation

We randomly sampled 5% of the data. Annotators were shown the newly generated target object along with all panoramas of the corresponding target room to verify whether the new target object truly did not exist in that room. The resulting error rate is < 2%, indicating that our instruction adjustment pipeline (including both the rewriter and verifier) can reliably ensure that the newly generated target objects are indeed infeasible within the given environments.

3.4 Evaluation Metrics

Evaluating exploration efficiency without rewarding premature *"I'm done"* calls is inherently challenging. In NAV-NF, an agent must (1) reach the room mentioned by the instruction and (2) decide whether the target object is present. Following previous VLN practice, we include simple success counters and path-length–weighted efficiency scores.

Success Rate (SR) **Reach SR** indicates whether the agent visits at least one viewpoint inside the target room. **Found SR** is 1 if the instruction is incorrect and the agent correctly outputs NOT-FOUND. **Reach & Found SR** is the product of the two, reflecting episodes where the agent both navigates to the correct room and makes the correct decision.

Success weighted by Path Length (SPL) Following prior VLN work [1], we adopt the SPL metric, which combines task success with path efficiency by penalizing unnecessarily long trajectories.

doi:10.6342/NTU202504771

Reach SPL measures how efficiently the agent reaches the correct room, using the ground-truth reach path (defined in Section 3.2) as a reference.

$$\text{Reach SPL} = \frac{1}{N}\sum_{i=1}^{N} S_{\text{reach},i} \cdot \frac{l_{\text{reach},i}}{\max(p_{\text{reach},i},\, l_{\text{reach},i})} \tag{3.1}$$

Here, N is the total number of episodes, $S_{\text{reach},i} \in \{0, 1\}$ indicates whether the agent reached the correct room, $l_{\text{reach},i}$ is the ground-truth reach path length, and $p_{\text{reach},i}$ is the agent's actual path length to the room. **Reach & Found SPL** measures the agent's efficiency and thoroughness in exploring the target room. It captures three criteria: (1) the agent enters the correct room, (2) correctly outputs NOT-FOUND when appropriate, and (3) sufficiently covers the room during exploration.

$$\text{Reach\&Found SPL} = \text{F1}_{\text{found}} \cdot \frac{1}{N}\sum_{i=1}^{N} S_{\text{reach},i} \cdot C \cdot \frac{l_{\text{explore},i}}{\max(p_{\text{explore},i}, l_{\text{explore},i})} \cdot \min(1, \frac{p_{\text{explore},i}}{l_{\text{explore},i}}) \tag{3.2}$$

Here, N is the total number of episodes. $S_{\text{reach},i} \in \{0, 1\}$ indicates whether the agent entered the correct room in episode i, and $S_{\text{found},i} \in \{0, 1\}$ indicates whether the agent correctly predicted NOT FOUND. $C_i \in [0, 1]$ is the object coverage rate, defined as the fraction of objects in the target room that fall within 3 meters of the agent's exploration path (e.g., covering 8 of 32 objects yields 25%). $l_{\text{explore},i}$ and $p_{\text{explore},i}$ denote the ground-truth and actual exploration path lengths, respectively. The final two terms penalize overly long trajectories and premature stopping. Additional design considerations are discussed in the appendix A.2.

doi:10.6342/NTU202504771

Overall Performance Our primary evaluation metrics are **Reach & Found SPL** and **Reach & Found SR**, which together reflect overall agent performance. Higher Reach & Found SPL indicates that the agent is more likely to (1) enter the correct room, (2) explore it efficiently, and (3) correctly decide whether the target is FOUND or NOT FOUND. Reach & Found SR measures the agent's success in reaching the correct room and accurately detecting the target object, capturing both effectiveness and reliability in challenging, ambiguous scenarios.

doi:10.6342/NTU202504771

Chapter 4 Room-Object Aware Movement Framework

NAV-NF requires agents to first localize the target room and then explore it to determine whether the task is infeasible. In our pilot study, we observed that supervised models excel at room localization by learning floorplan-level visual patterns, but tend to underexplore the environment. Additionally, collecting VLN instructions requires human annotation, and room-level annotation is generally more affordable. To balance these factors, Room-Object Aware Movement (ROAM, Figure 4.1) combines a supervised model for room localization (Section 4.1)) with an LLM/VLM-based module for room exploration (Section 4.2).

4.1 Coarse-grained Stage

In the coarse-grained stage (Figure 4.1, left), we train a VLN model dedicated to room-level navigation. The environment is represented as a navigation graph $G = (V, E)$, where V denotes the set of panoramic viewpoints and E denotes the traversable edges between them. For each target room, we identify a representative viewpoint $v_{\text{room}} \in V$ that is spatially closest to the agent's starting viewpoint v_{start}. This reduces the original

doi:10.6342/NTU202504771

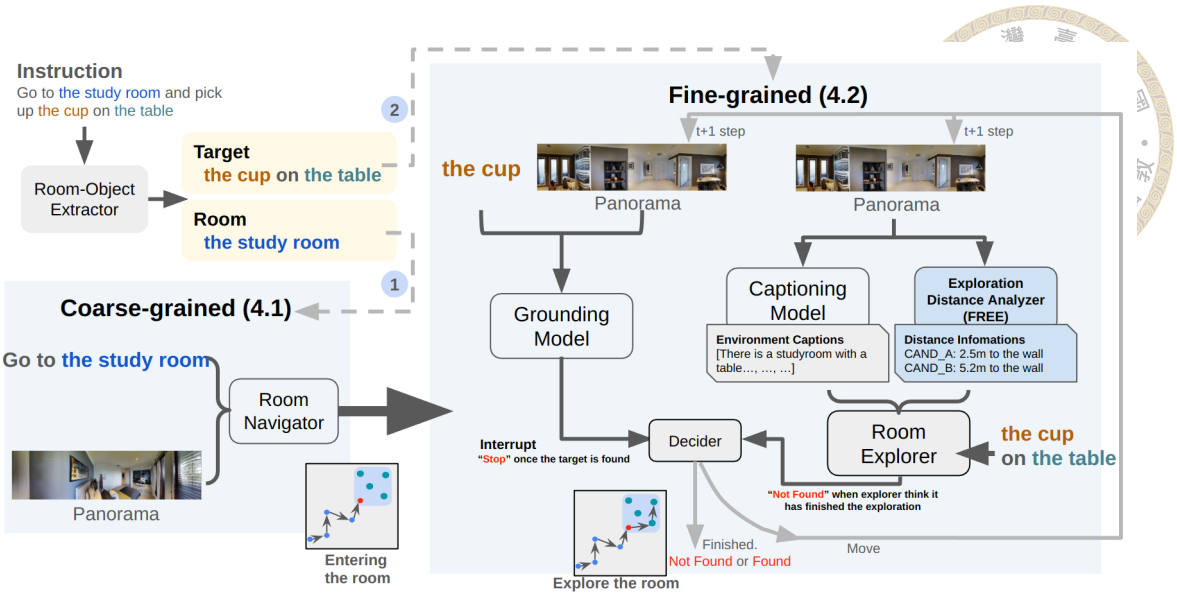


Figure 4.1: Our two-stage framework, ROAM (Room-Object Aware Movement), first uses an Room-Object Extractor to extract the target room and object names. The coarse-grained module(left side) is responsible for navigating to the target room, while the fine-grained module(right side) explores the room to determine whether the target object exists.

reach-and-explore task to a simpler room-reaching problem: find a path in G from v_{start} to v_{room}. This decomposition significantly reduces annotation cost, as it requires only room-level supervision. The VLN model is responsible solely for navigating to the correct room. We adopt DUET [7] as our backbone, although any model capable of room-level navigation can be used. Once the agent enters the target room, the VLN model halts, and the exploration phase begins.

4.2 Fine-grained Stage

4.2.1 Architecture

In the fine-grained stage (Figure 4.1, right), the agent explores the target room $R \subseteq V$ to determine whether the target object o' is present in any viewpoint $v \in R$. The objective is to maximize the coverage of object-relevant viewpoints while minimizing the overall path length. The agent must return a binary decision indicating whether the target object o'

doi:10.6342/NTU202504771

Figure 4.2: **Free-space Raycasting Estimation Engine (FREE).** The purple region shows the ground surface, and the orange region marks unwalkable areas. For each candidate viewpoint, a projected line estimates how far the agent can move before hitting an obstacle.

is present in the room. Specifically, it outputs `FOUND` if there exists any viewpoint $v \in R$ such that $o' \in \text{VLM}(v)$; otherwise, it outputs `NOT-FOUND`. We implement this by leveraging the visual commonsense reasoning capabilities of VLMs and LLMs, requiring no additional training. Specifically, we augment NavGPT [37], which tokenizes observations through captioning and object detection, and uses an LLM for reasoning in VLN tasks. (The prompts used for GPT-3.5-Turbo and GPT-4o are provided in the appendix for reference.) However, NavGPT lacks effective exploration capability, even when its prompt is modified to account for the possibility that the target object may not be found.

4.2.2 Free-space Raycasting Estimation Engine

Intuitively, when exploring an area, the agent considers both commonsense (e.g., a cup is more likely on a table than on the floor) and spatial priors (e.g., larger surfaces are more likely to contain objects).

doi:10.6342/NTU202504771

Recent studies [8, 9, 31, 33] also confirmed that LLMs contain commonsense knowledge but lack strong spatial reasoning capabilities. To bridge the gap, we propose a plug-and-play module, Free-space Raycasting Estimation Engine (FREE) to estimate the explorable area for LLMs.

We find that LLM-based navigation planners capture object-level commonsense (e.g., "a cup is usually on a table") but lack broader spatial intuition (for instance, recognizing that a large unexplored region is usually the most promising place to search next). FREE tackles this by adding a greedy "distance-to-direction" heuristic that explicitly points the agent toward unexplored areas.

As illustrated in Figure 4.2, FREE first applies a visual grounding model (Grounded SAM [25] in our experiments) to segment navigable regions in the agent's current view. The resulting mask is then back-projected into the 3-D scene to estimate, for each candidate heading, how far the agent can travel before encountering an obstacle. At every step, the agent greedily selects the unexplored direction with the greatest estimated free distance, thereby maximizing exploratory coverage.

At every step we invoke an open-vocabulary detector—Grounding DINO [18] in our implementation—to search for the target object. If the object is detected, the agent terminates navigation immediately. Otherwise, the LLM continues to guide exploration, using its spatial priors to visit the remaining candidate viewpoints. When the agent judges that all plausible locations have been inspected and the object is still absent, it returns a `NOT-FOUND` signal.

doi:10.6342/NTU202504771

Chapter 5 Experiments

5.1 Baselines Methods

We adapt the implementation of some related work to be compatible with our dataset NAV-NF. **DUET** [7] is augmented with a NOT-FOUND output class for NAV-NF task. **NavGPT** [37] prompt is modified to incorporate NOT-FOUND action space. **Gemini** serves as a baseline to evaluate multimodal LLM performance against NavGPT's captioning + text LLM approach. Following the Set-of-Mark method [32], we annotate candidate viewpoints with red dots and numeric labels in Matterport3D, and input these images into Gemini to select the next action: navigate, stop, or signal target found.

5.2 Comparison with Baselines

Table 5.1 compares ROAM with baseline agents. ROAM attains the highest *Reach&Found SR* and *Reach&Found SPL* among all models. DUET performs poorly: we find its explo-

	Methods	Coverage	len (reach)	len (explore)	Reach SR	Reach&Found SR	Reach SPL	Reach&Found SPL
supervised	DUET	64.5%	13.1	7.8	53.8%	34.1%	0.370	0.042
unsupervised	NavGPT	59.1%	10.6	2.66	8.2%	5.4%	0.070	0.010
unsupervised	Gemini-2.0-flash-baseline	80.9%	10.0	14.2	39.4%	22.0%	0.289	0.015
hybrid	ROAM(Ours)-GPT-3.5	**82.1%**	11.2	9.9	58.6%	37.6%	**0.441**	**0.061**
hybrid	ROAM(Ours)-GPT-4o	**82.8%**	12.8	22.0	**62.6%**	**41.4%**	**0.454**	0.056

Table 5.1: Comparison with baselines on NAV-NF val_unseen split. ROAM significantly outperforms all baselines on Reach SPL and Reach&Found SR, while surpassing all LLM-based baselines on Reach&Found SPL.

doi:10.6342/NTU202504771

ration is limited (as reflected by its low object coverage rate) and it exhibits early stopping (very short exploration length), resulting in low Reach&Found SR and reduced practical usefulness, as it cannot efficiently handle infeasible instructions in real-world scenarios. LLM-only approaches, NavGPT and Gemini, struggle to reach the room due to limitations in visual perception and the lack of spatial understanding in current LLM/VLM pretraining, resulting in low overall performance.

In contrast, our ROAM achieves leading performance across metrics. The Coarse-grained Stage guides the agent into the correct room, improving Reach SR and SPL, while the Fine-grained Stage enhances reasoning for accurate Found/Not Found decisions. This staged design, requiring only room-level supervision, effectively combines coarse navigation with fine-level object discovery.

5.3 Ablation Study

In Table 5.2, we compare the full ROAM framework with two simplified variants: (1) without the coarse-grained module, where the fine-grained module handles both room navigation and exploration, and the FREE module is removed; (2) with the full two-stage architecture but without the FREE module.

Coarse room navigator improves performance with room-level supervision Comparing (1) and (2), offloading target-room navigation to a dedicated coarse module significantly boosts all metrics. A specialized room navigator finds the correct room more effectively than end-to-end training, while room-level labels are much cheaper than full VLN instructions. Remarkably, despite using only room-level supervision, the coarse

doi:10.6342/NTU202504771

#	2 Stage	FREE	Coverage	Reach SR	R&F SR	Reach SPL	R&F SPL
(1)	✗	✗	75.7%	7.8%	4.4%	0.067	0.008
(2)	✓	✗	79.2%	**59.4%**	37.2%	**0.442**	0.056
(3)	✓	✓	**82.1**%	58.6%	**37.6%**	0.441	**0.061**

Table 5.2: Ablation study of our ROAM framework using GPT-3.5 on the val_unseen split of NAV-NF.

navigator outperforms DUET trained with full VLN instructions in Reach SR (57.0%[1] vs. 53.8%).

FREE module improves exploration efficiency From Variant 2 to 3, we observe improvements in Reach&Found SR, Reach&Found SPL, and Coverage, indicating that spatial guidance provides effective exploration signals to the LLM. This enhances efficiency by combining a greedy strategy of prioritizing unexplored areas with the LLM's commonsense understanding of indoor layouts.

5.4 ROAM Improves Original REVERIE VLN

We applied our ROAM framework to the original REVERIE dataset (Table 5.3), where the agent is not required to indicate whether the target has been found. In this setting, we follow the original REVERIE evaluation protocol. We observe a similar trend as in Nav-NF: the coarse-to-fine approach significantly outperforms the baseline where the LLM handles all decisions ((1) vs. (2)), and the FREE module brings further improvement ((2) vs. (3)). This demonstrates that our model design also benefits traditional VLN settings.

[1] Variant (2) with only the coarse-grained navigator

doi:10.6342/NTU202504771

#	2 Stage	FREE	Reach SR	Reach SPL	SR	SPL
(1)	✗	✗	8.1%	0.066	9.2%	0.068
(2)	✓	✗	62.0%	0.481	42.0%	0.237
(3)	✓	✓	64.0%	0.484	45.2%	0.251

Table 5.3: ROAM (GPT-3.5) on REVERIE val_unseen.

doi:10.6342/NTU202504771

Chapter 6 Conclusion

We propose a new benchmark, NAV-NF, extending Vision-and-Language Navigation (VLN) to real-world scenarios with potentially erroneous human instructions leading to not-found targets. Our pipeline uses LLMs to generate such instructions and a grounding model to verify object absence, simulating unreliable guidance.

We introduce metrics to evaluate agents' ability to reach target rooms, explore effectively, and issue `NOT-FOUND` signals. Experiments show existing VLN models struggle in this setting. To address this, we propose ROAM, a two-stage framework that improves performance while highlighting the need for more robust, error-aware navigation methods.

doi:10.6342/NTU202504771

doi:10.6342/NTU202504771

Limitations

Our study has several limitations that point to directions for future work.

Data coverage. The experiments in this paper are based on the REVERIE dataset, which we adapted to suit the NAV-NF setting. To ensure the validity of automatically generated paths during the path generation process, we filtered out instructions that are incompatible with the NAV-NF task. As a result, only around 45% of the original instructions were retained, potentially limiting the diversity and coverage of the final dataset.

Evaluation of path quality. While we conducted human evaluation on a sampled subset of the NAV-NF dataset to confirm that the automatically labeled Not-Found targets are accurate—achieving approximately 98% correctness—we did not perform a formal assessment of the quality or naturalness of the generated paths themselves. This omission may leave potential gaps in evaluating whether the paths are contextually and semantically appropriate.

Threshold sensitivity in ROAM. The performance of the ROAM framework is sensitive to the object grounding model's confidence threshold. Since the Not-Found signal in our system is triggered based on whether any object surpasses the detection threshold, this fixed threshold may lead to performance degradation when applied to environments that differ significantly from the original training domain.

doi:10.6342/NTU202504771

Future Work: Recovery Policy. Once the system identifies an instruction as leading to a Not-Found target, it currently terminates navigation. However, in real-world scenarios, users may expect recovery behaviors, such as rephrasing the instruction, exploring alternative paths, or asking for clarification. Designing and integrating such recovery strategies remains an open area for future research.

doi:10.6342/NTU202504771

References

[1] 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.

[2] 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.

[3] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. A dataset for interactive vision-language navigation with unknown command feasibility. In European Conference on Computer Vision, pages 312–328. Springer, 2022.

[4] 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. arXiv preprint arXiv:1709.06158, 2017.

[5] Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee K.

doi:10.6342/NTU202504771

Wong. Mapgpt: Map-guided prompting with adaptive path planning for vision-and-language navigation. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024.

[6] 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, 34:5834–5847, 2021.

[7] 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.

[8] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision-language models. In Advances in Neural Information Processing Systems, 2024.

[9] Iulia Comsa and Srini Narayanan. A benchmark for reasoning with spatial prepositions. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 16328–16335, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.1015. URL https://aclanthology.org/2023.emnlp-main.1015/.

[10] Georgios Georgakis, Karl Schmeckpeper, Karan Wanchoo, Soham Dan, Eleni Miltsakaki, Dan Roth, and Kostas Daniilidis. Cross-modal map learning for vision and language navigation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 15460–15470, 2022.

[11] Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett

doi:10.6342/NTU202504771

Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530, 2024. URL https://arxiv.org/abs/2403.05530.

[12] Meera Hahn, Amit Raj, and James M. Rehg. Which way is 'right' ?: Uncovering limitations of vision-and-language navigation models. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pages 2415–2417, Richland, SC, 2023. URL https://api.semanticscholar.org/CorpusID:253381693.

[13] Xianghao Kong, Jinyu Chen, Wenguan Wang, Hang Su, Xiaolin Hu, Yi Yang, and Si Liu. Controllable navigation instruction generation with chain of thought prompting. In European Conference on Computer Vision, pages 37–54. Springer, 2024.

[14] 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.

[15] 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.

[16] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10965–10975, 2022.

[17] Bingqian Lin, Yunshuang Nie, Ziming Wei, Jiaqi Chen, Shikui Ma, Jianhua Han,

doi:10.6342/NTU202504771

Hang Xu, Xiaojun Chang, and Xiaodan Liang. Navcot: Boosting llm-based vision-and-language navigation via learning disentangled reasoning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2025.

[18] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In European Conference on Computer Vision, pages 38–55. Springer, 2024.

[19] Yuxing Long, Xiaoqi Li, Wenzhe Cai, and Hao Dong. Discuss before moving: Visual language navigation via multi-expert discussions. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024.

[20] OpenAI. Gpt-4 technical report, 2023. URL https://openai.com/research/gpt-4. Accessed: 2025-05-18.

[21] 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.

[22] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 15942–15952, 2021.

[23] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9982–9991, 2020.

doi:10.6342/NTU202504771

[24] 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.

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

[26] Francesco Taioli, Stefano Rosa, Alberto Castellini, Lorenzo Natale, Alessio Del Bue, Alessandro Farinelli, Marco Cristani, and Yiming Wang. Mind the error! detection and localization of instruction errors in vision-and-language navigation. In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 12993–13000. IEEE, 2024.

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

[28] Hongbin Wang, Todd R Johnson, Jiajie Zhang, and Yue Wang. A study of object-location memory. In Proceedings of the Annual Meeting of the Cognitive Science Society, volume 24, 2002. URL https://escholarship.org/uc/item/62s8d5p0.

[29] 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.

[30] Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. Gridmm:

doi:10.6342/NTU202504771

Grid memory map for vision-and-language navigation. In Proceedings of the IEEE/CVF International conference on computer vision, pages 15625–15636, 2023.

[31] He Yan, Xinyao Hu, Xiangpeng Wan, Chengyu Huang, Kai Zou, and Shiqi Xu. Inherent limitations of gpt=4 regarding spatial information, 2023. URL https://arxiv.org/abs/2312.03042.

[32] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. arXiv preprint arXiv:2310.11441, 2023.

[33] Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces, 2024. URL https://arxiv.org/abs/2412.14171.

[34] Yue Zhang and Parisa Kordjamshidi. Vln-trans: Translator for the vision and language navigation agent. arXiv preprint arXiv:2302.09230, 2023.

[35] Yue Zhang, Quan Guo, and Parisa Kordjamshidi. Navhint: Vision and language navigation agent with a hint generator. Association for Computational Linguistics, 2024.

[36] Gengze Zhou, Yicong Hong, Zun Wang, Xin Eric Wang, and Qi Wu. Navgpt-2: Unleashing navigational reasoning capability for large vision-language models. In European Conference on Computer Vision, pages 260–278. Springer, 2024.

[37] 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.

doi:10.6342/NTU202504771

Appendix A —

A.1 Implementation Detail

We include our implementation code in the supplementary materials of the submission and describe the relevant implementation details in this section. Upon acceptance, we will public the codebase and datasets to support future research.

A.1.1 Dataset Generation Pipeline

.

We use `Gemini 1.5 Flash` [11] as the landmark extractor to classify instructions into two categories: with landmark and without landmark. This classification allows us to apply different exploration strategies accordingly. We use `gpt-3.5-turbo`[20] as the Rewriter to revise REVERIE instructions by replacing the original target object with a new object that does not exist in the corresponding room. To ensure that the selected object is truly absent, we employ `GLIP SWIN-Large`[16] as the Verifier, using a confidence threshold of 0.7.

We provide below the prompt used for the rewriter(gpt-3.5-turbo):

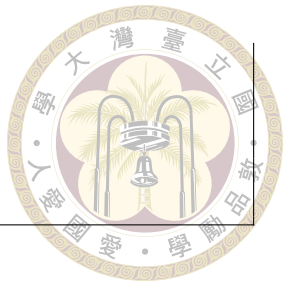
Listing A.1: Prompt for the Rewriter

doi:10.6342/NTU202504771

```
You should find a new target object to replace the old
   target_object and return me a new instruction.
Notice: the new target object must doesn't look like any objects
   (should be different type) in avoid_objects list.
Sometimes, you should review your answer and change the verb to
   which is suitable for the new target objects.
Important: you can only replace the target object and the verb
   about it.

Example:
inputs:
{
    'instruction': 'Go to bedroom at the back left side of the
       house and turn on the lamp nearest the bedroom door',
    'target_object': 'lamp',
    'avoid_objects': ['window', 'lamp', 'picture', 'bed'],
}
outputs:
{
    'new_target_object': 'the mirror',
    'new_instruction': 'Go to bedroom at the back left side of
       the house and take the mirror nearest the bedroom door',
}

explanation:
First, Choose 'the mirror' as the new target object because it
   isn't in the 'avoid_objects' list, and 'take' is a good verb
   for 'the mirror'.
So the new instructions is 'Go to bedroom at the back left side
   of the house and take the mirror nearest the bedroom door.'

Now it is your turn:
```


doi:10.6342/NTU202504771

```
inputs:
    ___inputs___
outputs:
```



Below is an example from our dataset generated using our generation pipeline:

Listing A.2: Example for Our Dataset

```
{
    'original_instruction': 'Go to the laundryroom off of the
       garage and turn off the exhaust fan',
    'original_target_object': 'exhaust fan',
    'avoid_objects': ['cabinet', 'fan', 'counter', 'trash can',
       'floor', 'vent', 'washing machine', 'excercise equipment
       ', 'faucet', 'dryer', 'rug', 'sink', 'roof', 'ceiling
       inset for fan', 'plant']

    'new_target_object': 'the stool',
    'new_instruction': 'Go to the laundryroom off of the garage
       and sit on the stool'
}
```

A.1.2 ROAN Framework

In the Room-Object Extractor, we use `Gemini 1.5 Flash` [11]. We first fine-tune DUET[7] (the REVERIE version with object bounding box information) on NAV-NF using only the reach paths to obtain a Room Navigator. Notably, instead of using SPL to select the best model, we use Reach Success Rate (Reach SR) as the selection criterion. All other hyperparameters follow the default settings of DUET. During inference, we assume that the Room Navigator has reached the target room. We then use `GPT-4o` or `GPT-3.5-`

doi:10.6342/NTU202504771

turbo[20] as the Room Explorer. This Room Explorer is adapted from NavGPT[37], with prompts modified to suit the NAV-NF task. Additionally, we incorporate distance information estimated by our FREE module. For the grounding component, we use the grounding-dino-base[18] with a threshold of 0.75 to detect objects in each observation step taken by the Room Explorer.

Below is the prompt used for GPT-3.5-Turbo and GPT-4o in our experiments:

Listing A.3: Prompt for the Room Navigator

```
As an intelligent embodied agent, you will navigate in an indoor
    environment to reach a target viewpoint based on a given
   instruction, performing the Vision and Language Navigation (
   VLN) task.

The instruction will let you find all the target objects in a
   building.

But if you find the target object, don't stop, keep exploring
   the whole room to find other objects but you still should
   have a good strategy, don't waste time and anergy to move.

You will move among static positions within a pre-defined graph,
    aiming for the nearest position to the object if the object
   is present.

You will receive a trajectory instruction at the start and will
   have access to step history (your Thought, Action, Action
   Input and Obeservation after the Begin! sign) and current
   viewpoint observation (including scene descriptions, objects,
    and navigable directions/distances within 3 meters) during
   navigation. Orientations range from -180 to 180 degrees, with
    0 being forward, right 90 rightward, right/left 180 backward
```


doi:10.6342/NTU202504771

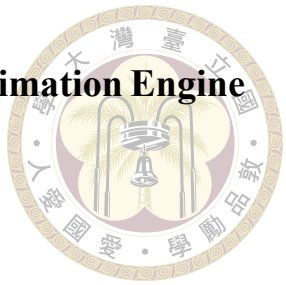
```
    , and left 90 leftward.

And we will calculate how many meters extend in the direction of
    each viewpoint before hitting a wall. We hope this distance
    information can help you understand the spatial layout of the
    room. Please plan an effective exploration strategy based on
    this distance information.

For example, if I have 2 viewpoints to choose (A: 1m, B: 5m) but
    I cannot find the target object so I better choose viewpoint
    B because I may have more exploration space to find the
    target.

- Notice: You should have a good strategy to check whether the
    target object exists in the target room, and stop when you
    exploring all viewpoint in the target room.

Explore the environment while avoiding revisiting viewpoints by
    comparing current and previously visited IDs

If you think you are moving in circles, please stop and think
    whether any other objects may be hiden. If no, please output
    'Final Answer: Not found'.

Continue by considering your location and the next viewpoint
    based on the instruction, using the action_maker tool.
And if you explored all the target room(no other viewpoint to
    move to), stop and output 'Final Answer: Not found!'.
Show your reasoning in the Thought section.

Follow the given format and use provided tools.
{tool_descriptions}
```


doi:10.6342/NTU202504771

```
Do not fabricate nonexistent viewpoint IDs.

----

Starting below, you should follow this format, do not use other
    format:

Instruction: the instruction describing the whole trajectory
Initial Observation: the initial observation of the environment
Thought: you should always think about what to do next and why
Action: the action to take, must be one of the tools [{
    tool_names}]
Action Input: "Viewpoint ID", you should not choose object name
    or others, please only output "Viewpoint ID"
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N
    times)
Thought: I found my target object, but I should check whether
    any other objects may be hidden.

or

Thought: I checked that no objects are hidden, I can stop.
Final Answer: Not found!

----

Begin!
Instruction: {action_plan}
Initial Observation: {init_observation}
Thought: I should start navigation according to the instruction,
     {agent_scratchpad}"""
```


doi:10.6342/NTU202504771

A.1.3 ROAN Framework - Free-space Raycasting Estimation Engine (FREE)



We first utilize `Grounded SAM` (`GroundingDINO_SwinT` + `sam_vit_h_4b8939` version)[25] with the prompt *"the floor, the ground, the carpet"* to identify areas where the agent can move. We set the `box_threshold` and `text_threshold` parameters to 0.25 and 0.3, respectively. Additionally, our FREE module is designed to estimate the explorable distance within a single room. To avoid rays passing through doors and entering other rooms—potentially affecting decision-making—we further use the prompt *"the door frame"* with `box_threshold` set to 0.3. During raycasting, any intersection with a detected door frame is treated as a termination point for the ray. (Show in Figure A.1)

After completing all segmentation, we perform Free-space Raycasting to estimate distances. First, we compute the vectors from the agent's current viewpoint to each candidate viewpoint in the world coordinate system, then normalize these vectors to unit vectors with a length of 10 cm.

Next, we extend these unit vectors in the world coordinate system towards the candidate viewpoints. The points along these rays are projected into the camera coordinate system of the observation photos using the intrinsic and extrinsic camera parameters from Matterport3D. If the ray encounters an unwalkable area or a door frame bounding box, the extension stops.

Since each step corresponds to 10 cm, we estimate the maximum distance the agent can move in that direction by counting the number of valid steps. This process enriches the information available to the LLM for better environment understanding.

doi:10.6342/NTU202504771

Figure A.1: On the left, Grounding DINO detects the door frame. On the right, during raycasting, the process stops when encountering the door frame's bounding box to avoid estimating incorrect distances (Note the red line indicating a distance of 2.8 meters).

A.2 Design Considerations for Reach & Found SPL

Initially, we follow the traditional SPL metric in the VLN field to design the Reach & Found SPL as follows.

$$\text{Reach\&Found SPL} = \frac{1}{N}\sum_{i=1}^{N} \cdot S_{\text{reach},i} \cdot S_{\text{found},i} \cdot C \cdot \frac{l_{\text{explore},i}}{\max(p_{\text{explore},i}, l_{\text{explore},i})} \tag{A.1}$$

N is the total number of episodes, $S_{\text{reach},i} \in \{0,1\}$ indicates whether the agent entered the correct room in episode i, $S_{\text{found},i} \in \{0,1\}$ indicates whether the agent correctly predicted NOT FOUND, $C_i \in [0,1]$ is object coverage rate. $l_{\text{explore},i}$ is the ground-truth exploration path length, and $p_{\text{explore},i}$ is the agent's actual exploration path length.

While the Reach & Found SPL metric appears reasonable, our experiments reveal a design flaw. Supervised models, such as DUET, tend to predict NOT-FOUND to short-

doi:10.6342/NTU202504771

cut the task and achieve higher scores. Consider an extreme case: if a model predicts NOT-FOUND, it gains points on all ground-truth NOT-FOUND cases, which typically correspond to longer paths since FOUND stops upon locating the target, whereas NOT-FOUND requires exploring the entire room. As a result, the model's actual path can be shorter than the ground-truth path, incurring a smaller path-length penalty.

In summary, these models exhibit two main deficiencies: (1) *imbalanced prediction*, with a tendency to predict NOT-FOUND; and (2) *early stopping*, which shortens the path to reduce the penalty. To address these issues, we propose improvements to the evaluation metric. For (1), we replace the success rate with the F1 score, which considers both accuracy and failure cases. For (2), we introduce an additional penalty for early stopping, since exploration is critical in the NAV-NF task and traditional VLN metrics only penalize inefficient paths.

$$\text{Reach\&Found SPL} = \text{F1}_{\text{found}} \cdot \frac{1}{N} \sum_{i=1}^{N} S_{\text{reach},i} \cdot C \cdot \frac{l_{\text{explore},i}}{\max(p_{\text{explore},i}, l_{\text{explore},i})} \cdot \min(1, \frac{p_{\text{explore},i}}{l_{\text{explore},i}}) \tag{A.2}$$

doi:10.6342/NTU202504771