VLN-NF: Feasibility-Aware Vision-and-Language Navigation with False-Premise Instructions

Hung-Ting Su[1*†‡] **Ting-Jun Wang**[1‡] **Jia-Fong Yeh**[1] **Min Sun**[2] **Winston H. Hsu**[1]

[1]National Taiwan University [2]National Tsing Hua University
https://vln-nf.github.io/

Abstract

Conventional Vision-and-Language Navigation (VLN) benchmarks assume instructions are feasible and the referenced target exists, leaving agents ill-equipped to handle false-premise goals. We introduce **VLN-NF**, a benchmark with false-premise instructions where the target is absent from the specified room and agents must navigate, gather evidence through in-room exploration, and explicitly output `NOT-FOUND`. VLN-NF is constructed via a scalable pipeline that rewrites VLN instructions using an LLM and verifies target absence with a VLM, producing plausible yet factually incorrect goals. We further propose **REV-SPL** to jointly evaluate room reaching, exploration coverage, and decision correctness. To address this challenge, we present **ROAM**, a two-stage hybrid that combines supervised room-level navigation with LLM/VLM-driven in-room exploration guided by a free-space clearance prior. ROAM achieves the best REV-SPL among compared methods, while baselines often under-explore and terminate prematurely under unreliable instructions.

1 Introduction

Language provides a flexible interface for specifying goals to situated agents in the physical world. Vision-and-Language Navigation (VLN) is a crucial robotic task studying language grounding and path planning under partial observability. Existing VLN datasets (Anderson et al., 2018b; Ku et al., 2020; Qi et al., 2020; Thomason et al., 2020) and supervised agents (Pashevich et al., 2021; Chen et al., 2021, 2022) or agents based on LLM built on them (Zhou et al., 2024b,a; Chen et al., 2024; Lin et al., 2025) typically assume that *every instruction is feasible*. In real deployments, however, humans often make mistakes when instructing robots. For example, a user may say "*pick up the plate on the table in the kitchen*" when the plate is actually elsewhere (e.g., in the lounge or even in the car). A cognitive science study (Wang et al., 2002) found that humans mislocate an item roughly once in every seven object-location recall trials. Such instructions are semantically well-formed but factually incorrect, and they can cause agents to hallucinate similar objects or search indefinitely. Therefore, an agent must be able to explore and explicitly report when the target cannot be found, rather than simply assuming success and exploiting task priors.

Related efforts study instruction unreliability across embodied settings, including feasibility in 2D interfaces (Burns et al., 2022), stepwise VLN instruction errors (Taioli et al., 2024), obstruction-induced instruction–environment mismatch (Hong et al., 2024), and false-premise/abstention/hallucination in embodied QA or VLA (Wu et al., 2024, 2025; Hsieh et al., 2025; Chakraborty et al., 2025). Yet *evidence-grounded* `NOT-FOUND` for *3D partially observable* VLN—where absence must be justified through autonomous exploration—is still largely unaddressed. To this end, we introduce **VLN-Not Found (VLN-NF)**, a new VLN task that evaluates agents' ability to handle false-premise instructions referring to non-existent targets. We focus on this setting as a controlled but practical form of goal-level false premise, where absence must be established through exploration under partial observability rather than inferred from a single observation. In VLN-NF, the agent must predict `NOT-FOUND` when the target object is absent from the region specified by the instruction, indicating that the instruction is infeasible. To enable low-cost construction, we develop a scalable pipeline that converts feasible VLN instances into infeasible ones via instruction rewriting and automatic verification. An LLM replaces

[*]This work was conducted at National Taiwan University. The author is currently affiliated with DRIC.
[†]Project lead.
[‡]Equal technical contribution.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 25134–25150

Figure 1: **A toy illustration of failure modes under unreliable instructions.** For compactness, we visualize two *separate single-target* queries in the same scene (cup: feasible; plate: infeasible/absent in the kitchen). **Left: Standard VLN (DUET+VLN)** lacks an explicit `NOT-FOUND` output (hence "?" for plate). **Middle: Adding `NOT-FOUND` to action space (DUET+VLN-NF)** can lead to premature abstention. **Right: Our proposed ROAM+VLN-NF** performs evidence-gathering exploration and outputs the correct decisions.

the original target with a plausible but absent alternative to simulate false-premise instructions, and a VLM verifies the target's absence. This yields linguistically natural instructions with factually incorrect target descriptions. Human evaluation on a subset confirms high quality, with <2% errors.

Unlike conventional VLN, where success is defined by reaching a single goal and efficiency is measured against a shortest path, VLN-NF requires *evidence-gathering exploration* under partial observability: the absence of a target cannot be confirmed from a single viewpoint. Naively reusing standard VLN metrics encourages degenerate behaviors (e.g., stopping early or predicting `NOT-FOUND` without sufficient search). To enable principled comparison, we refine SPL and propose **Reach–Explore–Verify SPL (REV-SPL)**. REV-SPL (1) defines a reference exploration protocol that specifies *where* an agent should reasonably search given the instruction (e.g., local exploration when landmark cues exist and a broader room sweep otherwise), and uses the resulting reference length to normalize exploration efficiency; (2) evaluates decision quality on mixed feasible/infeasible episodes, capturing the trade-off between abstaining when appropriate and avoiding false alarms; and (3) discourages premature termination by rewarding coverage of the target room and penalizing unsupported `NOT-FOUND`. Together, this design extends SPL from shortest-path goal reaching to evidence-grounded verification under partial observability.

Standard VLN benchmarks and agents typically assume feasibility and do not expose an explicit `NOT-FOUND` decision (Fig. 1, left). Naively adding a `NOT-FOUND` action and training via imitation learning on reference exploration trajectories can in turn encourage premature abstention (Fig. 1, middle). This arises because exploration is underdetermined—many trajectories can be reasonable and there is no explicit "search finished" signal—so imitation-learned VLN agents can suffer covariate shift (Kamath et al., 2023; Seo et al., 2024) and compounding errors (Zhang et al., 2024a), which often manifest as unreliable stopping behavior (Xiang et al., 2020). In VLN-NF, this can surface as premature `NOT-FOUND` predictions that are not supported by sufficient evidence (Chakraborty et al., 2025; Wu et al., 2025). Moreover, purely supervised approaches typically require costly trajectory-level annotations (especially for exploration), while LLM-based agents often struggle with room-to-room navigation under partial observability without stepwise guidance. To tackle this challenge, we propose a two-stage hybrid framework, **R**oom-**O**bject **A**ware **M**ovement (**ROAM**; Fig. 1, right), which combines supervised room localization with LLM/VLM-driven in-room exploration and verification. ROAM first localizes the target room using a weakly supervised model, then evidence-gathering explores the room and decides whether the target object is present or should be reported as `NOT-FOUND`.

Experimental results show that existing VLN agents struggle on VLN-NF: a supervised baseline (Chen et al., 2022) achieves 4.2 REV-SPL, while LLM-based agents (Zhou et al., 2024b; Georgiev et al., 2024) reach only 1.0 and 1.5 REV-SPL. In contrast, **ROAM** attains 6.1 REV-SPL, improving over the supervised baseline by 45% while requiring only room-level annotations.

Contributions. **1. Task, dataset, and construction.** We introduce **VLN-Not Found (VLN-NF)**, a VLN benchmark of false-premise instructions requiring evidence-gathering exploration and NOT-FOUND, built via an LLM rewrite + VLM absence verification pipeline with <2% human-judged errors. **2. Evaluation.** We propose **REV-SPL**, a variant of SPL that uses a reference exploration protocol and decision-quality metrics to discourage premature stopping and unsupported NOT-FOUND. **3. Method.** We present **ROAM**, a two-stage hybrid that localizes the target room with supervision and then performs VLM/LLM-driven in-room exploration and verification. **4. Results.** ROAM achieves **6.1** REV-SPL, outperforming a supervised baseline by **45%** while requiring only room-level annotations.

2 Related Work

Vision-and-Language Navigation. VLN studies how an embodied agent follows natural-language instructions to navigate in partially observed 3D environments. Benchmarks such as R2R (Anderson et al., 2018b), RxR (Ku et al., 2020), REVERIE (Qi et al., 2020), and CVDN (Thomason et al., 2020) (built on Matterport3D (Chang et al., 2017)) have driven rapid progress, but they typically assume that the instruction is *feasible* and do not require agents to explicitly report failure. A related challenge is reliable exploration and termination under partial observability: exploration is often underdetermined and stopping decisions can be brittle, especially under imitation learning due to covariate shift and compounding errors (Kamath et al., 2023; Seo et al., 2024; Zhang et al., 2024a; Xiang et al., 2020). VLN-NF targets this reliability axis by requiring evidence-grounded NOT-FOUND in addition to navigation.

VLN agents and instruction augmentation. VLN approaches span memory-centric models (Pashevich et al., 2021; Chen et al., 2021), map-based/topological methods (Chen et al., 2022; Wang et al., 2023b; Georgakis et al., 2022), and LLM-based planners or copilots (Zhou et al., 2024b,a; Pan et al., 2023; Qiao et al., 2024; Long et al., 2024; Chen et al., 2024; Lin et al., 2025). Several works further improve navigation by rewriting or generating auxiliary guidance, e.g., disambiguating landmarks (Zhang et al., 2024b; Zhang and Kordjamshidi, 2023), jointly learning instruction following and generation (Wang et al., 2023a), or enabling controllable instruction generation (Kong et al., 2024). These methods largely assume that a valid goal exists and use rewriting to make instructions *easier to follow*. In contrast, our pipeline rewrites feasible instances into *false-premise* (infeasible) ones and automatically verifies target absence, enabling systematic evaluation of NOT-FOUND decision-making.

Instruction unreliability in VLN. Prior work has examined how corrupted or imperfect instructions affect VLN agents. Hahn et al. (2023) perturb instruction tokens (e.g., directions, nouns, numbers) to analyze sensitivity, highlighting the importance of object nouns. Mind the Error (Taioli et al., 2024) modifies VLN-CE instructions (Krantz et al., 2020) and studies detection/correction of *stepwise* instruction errors, where the high-level goal remains valid but intermediate guidance is flawed. Our setting differs in that the instruction can be semantically well-formed yet factually incorrect at the *goal level* (false premise), so the correct behavior is to gather evidence and output NOT-FOUND when appropriate.

Infeasibility, mismatch, and abstention across embodied tasks. MoTIF (Burns et al., 2022) introduces unknown command feasibility with follow-up questions, but operates in a mobile-app setting where the agent has a fully observable 2D screen and access to a complete human exploration trace. R2R-UNO (Hong et al., 2024) creates instruction–environment mismatch via unexpected obstructions, focusing on navigability changes rather than semantic goal absence. CARe (Ko et al., 2024) revises object-navigation decisions when pre-explored semantic maps are inaccurate using uncertainty estimation and multi-view consistency. Concurrent with our work, ADAPT (Chen et al., 2026) also studies infeasibility during navigation, but focuses more on resource-level affordance failures (e.g., unavailable ovens) than on evidence-grounded absence verification. Beyond navigation, recent work studies false-premise handling in VLA manipulation (Hsieh et al., 2025) and abstention or hallucinations under instruction–visual inconsistencies in embodied QA and embodied agents (Wu et al., 2024, 2025; Chakraborty et al., 2025). **VLN-NF complements these lines by instantiating *goal-level false premises* in 3D partially observable VLN, where absence must be justified through autonomous exploration and explicit NOT-FOUND.**

Split	#Scans	#Pairs	#Instr.	#FOUND	#NF
Train	55	2,362	4,724	2,362	2,362
Val-seen	38	234	468	234	234
Val-unseen	10	718	1,436	718	718

Table 1: **VLN-NF statistics.** We use *scan* to denote a Matterport3D building (scan ID). Each *pair* consists of one original REVERIE instruction (`FOUND`, feasible) and one generated VLN-NF instruction (NF, `NOT-FOUND`/infeasible), sharing the same scan, start viewpoint, and target room. Val-seen scans are a subset of the training scans.

3 The VLN-NF Dataset

VLN-NF builds on the Matterport3D environment (Chang et al., 2017) and extends REVERIE (Qi et al., 2020) with infeasible goal descriptions. We adopt REVERIE as our base benchmark because it emphasizes goal-driven navigation and remote object grounding: instructions typically specify the target (and its surrounding context) rather than prescribing a step-by-step route as in R2R (Anderson et al., 2018b) and related datasets. Importantly, our construction pipeline is modular and can be transferred to other VLN datasets in future work.

Dataset statistics and pairing. Table 1 summarizes VLN-NF. We inherit REVERIE's environment, viewpoint graph, and official train/val_seen/val_unseen splits (Qi et al., 2020) (Matterport3D scans (Chang et al., 2017)); we use *scan* to denote a building and *room* to denote a segmented region within a scan. Starting from REVERIE, we filter out instruction instances that are incompatible with VLN-NF, including: (i) instructions without a well-defined target room/object for reliable navigation and verification, (ii) cases where our rewriting+verification pipeline cannot confidently produce an absent-but-plausible replacement target after resampling (e.g., the verifier repeatedly detects the substituted object), and (iii) episodes where the target-room connectivity/visibility is insufficient for our reference exploration protocol to achieve the minimum object-coverage requirement (used for filtering and evaluation). For each remaining REVERIE instruction instance, we construct a one-to-one VLN-NF counterpart by rewriting the target object to a plausible but absent object and verifying its absence, yielding paired `FOUND` (feasible)/NF (`NOT-FOUND`, infeasible) instructions with identical navigation context (same scan, start viewpoint, and target room). The detailed statistics are shown in Table 1.

3.1 Task Definition

VLN-NF extends REVERIE-style VLN (Qi et al., 2020) with a `NOT-FOUND` option. An agent follows an instruction $X = \langle w_1, \ldots, w_L \rangle$ to navigate a Matterport3D scan represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ is a navigable viewpoint with a panoramic RGB observation (discretized into n view images). At step t, the agent selects (i) a neighboring viewpoint to move to, (ii) STOP and output a grounded target object as in REVERIE, or (iii) `NOT-FOUND` to abstain when the instruction is infeasible (i.e., the referenced target does not exist in the specified room). Thus, each episode requires both navigation to the target room and a final decision of `FOUND` vs. `NOT-FOUND`.

3.2 Dataset Construction

Annotating exploration behavior is inherently challenging: exploration trajectories are typically long (making dense annotation costly) and underdetermined—many distinct paths can be valid—unlike traditional VLN exploitation, where shortest paths provide a clear reference. To address this, we propose a scalable pipeline to augment existing VLN datasets by (1) rewriting each instruction to create a linguistically natural but infeasible counterpart, (2) constructing an exploration protocol that can supervise imitation learning and serve as reference labels for evaluation.

As shown in Fig. 2, we generate an infeasible counterpart for each REVERIE instance via a **rewrite-and-verify** loop[1]. Given instruction $\mathcal{X}$ with target object o, the Rewriter [2] proposes a plausible replacement o' that is not in the target-room object list $\mathcal{A}$, and rewrites $\mathcal{X}$ into $\mathcal{X}'$ by substituting $o \rightarrow o'$ while enforcing commonsense plausibility. The Verifier then runs open-vocabulary detection over all panoramas in the target room; if any view detects o', we blacklist it and resample, otherwise we accept $\mathcal{X}'$ as an NF instruction. A manual audit on 5% of pairs yields $< 2\%$ errors[3], indicating that our rewriter-verifier pipeline reliably produces truly infeasible (NF) targets in the specified room.

Generating Reference Exploration Paths. To decide `FOUND` vs. `NOT-FOUND`, an agent must explore the target room; however, exploration tra-

[1]An example is provided in the Appendix.
[2]Prompts are provided in the Appendix.
[3]Details are provided in the Appendix.

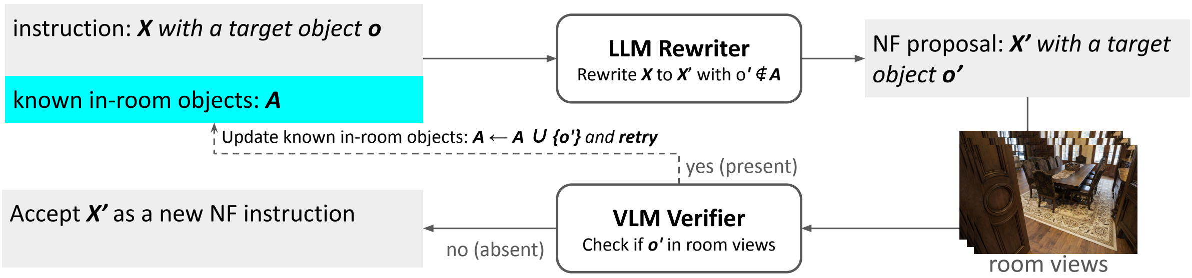

Figure 2: We design a scalable pipeline to rewrite VLN instructions and generate NF (`NOT-FOUND`, infeasible) ones. An LLM-based rewriter uses commonsense priors to generate physically plausible instructions (e.g., *water the plant underneath the window*→*clean the couch underneath the window*), while a VLM-based verifier checks that the referenced target entity is absent from the scene, ensuring NF correctness (Sec. 3).

Algorithm 1 Extract Reach Path

original_final_vp $\leftarrow$ gt_path$[-1]$
room_vps $\leftarrow$ VPSINSAMEROOM(original_final_vp)
reach_path $\leftarrow$ [ ]
for each vp in gt_path **do**
    reach_path.append(vp)
    if vp $\in$ room_vps **then**
        break
return reach_path

Algorithm 2 Reference Exploration Path (Localized / Landmark-Cued)

o: original target object (from REVERIE annotations)
$\mathcal{V}_{\text{vis}}(o)$: viewpoints in the target room that can see o
v_{start}: endpoint of the reach path (room entry viewpoint)

cand_vps $\leftarrow \mathcal{V}_{\text{vis}}(o)$
explore_path $\leftarrow$ BRUTEFORCETSP(v_{start}, cand_vps)
return explore_path

Algorithm 3 Reference Exploration Path (Coverage / No-Landmark)

v_{start} = endpoint of the reach path (room entry viewpoint)
$\mathcal{V}_{\text{room}}$ = all viewpoints in the target room
$\mathcal{O}$ = set of all objects in the target room
$\mathcal{O}_{\text{seen}} = \emptyset$
explore_path $= [v_{\text{start}}]$
visited $= \{v_{\text{start}}\}$
while True **do**
    v_{curr} = explore_path$[-1]$
    $\mathcal{N}$ = unvisited neighbors of v_{curr}
    if $\mathcal{N} = \emptyset$ **then**
        break
    $v_{\text{next}} = \arg\max_{v \in \mathcal{N}} |\text{OBJECTSVISIBLE}(v) \setminus \mathcal{O}_{\text{seen}}|$
    explore_path.append(v_{next})
    visited.add(v_{next})
    $\mathcal{O}_{\text{seen}} \leftarrow \mathcal{O}_{\text{seen}} \cup \text{OBJECTSVISIBLE}(v_{\text{next}})$
    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

jectories are long and underdetermined, making dense human annotation costly. We therefore define a *reference exploration protocol* (used for training/evaluation only; not exposed to the agent) with two cases. (1) **Localized / landmark-cued search.** When the instruction provides strong spatial cues (e.g., "beside the chair"), the plausible search region is localized. Since our NF instruction is created by replacing the target object while keeping the surrounding description unchanged, we approximate this region by the viewpoints in the target room that can see the original target object o. The candidate set is small (typically $N \leq 15$), so we compute a shortest visit order starting from the room entry viewpoint using an exhaustive TSP solver (Alg. 2). (2) **Coverage / no-landmark search.** Without such cues, we use a greedy explorer that repeatedly moves to the neighboring viewpoint that reveals the most *new* room objects , stopping when all room objects are observed or no unvisited neighbors remain (Alg. 3). If the final object coverage is below 85%, we discard the instance. Across retained subsets induced by stricter minimum reference-coverage thresholds (0.85/0.90/0.95), method ranking remains unchanged; we report the full sensitivity analysis in the appendix.

3.3 Evaluation Metrics

Exploration under partial observability is underdetermined and long-horizon, so metrics must avoid rewarding premature "I'm done" decisions. In VLN-NF, an agent must (i) *reach* the target room and (ii) make a final *decision* of `FOUND` vs. `NOT-FOUND`. We report a primary composite metric (REV-SPL) together with a small set of decomposed diagnostics for interpretability.

Success Rates. **Reach SR** is the fraction of episodes where the agent visits at least one viewpoint inside the target room. **Reach&Decision SR** counts episodes where the agent both reaches the correct room and makes the correct final decision.

Path-length Efficiency. Following SPL (Anderson et al., 2018a), **Reach SPL** evaluates efficiency of reaching the target room using the reference reach path length $l_{\text{reach},i}$:

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

REV-SPL (Reach–Explore–Verify SPL). Our primary metric **REV-SPL** extends SPL to capture (i) reaching the target room, (ii) correct FOUND/NOT-FOUND decision, and (iii) sufficient in-room exploration. Let $S_i^r \in \{0,1\}$ indicate whether the agent reaches the target room, and let $S_i^d \in \{0,1\}$ indicate whether it makes the correct final decision. We define object coverage $C_i \in [0,1]$ as $C_i = \frac{\left|\bigcup_{v\in P_i} \texttt{OBJECTSVISIBLE}(v)\right|}{|O_i|}$, where P_i is the agent's in-room trajectory and O_i is the set of annotated object instances in the target room. Let ℓ_i and p_i denote the reference and actual exploration lengths, respectively.

We compute

$$\text{REV}-\text{SPL} = \frac{1}{N}\sum_{i} S_i^r S_i^d C_i \frac{\ell_i}{\max(p_i, \ell_i)} \cdot \min\left(1, \frac{p_i}{\ell_i}\right). \tag{2}$$

We additionally report **Coverage** and average path lengths as diagnostics.

4 ROAM Framework

VLN-NF evaluates *evidence-grounded* NOT-FOUND under partial observability: the agent must reach the referenced room, search for the target, and output FOUND or NOT-FOUND. This requires reliable room localization and coverage-aware in-room exploration, declaring NOT-FOUND only after sufficient search yields no evidence of the target. Note that our goal here is not to formalize absence as a complete belief-theoretic inference problem, but to provide a tractable and reproducible operationalization that can be benchmarked under realistic VLN constraints. We observe that supervised VLN policies such as DUET (Chen et al., 2022) are strong at inter-room navigation but often under-explore once inside the target room, since exploration is underdetermined and lacks an explicit "search finished" signal. In contrast, LLM-based policies (Zhou et al., 2024b) can leverage commonsense priors to guide in-room search given context, but they often struggle with floorplan-level navigation under partial observability. Motivated by these observations, we propose Room-Object Aware Movement (ROAM, Fig. 3), a hybrid framework with a **Room-Level Navigator** (Sec. 4.1) for robust room localization and an **In-room Explorer** (Sec. 4.2) for evidence-gathering search. To compensate for LLMs' weak geometric intuition, we further add a plug-and-play **Free-space Raycasting Estimation Engine** (FREE) that provides a free-space clearance signal (Sec. 4.3) to steer exploration toward larger unsearched regions with semantic context.

4.1 Room-Level Navigator

We model each environment as a navigation graph $G = (V, E)$, where each node $v \in V$ is a panoramic viewpoint and each edge $(u, v) \in E$ denotes a traversable transition. Let $r(v)$ be the room label associated with viewpoint v, and let $r^\star$ be the target room parsed from the instruction. We define the target-room subgraph as $R = \{v \in V \mid r(v) = r^\star\}$.

Given an episode starting at v_{start}, we define an *entry viewpoint*

$$v_{\text{room}} = \arg\min_{v\in R} d_G(v_{\text{start}}, v), \tag{3}$$

where $d_G(\cdot,\cdot)$ is the shortest-path distance on G. This reduces the original reach-and-explore task to a room-reaching problem: navigate from v_{start} to v_{room}. In our experiments, we obtain the room-entry target viewpoint from reach-to-room path prefixes (ending at room entry); more generally, since this stage only requires room identity, it can be trained from weak supervision that labels viewpoints with room IDs and synthesizes reach-to-room paths (e.g., shortest paths to any viewpoint in the target room), without trajectory-level *in-room exploration* annotation. We adopt DUET (Chen et al., 2022) as the backbone, though any VLN policy capable of room-level navigation can be used. At inference, the navigator runs until the agent enters the target room (i.e., $r(v_t) = r^\star$) or a step budget is reached, after which we hand off to the in-room exploration stage.

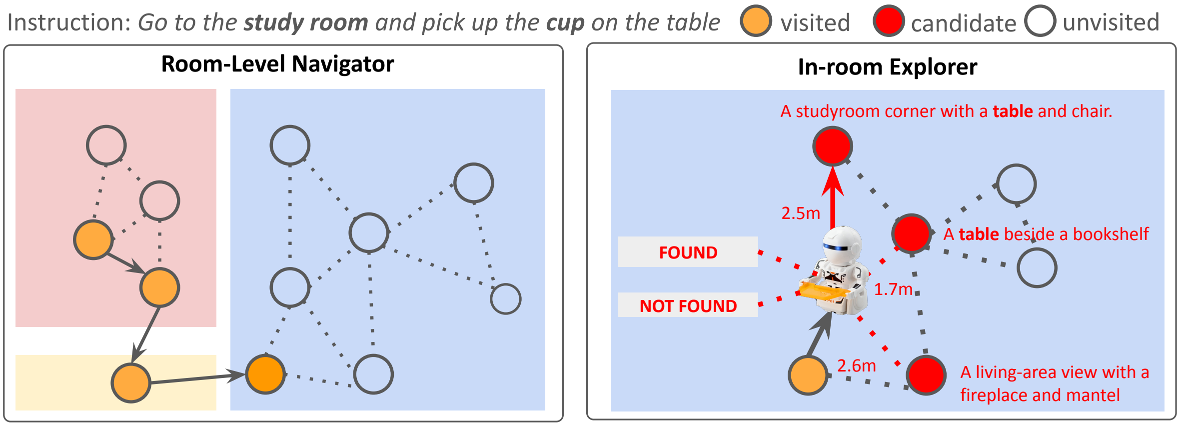

Figure 3: Overview of our two-stage framework, **ROAM** (Sec. 4). **Left** (Sec. 4.1): a room-level navigator trained with *room labels only* guides the agent to enter the target room. **Right** (Sec. 4.2): an LLM-based in-room explorer selects the next viewpoint using VLM captions as semantic context and a geometric coverage prior from FREE (Sec. 4.3) to favor headings that lead to larger unsearched regions.

4.2 In-room Explorer

After entering the target room R, the In-room Explorer[4] searches for the target object o' and decides whether to output FOUND or NOT-FOUND. At time t, the agent is at viewpoint $v_t \in R$ with a history H_t (visited viewpoints and accumulated evidence), and has a set of reachable neighbors $\mathcal{N}(v_t)$.

We use a VLM stack to convert observations into semantic context (scene captions and detected objects), and an LLM to select actions in a closed loop. The action space is

$$\mathcal{A}_t = \{\texttt{move}(v) \mid v \in \mathcal{N}(v_t)\} \cup \{\texttt{FOUND}, \texttt{NOT-FOUND}\}. \quad (4)$$

Let $D(v, o') \in [0, 1]$ denote the confidence from an open-vocabulary detector for object o' at viewpoint v (Grounding-DINO (Liu et al., 2024) in our implementation). In addition, for each candidate move $v \in \mathcal{N}(v_t)$, we compute a free-space clearance cue $d_{\text{free}}(v_t \to v)$ using FREE (Sec. 4.3).

The explorer terminates with FOUND if $D(v_t, o') \geq \tau$ at any step; otherwise, it continues exploring and may output NOT-FOUND only when an evidence-backed stopping criterion is satisfied or a hard constraint is met (e.g., frontier exhaustion or a fixed exploration budget). We implement the explorer by augmenting NavGPT (Zhou et al., 2024b) with the FREE clearance cue, where NavGPT tokenizes observations via captioning and open-vocabulary detection and uses an LLM to reason over accumulated evidence and decide whether to continue or terminate.

[4] Prompts are provided in the Appendix.

4.3 FREE

When exploring, an agent benefits from both commonsense priors (e.g., cups are likely on tables) and geometric coverage cues (e.g., moving toward a large unsearched area is often informative). Prior work suggests that LLMs encode commonsense knowledge but can be unreliable at geometric/spatial reasoning (Comsa and Narayanan, 2023; Yan et al., 2023; Cheng et al., 2024; Yang et al., 2024). To bridge this gap, we introduce a plug-and-play **Free-space Raycasting Estimation Engine** (FREE), which provides a *free-space clearance* signal to guide exploration.

Given the current observation at v_t, FREE estimates, for each candidate heading θ_k, a clearance distance $d_{\text{free}}(\theta_k)$ indicating how far the agent can move before encountering an obstacle (Fig. 4). Concretely, we segment navigable regions (Grounded-SAM (Ren et al., 2024) in our experiments) and back-project the mask into 3D to perform raycasting. We then map each candidate move $v \in \mathcal{N}(v_t)$ to its heading $\theta(v)$ and attach $d_{\text{free}}(\theta(v))$ to the planner prompt as a geometric prior, encouraging actions that lead to larger unsearched regions.

5 Experiments

5.1 Baseline Methods

We adapt several representative VLN baselines to our dataset VLN-NF. **DUET** (Chen et al., 2022) is extended by adding an additional NOT-FOUND class in its decision head to support abstention on infeasible episodes. **NavGPT** (Zhou et al., 2024b)

Setting	Method	Coverage	Len (reach)	Len (explore)	Reach SR	Reach&Decision SR	Reach SPL	REV-SPL
supervised	DUET	69.5%	13.1	7.8	53.8%	33.8%	37.0%	4.2%
unsupervised	NavGPT	59.1%	10.6	2.7	8.2%	5.4%	7.0%	1.0%
unsupervised	MapGPT	63.3%	13.3	3.5	30.0%	14.0%	18.2%	3.2%
unsupervised	SoM-Gemini-2.0-Flash	80.9%	10.0	14.2	39.4%	22.0%	28.9%	1.5%
hybrid	ROAM (Ours)-GPT-3.5	82.1%	11.2	9.9	58.6%	37.6%	44.1%	**6.1%**
hybrid	ROAM (Ours)-GPT-4o	**82.8%**	12.8	22.0	**62.6%**	**41.4%**	**45.4%**	5.6%

Table 2: Comparison with baselines on VLN-NF `val_unseen`. We report our primary metric **REV-SPL** (Eq. 2), together with decomposed diagnostics, **Reach SR**, **Reach&Decision SR**, and **Reach SPL**. (Sec. 5.2)

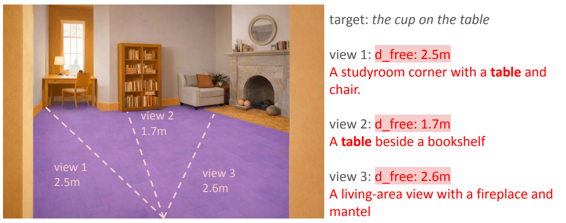

Figure 4: **FREE (Sec. 4.3)** segments navigable floor regions from the current view using an open-vocabulary visual model and uses depth-based raycasting to estimate a free-space clearance d_{free} for each candidate direction. The resulting clearance cues are appended to the LLM prompt to encourage coverage-oriented exploration.

is modified to include an explicit `NOT-FOUND` action in its action space/prompt, allowing the agent to terminate with `NOT-FOUND` when appropriate. **MapGPT** (Chen et al., 2024) is a map-guided GPT-based VLN agent that constructs an online linguistic topological map and performs adaptive multi-step path planning. **SoM-Gemini-2.0-Flash** follows the Set-of-Marks prompting strategy (Yang et al., 2023) and uses Gemini-2.0-Flash as the multimodal backbone, providing a captioning-free alternative to NavGPT's captioning-then-LLM pipeline. Concretely, we overlay candidate viewpoints in Matterport3D with red markers and numeric labels, and query Gemini to select the next action (navigate, stop/`FOUND`, or `NOT-FOUND`). We include it as a planner/map-augmented baseline and adapt it to VLN-NF by evaluating it under the same `FOUND`/`NOT-FOUND` decision protocol and evaluation pipeline as the other methods.

5.2 Results

As shown in Table 2, ROAM achieves the best performance among all compared baselines on VLN-NF under our primary metric REV-SPL.

Findings. (1) Room reaching vs. in-room verification. DUET achieves strong room-reaching (Reach SR 53.8%, Reach SPL 37.0%) but attains only 69.5% coverage and 4.2 REV-SPL, indicating that reaching the target room does not guarantee evidence-gathering verification. **(2) LLM/VLM baselines are unreliable under partial observability.** NavGPT rarely reaches the target room (Reach SR 8.2%) and yields 1.0 REV-SPL; while SoM explores more (Coverage 80.9%, Len(explore) 14.2), it still falls short on joint reach+decision (Reach&Decision SR 22.0%, REV-SPL 1.5), showing that commonsense alone is insufficient for robust long-horizon navigation and stopping. MapGPT improves over other LLM-based baselines (REV-SPL 3.2 vs. 1.0 for NavGPT and 1.5 for SoM), indicating that stronger global planning helps under false-premise instructions. However, it still remains below DUET (4.2) and both ROAM variants (6.1/5.6), suggesting that map-guided planning alone is insufficient for evidence-grounded `FOUND`/`NOT-FOUND` decisions under partial observability. **(3) ROAM closes both gaps.** By combining weakly supervised navigation with LLM/VLM-driven in-room search, ROAM substantially improves both reaching and decision quality (Reach SR 58.6/62.6%, Reach&Decision SR 37.6/41.4%) and achieves the best REV-SPL (6.1/5.6), outperforming DUET by 45% while using only room-level supervision. **(4) REV-SPL captures the efficiency–thoroughness trade-off.** ROAM-GPT-4o attains higher Reach&Decision SR than ROAM-GPT-3.5 (41.4% vs. 37.6%) but explores much longer (22.0 vs. 9.9), leading to slightly lower REV-SPL (5.6 vs. 6.1), consistent with REV-SPL penalizing inefficient wandering despite correct decisions.

Ablation Study. Table 3 demonstrates the effectiveness of our two-stage hybrid framework and the FREE design. Comparing (1) and (2), adding a room-level navigator trained with *room labels only* dramatically improves performance (e.g., Reach SR 7.8%→59.4% and REV-SPL 0.8%→5.6%), indicating that an LLM-only controller struggles with long-horizon room reaching under partial observability. Comparing (2)

#	2 Stage	FREE	Coverage	Reach SR	R&D SR	Reach SPL	REV-SPL
(1)	✗	✗	75.7%	7.8%	4.4%	6.7%	0.8%
(2)	✓	✗	79.2%	**59.4%**	37.2%	**44.2%**	5.6%
(3)	✓	✓	**82.1%**	58.6%	**37.6%**	44.1%	**6.1%**

Table 3: Ablation study of our ROAM framework using GPT-3.5 on the val_unseen split of VLN-NF.

#	2 Stage	FREE	Reach SR	Reach SPL	SR	SPL
(1)	✗	✗	8.1%	6.6%	9.2%	6.8%
(2)	✓	✗	62.0%	48.1%	42.0%	23.7%
(3)	✓	✓	64.0%	48.4%	45.2%	25.1%

Table 4: ROAM (GPT-3.5) on REVERIE val_unseen.

and (3), FREE further increases evidence coverage (79.2%→**82.1%**) and improves REV-SPL (5.6%→**6.1%**), validating that geometric clearance cues complement caption-based commonsense by steering exploration toward larger unsearched regions, while leaving reaching performance essentially unchanged.

Transfer to standard feasible VLN. **(Table 4) shows that ROAM also enhances VLN performance on REVERIE `val_unseen`.** Removing the room-level navigator again leads to very poor navigation (Reach SR 8.1%, SPL 6.8%), while the two-stage design substantially improves both reaching and task success (Reach SR 62.0%, SPL 23.7%). Importantly, **FREE remains beneficial** even in the feasible-only setting, improving SR (42.0%→45.2%) and SPL (23.7%→25.1%), suggesting that FREE-style coverage cues help the LLM planner choose more informative viewpoints without hurting conventional VLN performance.

False `NOT-FOUND` on feasible episodes. As shown in Table 5, on VLN-NF val_unseen, we further audit decision-level false `NOT-FOUND` predictions made by ROAM (GPT-3.5) on `FOUND` episodes. Most such errors are classic room-reaching failures (55.7%), while the remainder are dominated by perception/grounding uncertainty (31.0%) and a smaller exploration-control component (13.3%), suggesting that enabling `NOT-FOUND` mainly amplifies calibration and verification challenges rather than introducing a fundamentally new failure mode.

6 Conclusion

We study *evidence-grounded* `NOT-FOUND` for 3D VLN under unreliable (false-premise) instructions. We introduce VLN-NF, constructed by pairing feasible REVERIE episodes with automatically

Error source	**Share**
Room-reaching failure	55.7%
Perception/grounding uncertainty	31.0%
Exploration-control failure	13.3%

Table 5: Error-source decomposition of decision-level false `NOT-FOUND` predictions made by ROAM (GPT-3.5) on feasible (`FOUND`) episodes in VLN-NF val_unseen.

rewritten-and-verified infeasible counterparts, and propose REV-SPL to evaluate reaching, decision correctness, and evidence-gathering exploration efficiency. We further present ROAM, a two-stage hybrid that combines room-level supervision for robust room localization with VLM/LLM-driven in-room exploration and verification, augmented by a free-space clearance cue (FREE). Across baselines, ROAM achieves the best performance on VLN-NF (6.1 REV-SPL), improving over a strong supervised agent by 45% while requiring only room-level annotations. We view ROAM as a structured first step toward evidence-grounded abstention in 3D VLN under partial observability, while more principled absence reasoning and broader recovery behaviors remain important future directions.

Limitations

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

Data coverage and filtering. The experiments in this paper are based on REVERIE, which we adapt to the VLN-NF setting. To ensure reliable verification and stable reference-exploration generation, we apply strict filtering and currently retain 45% of REVERIE episodes. This trades raw coverage for higher confidence in NF correctness and evaluation stability. Importantly, this filtering is intended as a quality-control gate for evaluability rather than a mechanism to skew the benchmark distribution. As shown in Appendix, the filtering-only subset induces only very small language/object distribution shift relative to full REVERIE, while the additional shift in final VLN-NF is modest and expected from substituting absent-but-plausible targets.

Evaluation of path quality. While we conducted human evaluation on a sampled subset of the VLN-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.

Grounding-confidence threshold sensitivity in ROAM. The performance of the ROAM framework is sensitive to the object grounding model's (e.g., Grounding-DINO) 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.

Scope of false-premise types. VLN-NF currently instantiates a focused form of goal-level false premise: the referenced target object is absent from the room specified by the instruction. We choose this setting as a controlled but practical starting point, since mis-specified object locations are common in human instructions and, under partial observability, absence cannot be verified from a single viewpoint. Importantly, several broader false-premise variants reduce to the same core decision problem once conditioned on a described region; for example, a target located in the wrong room corresponds to "absent in the specified room," and wrong attribute/object descriptions correspond to "absence of the described entity." However, cases with multiple plausible referents or under-specified instructions may require clarification or interactive recovery rather than pure absence verification. Extending VLN-NF to cover these broader variants and richer recovery behaviors is an important direction for future work.

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.

Acknowledgment This work was supported in part by National Science and Technology Council, Taiwan, under Grant NSTC 113-2634-F-002-007.

References

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

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018b. 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.

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A. Plummer. 2022. A dataset for interactive vision-language navigation with unknown command feasibility. In *Computer Vision – ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*, volume 13668 of *Lecture Notes in Computer Science*, pages 312–328. Springer.

Trishna Chakraborty, Udita Ghosh, Xiaopan Zhang, Fahim Faisal Niloy, Yue Dong, Jiachen Li, Amit Roy-Chowdhury, and Chengyu Song. 2025. HEAL: An empirical study on hallucinations in embodied agents driven by large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21226–21243, Suzhou, China. Association for Computational Linguistics.

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

Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee K. Wong. 2024. 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*.

Pei-An Chen, Yong-Ching Liang, Jia-Fong Yeh, Hung-Ting Su, Yi-Ting Chen, Min Sun, and Winston Hsu. 2026. Adapt: Benchmarking commonsense planning under unspecified affordance constraints. *Preprint*, arXiv:2604.14902.

Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. 2021. History aware multimodal transformer for vision-and-language navigation. *Advances in neural information processing systems*, 34:5834–5847.

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

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei

Liu. 2024. Spatialrgpt: Grounded spatial reasoning in vision-language models. In *Advances in Neural Information Processing Systems*.

Iulia Comsa and Srini Narayanan. 2023. A benchmark for reasoning with spatial prepositions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16328–16335, Singapore. Association for Computational Linguistics.

Georgios Georgakis, Karl Schmeckpeper, Karan Wanchoo, Soham Dan, Eleni Miltsakaki, Dan Roth, and Kostas Daniilidis. 2022. 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.

Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Meera Hahn, Amit Raj, and James M. Rehg. 2023. 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.

Haodong Hong, Sen Wang, Zi Huang, Qi Wu, and Jiajun Liu. 2024. Navigating beyond instructions: Vision-and-language navigation in obstructed environments. *arXiv preprint arXiv:2407.21452*. Accepted to ACM Multimedia (MM) 2024.

Wen-Han Hsieh, Elvis Hsieh, Dantong Niu, Trevor Darrell, Roei Herzig, and David M. Chan. 2025. Do what? teaching vision-language-action models to reject the impossible. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11861–11869, Suzhou, China. Association for Computational Linguistics.

Aishwarya Kamath, Peter Anderson, Su Wang, Jing Yu Koh, Alexander Ku, Austin Waters, Yinfei Yang, Jason Baldridge, and Zarana Parekh. 2023. A new path: Scaling vision-and-language navigation with synthetic instructions and imitation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10813–10823.

Po-Chen Ko, Hung-Ting Su, CY Chen, Jia-Fong Yeh, Min Sun, and Winston H. Hsu. 2024. Context-aware replanning with pre-explored semantic map for object navigation. In *8th Annual Conference on Robot Learning*.

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

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

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

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

Bingqian Lin, Yunshuang Nie, Ziming Wei, Jiaqi Chen, Shikui Ma, Jianhua Han, Hang Xu, Xiaojun Chang, and Xiaodan Liang. 2025. Navcot: Boosting llm-based vision-and-language navigation via learning disentangled reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, and 1 others. 2024. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer.

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

OpenAI. 2023. Gpt-4 technical report. Accessed: 2025-05-18.

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

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

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. 2020. 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.

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

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, and 1 others. 2024. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*.

Seokin Seo, Byung-Jun Lee, Jongmin Lee, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. 2024. Mitigating covariate shift in behavioral cloning via robust stationary distribution correction. In *Advances in Neural Information Processing Systems*, volume 37.

Francesco Taioli, Stefano Rosa, Alberto Castellini, Lorenzo Natale, Alessio Del Bue, Alessandro Farinelli, Marco Cristani, and Yiming Wang. 2024. 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.

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

Hongbin Wang, Todd R Johnson, Jiajie Zhang, and Yue Wang. 2002. A study of object-location memory. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 24.

Xiaohan Wang, Wenguan Wang, Jiayi Shao, and Yi Yang. 2023a. 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.

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

Tao Wu, Chuhao Zhou, Yen Heng Wong, Lin Gu, and Jianfei Yang. 2024. Noisyeqa: Benchmarking embodied question answering against noisy queries. *arXiv preprint arXiv:2412.10726*.

Tao Wu, Chuhao Zhou, Guangyu Zhao, Haozhi Cao, Yewen Pu, and Jianfei Yang. 2025. When robots should say “i don’t know”: Benchmarking abstention in embodied question answering. *arXiv preprint arXiv:2512.04597*.

Jiannan Xiang, Xin Wang, and William Yang Wang. 2020. Learning to stop: A simple yet effective approach to urban vision-language navigation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 699–707, Online. Association for Computational Linguistics.

He Yan, Xinyao Hu, Xiangpeng Wan, Chengyu Huang, Kai Zou, and Shiqi Xu. 2023. Inherent limitations of gpt=4 regarding spatial information. *Preprint*, arXiv:2312.03042.

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

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. 2024. Thinking in space: How multimodal large language models see, remember, and recall spaces. *Preprint*, arXiv:2412.14171.

Xiaoyu Zhang, Matthew Chang, Pranav Kumar, and Saurabh Gupta. 2024a. Diffusion meets DAgger: Supercharging eye-in-hand imitation learning. In *Robotics: Science and Systems (RSS)*.

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

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

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

Gengze Zhou, Yicong Hong, and Qi Wu. 2024b. 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.

A Appendix

A.1 Implementation Details

Dataset Generation Pipeline. We use `Gemini 1.5 Flash` (Georgiev et al., 2024) 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`(OpenAI, 2023) 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`(Li et al., 2022) as the Verifier, using a confidence threshold of 0.7.

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

Listing 1: Prompt for the Rewriter

```
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:
inputs:
    ___inputs___
outputs:
```

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

Listing 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'
}
```

ROAM Framework. In the Room-Object Extractor, we use `Gemini 1.5 Flash` (Georgiev et al., 2024). We first fine-tune DUET(Chen et al., 2022) (the REVERIE version with object bounding box information) on VLN-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. At inference, we first run the Room Navigator from the start viewpoint. The Room Explorer is invoked only after the navigator enters the target room (i.e., $r(v_t) = r^\star$); otherwise, the episode terminates when the navigation step budget is exhausted. No ground-truth room entry or oracle hand-off is used at test time. We then use `GPT-4o` or `GPT-3.5-turbo`(OpenAI, 2023) as the Room Explorer. This Room Explorer is adapted from NavGPT(Zhou et al., 2024b), with prompts modified to suit the VLN-NF task. Additionally, we incorporate distance information estimated by our FREE module. For the grounding component, we use the `grounding-dino-base`(Liu et al., 2024) 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 3: Prompt for the In-Room Explorer

```
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 cannot 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
    energy 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
    Observation 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, 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 hidden. 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}
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"
```

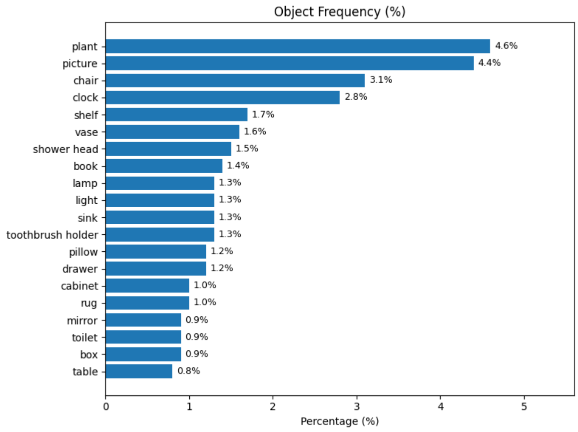

Figure 5: Most occurred objects for NOT-FOUND instructions.

```
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}"""
```

ROAM Framework - Free-space Raycasting Estimation Engine (FREE) . We first utilize Grounded SAM (GroundingDINO_SwinT + sam_vit_h_4b8939 version)(Ren et al., 2024) 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 6)

After completing all segmentation, we perform Free-space Raycasting to estimate distances. First,

Figure 6: 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).

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.

Human Evaluation. We randomly sampled 5% of the data. Annotators were shown: (1) the newly generated target object, and (2) all panoramas of the corresponding target room. They verified whether the new object truly did not exist in that room.

A.2 Potential Risks

We do not foresee direct risks related to malicious misuse, as our dataset VLN-NF does not include sensitive information, human identities, or unauthorized photos of private residences.

However, we acknowledge potential risks specific to our work. Since our dataset VLN-NF and framework ROAM are designed to detect and respond to potentially incorrect instructions, improper handling of uncertainty may lead to premature task termination or refusal to act when the instruction is merely ambiguous rather than incorrect. Over-reliance on `NOT-FOUND` predictions could reduce system reliability in real-world deployment. To mitigate these risks, we recommend incorporating uncertainty estimation, confidence calibration and fallback interaction (e.g., asking for clarification).

Fairness concerns such as overexposure to certain environments or instruction styles are also worth noting. As our dataset is built on REVERIE and Matterport3D Simulator, it primarily reflects a limited range of indoor environments. Additionally, all instructions are currently in English, which may disadvantage models trained for multilingual or non-English scenarios.

Overall, our work does not involve large-scale model training (e.g., LLM pretraining), does not pose privacy risks. We believe this benchmark promotes safer, more cautious navigation behavior by encouraging systems to recognize failure modes.

A.3 License and Usage of REVERIE

The REVERIE dataset used in this work is based on the Matterport3D environment and is distributed under a non-commercial research license. We follow all usage terms specified in the original REVERIE paper (Qi et al., 2020) and the Matterport3D dataset (Chang et al., 2017). Our extensions (e.g., instruction augmentation and path generation) are built upon these publicly released annotations and simulation environments.

We confirm that our use of REVERIE complies with the dataset's license, and all modifications are intended for non-commercial academic research. The newly generated data will be released under similar terms to support future research in the community.

Furthermore, the datasets used do not contain any personally identifiable information or offensive content. We conducted manual sampling to verify this and ensure data quality. We have ensured that our use and extension of these artifacts respect the original creators' rights and adhere to ethical standards.

A.4 Sensitivity to Minimum Reference-Coverage Threshold

Our reference exploration protocol uses a minimum reference-coverage threshold to filter episodes whose target-room visibility graph is too sparse for stable evaluation. In the main paper, we retain episodes with minimum reference coverage ≥ 0.85. To verify that our conclusions are not sensitive to this choice, we evaluate stricter retained subsets induced by thresholds 0.90 and 0.95, while keeping agent trajectories unchanged.

th	ROAM R&D / REV	MapGPT R&D / REV	Retained episodes
85%	37.6 / 6.1	14.0 / 3.2	100.0%
90%	37.6 / 6.1	15.4 / 3.7	83.8%
95%	37.8 / 6.5	14.2 / 3.3	76.1%

Table 6: Sensitivity to the minimum reference-coverage threshold. Each cell reports Reach&Decision SR / REV-SPL (both in %). This analysis varies only the retained evaluation subset (i.e., stricter evaluability filtering), while keeping agent trajectories unchanged. The retained-episodes column is reported relative to the 0.85 retained set.

Interpretation. The results are stable across stricter retained subsets. ROAM (GPT-3.5) maintains nearly identical Reach&Decision SR (37.6–37.8%) and similar REV-SPL (6.1–6.5%) across thresholds, while MapGPT also shows only limited variation (R&D SR 14.0–15.4%, REV-SPL 3.2–3.7%). Crucially, the method ranking is unchanged under all thresholds (ROAM > MAPGPT), indicating that our conclusions are not sensitive to the specific 0.85 choice and that the threshold does not create method-dependent advantages. Increasing the threshold simply filters out additional low-visibility cases, reducing the number of retained episodes, while leaving the qualitative conclusions unchanged.

A.5 Use of AI Assistants

We used AI assistants during the research and writing process to improve writing clarity and presentation. Specifically, ChatGPT was used for language polishing (e.g., grammar, rephrasing, and clarity edits), and for suggesting code improvements. It was also used to draft or edit *non-empirical schematic figures* (e.g., pipeline diagrams/icons) for illustration purposes only. We also used ChatGPT and NotebookLLM to help organize and summarize literature during the early stages of the related work review. **No AI-generated images, captions, or other synthetic content were used as experimental inputs or as evidence for the reported results. All AI-assisted outputs were reviewed and edited by the authors, and all critical decisions regarding modeling, implementation, and analysis were made by the authors.**

A.6 Filtering, Distribution Shift, and Scalability

We further analyze whether the strict filtering used in VLN-NF substantially changes the underlying language/object distribution inherited from REVERIE. To separate the effect of filtering from the effect of NF construction itself, we compare: (i) **REVERIE vs. retained REVERIE** (filtering only), and (ii) **REVERIE vs. final VLN-NF** (post rewrite/verify).

Interpretation. The filtering-only comparison shows extremely small distribution shift: word/object rank agreement remains very high and normalized object proportions remain nearly unchanged. This supports the view that strict filtering primarily enforces evaluability and label reliability, rather than introducing a substantial bias in language or object domain. The additional shift from final VLN-NF is modest and expected, since NF construction intentionally replaces the original target with an absent but plausible alternative. Importantly, this shift does not collapse the benchmark into a degenerate object distribution; rather, it reflects the intended transformation needed to create false-premise instructions.

Scalability. Although strict QC reduces the current benchmark size, the rewrite-and-verify pipeline is naturally scalable. On the negative side, a single feasible episode can yield multiple NF variants by resampling alternative absent targets with blacklisting, and the difficulty of generated negatives can be controlled through prompting or sampling priors (e.g., same-category vs. distant-category substitutes, or landmark-cued vs. no-landmark settings). On the positive side, feasible instances can be expanded through lightweight heuristics such as instruction paraphrases or start-node variations that preserve target-room reachability, and then paired with the same rewrite-and-verify pipeline to generate additional NF counterparts. These properties make VLN-NF a high-confidence benchmark in its current form while leaving a clear path for future scaling.

A.7 Model Size And Budget

Baseline - DUET The number of parameters is nearly identical to that of DUET. Training on an RTX 4090 took approximately 48 hours

Baseline - NavGPT Using GPT-4 to run inference on the val_unseen split costs approximately less than $8 USD.

Baseline - MapGPT Using GPT-4 to run inference on the val_unseen split costs approximately $ 215 USD.

Metric	Filtering only	Final VLN-NF
Words: Jaccard@50 $\uparrow$	0.887	0.852
Words: Spearman ρ@50 $\uparrow$	0.942	0.863
Words: JS $\downarrow$	0.0106	0.0148
Objects: Spearman ρ $\uparrow$	0.985	0.859
Objects: Cosine sim. $\uparrow$	0.998	0.958
Objects: JS $\downarrow$	0.0017	0.0414
Objects: TV $\downarrow$	0.0330	0.1627

Table 7: Distribution-shift analysis for strict filtering and final NF construction. "Filtering only" compares full REVERIE against the retained REVERIE subset after QC filtering. "Final VLN-NF" compares full REVERIE against the final post rewrite/verify benchmark. Higher is better for Jaccard, Spearman, and cosine similarity; lower is better for JS and TV.

Baseline - SoM Using Gemini-2.0 Flash to run inference on the val_unseen split costs approximately $ 180 USD.

Baseline - ROAM We trained DUET as a Room Navigator. Training on an RTX 4090 took approximately 48 hours. During the Fine-grained stage, we used GPT-3.5 for inference on the val_unseen split, which incurred a cost of approximately less than $3 USD.