GHOSTANCHOR NAVIGATION: EXECUTION-UPDATED TOPOLOGICAL CONTROL WITH LOCATE–VERIFY–GROUND TARGET APPROACH FOR ZERO-SHOT OBJECT NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Zero-shot object navigation requires an agent to find and approach a text-specified object category in an unknown environment. Vision-language models (VLMs) can infer search directions and local actions from current observations, but a one-step decision does not retain the execution history of previously selected candidates. We present GhostAnchor Navigation (GA-Nav), a training-free framework that applies VLMs to zero-shot ObjectNav through execution-updated topological control. Its Node–Ghost Topology Memory (NGTM) associates depth-derived candidate waypoints across observations and stores candidates that are not matched to Nodes as Ghosts with explicit states. After each action, NGTM updates the selected Ghost from the measured pose change and uses the resulting topology to filter the directions available to the next VLM decision. When the target becomes visible, the Locate–Verify–Ground (LVG) procedure selects and verifies a target observation waypoint, then estimates a 3D target anchor from RGB-D for approach and stopping. We evaluate GA-Nav on the complete validation sets of HM3D v0.1, HM3D v0.2, and MP3D. Among the training-free zero-shot methods compared, GA-Nav ranks first in both SR and SPL on all three benchmarks. Specifically, GA-Nav achieves absolute SR/SPL gains of $0.2/4.1$ percentage points on HM3D v0.1, $0.9/5.8$ points on HM3D v0.2, and $4.1/7.5$ points on MP3D, relative to the best prior training-free zero-shot result for each metric.

1 INTRODUCTION

Object-goal navigation (ObjectNav) requires an embodied agent to find and approach an object of a specified category in an unknown environment, and is a fundamental capability for household service robots to acquire task-relevant objects Batra et al. (2020). Conventional methods learn search policies through reinforcement learning, imitation learning, or semantic mapping, whereas zero-shot methods use open-vocabulary visual representations and commonsense knowledge to handle targets and environments not covered during training Majumdar et al. (2022); Gadre et al. (2023); Zhou et al. (2023). Vision-language models (VLMs) further use current observations and target semantics to estimate the exploration value of candidate regions or directly infer navigation directions and local actions Yokoyama et al. (2024); Goetting et al. (2025); Nie et al. (2025).

Existing zero-shot methods connect target cues to navigation through several memory and action interfaces. Semantic frontiers and open-set maps project target relevance onto spatial regions Chen et al. (2023); Kuang et al. (2024); VoroNav and TopV-Nav use topological or top-view structures for direction selection Wu et al. (2024); Zhong et al. (2026); PixelNav and VLMNav generate target positions or local actions from egocentric images Cai et al. (2024); Goetting et al. (2025). WMNav accumulates panoramic predictions in a Curiosity Value Map Nie et al. (2025), whereas ReMemNav retrieves pose-indexed observation history when reconsidering a direction Wu et al. (2026). These representations preserve spatial regions, places, or observations, but they differ in how execution changes the information available to the next decision.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Online topological maps already organize visited places and visible but unexplored regions through regular and Ghost nodes Chaplot et al. (2020); Cui et al. (2024). ETPNav further self-organizes predicted waypoints into a topology for learned vision-language navigation and associates waypoints across observations An et al. (2025). These works establish Node–Ghost topology as a navigation representation. For training-free zero-shot ObjectNav with VLMs, the remaining control problem is to carry the outcome of an action toward a candidate into the next semantic decision.

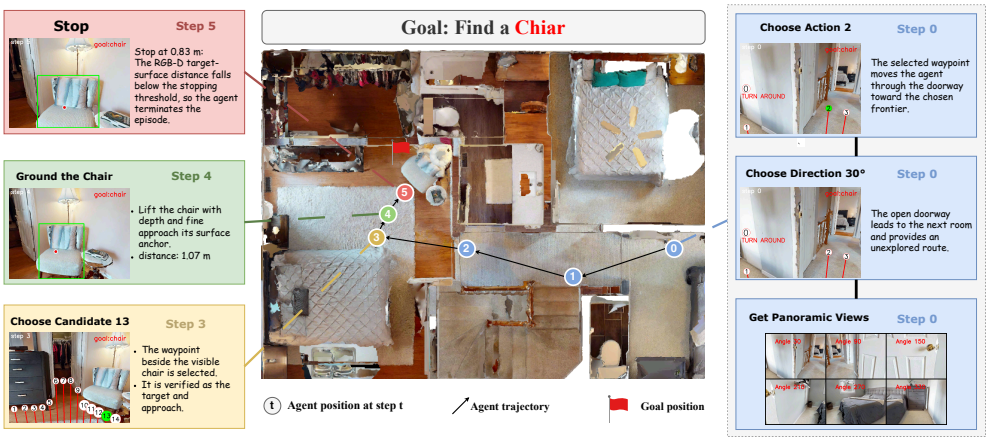

Figure 1: A successful chair-navigation episode with GA-Nav. At Step 0, the agent acquires panoramic RGB-D observations, selects the 30° direction under the Node–Ghost topology constraint, and executes Action 2 to pass through the doorway toward an unexplored area. After the chair becomes visible, LocatorVLM selects Candidate 13 and VerifierVLM confirms that the chair is visible and Candidate 13 is a suitable approach position, establishing a target observation waypoint. GrounderVLM then localizes the chair region, from which the depth observation yields a 3D target anchor for fine approach. At Step 5, the agent issues STOP because its planar distance to the target anchor is 0.83 m, below the stopping threshold.

We use each Ghost as a persistent record of an explorable candidate waypoint. NGTM associates observations of the same candidate across views. After an action, it updates the selected Ghost from measured motion and uses the new state to restrict the directions available at the next planning step.

Figure 1 illustrates the navigation process of GA-Nav. During target search, NGTM supplies PlannerVLM with the directions that remain available. When the target becomes visible, LVG selects and verifies a target observation waypoint and estimates a 3D target anchor from RGB-D. If target approach is interrupted after a target position has been established, the position is stored as a target-memory Ghost for subsequent search.

Our contributions are as follows:

- We introduce NGTM for execution-updated topological control in our training-free zero-shot ObjectNav framework. It tracks candidate identities and execution states across observations. After each action, measured motion updates the selected Ghost, and the stored states restrict the directions available to VLM planning.
- We develop LVG to select and verify a target observation waypoint and estimate a 3D target anchor for approach and stopping. A target-memory Ghost preserves an established target position after an interrupted approach.
- We evaluate GA-Nav on the complete validation sets of three ObjectNav benchmarks. The common-backbone comparison uses the same VLM backbone across methods, and a $2\times2\times2$ ablation shows that retaining cross-step NGTM provides the largest improvement within the fixed pipeline, with verification and 3D grounding contributing additional gains.

2 Related Work

2.1 Zero-Shot Object-Goal Navigation

The task definition and evaluation protocol of ObjectNav were systematized in prior work Batra et al. (2020). Learning-based methods acquire navigation policies from large-scale human demonstrations or offline visual representation learning Ramrakhya et al. (2022); Yadav et al. (2023b). Zero-shot studies use pretrained vision-language representations, open-vocabulary detection, and commonsense reasoning to transfer to new target categories and environments without task-specific ObjectNav training. ZSON and CoW establish target semantics through multimodal goal representations and open-vocabulary detection, respectively Majumdar et al. (2022); Gadre et al. (2023), while semantic-frontier methods introduce target-related knowledge into exploration Chen et al. (2023); Zhou et al. (2023); Yokoyama et al. (2024).

Zero-shot navigation methods use dense maps, topological structures, and image-space candidates as representations and action interfaces. OpenFMNav and SG-Nav construct an open-set map and an online 3D scene graph, respectively Kuang et al. (2024); Yin et al. (2024). VoroNav and TopV-Nav use topological or top-view representations to support direction selection Wu et al. (2024); Zhong et al. (2026), while PixelNav and VLMNav produce target positions or local actions from egocentric images Cai et al. (2024); Goetting et al. (2025).

2.2 VLM Planning with Navigation Memory

VLM-based navigation uses semantic reasoning to guide exploration and action selection Shah et al. (2023); Yu et al. (2023); Goetting et al. (2025); Long et al. (2025). Multi-step planning also needs a record of earlier observations and decisions. WMNav accumulates panoramic predictions in a Curiosity Value Map Nie et al. (2025), ReMemNav retrieves pose-indexed descriptions when a proposed direction overlaps an explored location Wu et al. (2026), and PanoNav maintains dynamic memory from panoramic scene parsing Jin et al. (2026).

Topological memory stores navigation history as places and connections Savinov et al. (2018), while frontier exploration identifies candidates at the boundary of observed free space Yamauchi (1997). Ghost nodes extend this structure with explorable candidates. Neural Topological SLAM predicts Ghost nodes for explorable areas, ETPNav merges predicted waypoints into Ghost nodes, and Frontier-enhanced Topological Memory generates Ghost nodes geometrically for visible unexplored regions Chaplot et al. (2020); An et al. (2025); Cui et al. (2024).

GA-Nav uses the Node–Ghost representation to constrain VLM planning. After an action toward a Ghost, NGTM updates its state from the measured pose change. Ghost states and edge traversal counts then determine the direction set passed to PlannerVLM. In contrast to direction values in WMNav or retrieved descriptions in ReMemNav, this memory directly controls which directions PlannerVLM can select.

2.3 Grounded Target Approach

Target approach requires more specific positional information than a search direction. PixelNav converts target positions in an image into navigation actions Cai et al. (2024), open-set visual maps combine detection and projection to store target locations Kuang et al. (2024), and multimodal foundation models can output target boxes and reference points Bai et al. (2025). GA-Nav uses the LVG procedure to convert a detected target direction into a verified target observation waypoint and then a 3D target anchor estimated from RGB-D. The waypoint guides the initial approach, while the anchor supports continued movement toward the target surface and the stopping decision.

3 Method

3.1 Task Definition

The zero-shot ObjectNav task requires an agent to explore unknown indoor environments without prior map information and navigate to any target instance within a given category g. At each time

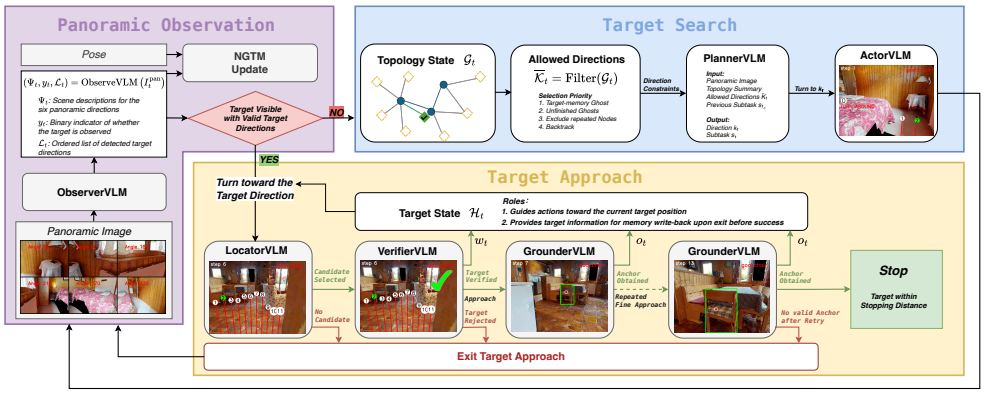

Figure 2: Overview of GA-Nav. The purple, blue, and yellow regions denote *panoramic observation*, *target search*, and *target approach*, respectively. Panoramic observation uses ObserverVLM to analyze six views and NGTM to update the online topology. Without a valid target direction, NGTM, PlannerVLM, and ActorVLM perform a topology-constrained search action before the next macro step returns to panoramic observation. With a valid target direction, the system enters target approach, where the Locate–Verify–Ground procedure establishes a target observation waypoint and a 3D target anchor. This stage continues until navigation succeeds or an exit condition is triggered. If the stage exits before success after a target position has been established, that position is written back to NGTM as a target-memory Ghost. The next macro step then returns to panoramic observation.

step t, the agent receives an RGB-D observation (I_t, D_t) and its pose $\boldsymbol{x}_t$. The agent then determines an action a_t using polar coordinates (r_t, θ_t), where θ_t denotes the relative yaw angle and r_t denotes the movement distance. The task is considered successful if the agent stops within a predefined distance threshold d_{succ} from the target.

3.2 Overview

Figure 2 summarizes the control flow. An active *target approach* continues without panoramic observation. Otherwise, *panoramic observation* updates the topology, followed by *target search* if ObserverVLM returns no target direction, or *target approach* otherwise. The VLM calls serve six functions: ObserverVLM, PlannerVLM, ActorVLM, LocatorVLM, VerifierVLM, and GrounderVLM.

During *panoramic observation*, the system acquires 12 RGB-D observations at 30° intervals and concatenates the images from six fixed directions into the panoramic input to ObserverVLM. Let $\mathcal{K}^{\text{pan}} = \{30^\circ, 90^\circ, 150^\circ, 210^\circ, 270^\circ, 330^\circ\}$ denote these directions and let I_t^{pan} denote the concatenated panoramic image. ObserverVLM outputs

$$(\Psi_t, y_t, \mathcal{L}_t) = \text{ObserverVLM}(I_t^{\text{pan}}), \qquad y_t \in \{0, 1\}. \tag{1}$$

In Eq. (1), Ψ_t contains scene descriptions for the six directions, y_t indicates whether the target is observed, and $\mathcal{L}_t$ is an ordered list of target directions drawn from $\mathcal{K}^{\text{pan}}$. When $y_t = 1$ and $\mathcal{L}_t \neq \varnothing$, the first direction in the list is denoted by k_t^{tar}.

After the panoramic analysis, NGTM updates the state of the Ghost selected at the preceding step from the measured pose change. It also registers or matches the current Node, records the traversed edge, and associates depth-derived candidates with Nodes or Ghosts. These elements form the updated online topology $\mathcal{G}_t$.

If ObserverVLM provides no target direction, NGTM filters the available directions from $\mathcal{G}_t$, PlannerVLM selects one, and ActorVLM chooses a depth-feasible local action. After execution, the next macro step updates the selected Ghost from the measured pose change.

If ObserverVLM provides a target direction, the agent turns toward k_t^{tar} and enters *target approach*. LVG selects and verifies a target observation waypoint, estimates a 3D target anchor, and approaches the target. If the approach ends after a target position has been established but before navigation succeeds, that position is stored as a target-memory Ghost for subsequent search.

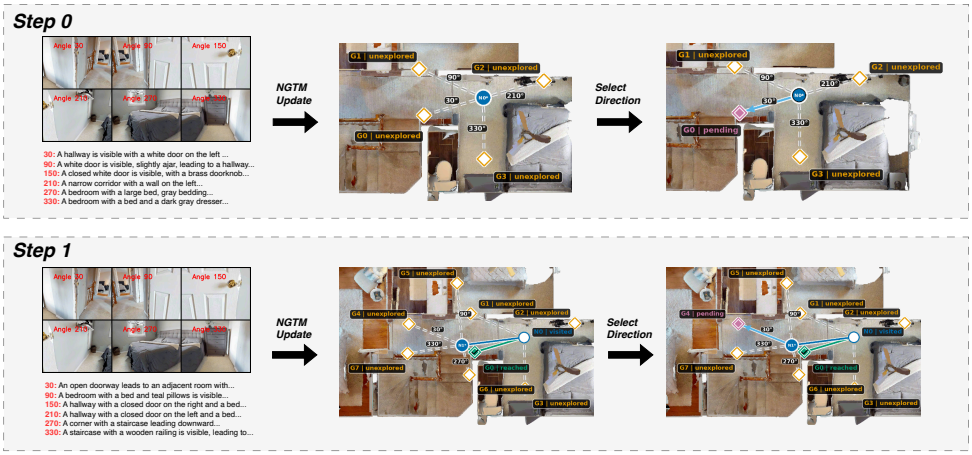

Figure 3: Cross-step construction and update of NGTM. At Step 0, panoramic scene descriptions and depth-derived candidates are associated with the current Node to establish Ghosts, and the selected Ghost changes from *unexplored* to *pending*. At Step 1, the motion outcome updates the selected Ghost, the reached position is registered as a new Node, and the new panoramic observation extends the topology with additional Ghosts before the next direction is selected.

3.3 Node–Ghost Topology Memory

Directions observed from different positions may point to the same visited location or candidate waypoint. NGTM represents locations reached by the agent as Nodes. Candidate waypoints that are not matched to Nodes are stored as Ghosts with cross-step execution states. The online topology at step t is defined as

$$\mathcal{G}_t = \left(\mathcal{V}_t^N \cup \mathcal{V}_t^G, \mathcal{E}_t\right), \tag{2}$$

where $\mathcal{V}_t^N$ and $\mathcal{V}_t^G$ denote the Node and Ghost sets, respectively. The edge set $\mathcal{E}_t$ contains Node–Node relations and Node–Ghost associations. A Node–Node relation can be established by observing a historical Node along the current direction or confirmed by physical traversal. Confirmed relations record traversal counts and support graph backtracking. Each Ghost stores its world position, topological associations, scene description, execution state, and attempt and failure counts. A target-memory Ghost additionally marks an established target position.

For a view in direction k, the depth module transforms each candidate waypoint into world coordinates using its estimated distance, relative direction, and the pose at which the view was acquired. Association follows a fixed Node-first order. A candidate first matches a Node by spatial proximity or by the direction of a traversed adjacent Node. If no Node matches, it is compared with Ghosts by spatial proximity, excluding those already marked *reached*; an unmatched candidate creates a new Ghost. Scene descriptions do not determine Node or Ghost identity. They are used to remove explicitly non-traversable directions and to provide context to PlannerVLM. For a Ghost that remains explorable, NGTM updates its scene description and source Nodes. This association identifies the same candidate across viewpoints and allows a failed Ghost to block that candidate when it is observed again. Figure 3 illustrates the update over two consecutive macro steps.

Ghost states are determined from pose changes before and after action execution. A new Ghost is added to the graph as *unexplored* and temporarily marked as *pending* when selected. For the current *pending* Ghost, let d_0 and d_1 be the distances to that Ghost before and after the action, m be the distance between the two agent positions, $\Delta d = d_0 - d_1$ be the decrease in distance, and n and n' be the failure counts before and after the update. The Ghost state is updated as

$$\sigma' = \begin{cases} \text{reached}, & d_1 \le \rho_{\text{reach}} \text{ or } \Delta d \ge \max\left(\epsilon_m, \frac{1}{2} d_0\right), \\ \text{failed}, & n' \ge N_{\text{fail}}, \\ \text{unexplored}, & \text{otherwise}. \end{cases} \tag{3}$$

Here ρ_{reach} is the reach distance, ϵ_m is the minimum displacement or distance decrease, and N_{fail} is the maximum failure count. If the current attempt does not satisfy the completion condition in the

first row and either $m < \epsilon_m$ or $\Delta d \leq 0$, then $n' = n + 1$; otherwise, $n' = n$. The *reached* state indicates that the agent has either approached the candidate waypoint or made sufficient progress along that direction to end the current attempt. A Ghost returns to *unexplored* when the action makes progress without satisfying the completion condition, and becomes *failed* after it has been blocked N_{fail} times. Once a Ghost becomes *reached*, the post-action agent location is registered or matched as a Node and connected to the preceding Node by the physically traversed edge.

3.4 Topology-Constrained Direction Planning and Local Action Selection

NGTM constructs the allowed direction set $\overline{\mathcal{K}}_t$ before PlannerVLM is called. It excludes directions described as non-traversable and directions matched to *failed* Ghosts. If a target-memory Ghost is visible, only its direction is retained. Otherwise, directions associated with unresolved Ghosts remain available, while Node–Node traversal counts suppress repeatedly used directions. If no direction is associated with a Ghost, the system backtracks along traversed Node–Node edges toward the nearest branch with an unresolved Ghost. If no executable direction remains, all six directions are made available as a fallback. We denote this procedure by Filter. PlannerVLM then selects a direction k_t and subtask s_t within the allowed set:

$$\begin{aligned} \overline{\mathcal{K}}_t &= \mathrm{Filter}(\mathcal{G}_t), \\ (k_t, s_t) &= \mathrm{PlannerVLM}(I_t^{\mathrm{pan}}, \tau(\mathcal{G}_t), s_{t-1}, \overline{\mathcal{K}}_t), \qquad k_t \in \overline{\mathcal{K}}_t. \end{aligned} \tag{4}$$

Here $\tau(\mathcal{G}_t)$ is the topology summary supplied to PlannerVLM, and s_t is a natural-language subtask describing the current search objective. The same subtask guides ActorVLM at step t and is passed to PlannerVLM as the previous subtask at step $t + 1$.

After the camera turns toward k_t, the depth module generates a set of polar actions $\mathcal{A}_t$ within the traversable region of the current view. Each action is projected onto the RGB image and assigned an index, producing the annotated image I_t^{ann}. ActorVLM selects an action index from the annotated image and subtask s_t:

$$i_t^{\mathrm{act}} = \mathrm{ActorVLM}(I_t^{\mathrm{ann}}, s_t), \qquad i_t^{\mathrm{act}} \in \{0, \ldots, |\mathcal{A}_t|\}. \tag{5}$$

In Eq. (5), a positive index i_t^{act} selects the corresponding element of $\mathcal{A}_t$ as the current action a_t, while index 0 denotes an in-place turn. The next observation updates the selected Ghost from the post-action pose according to Eq. (3). Figure 6 in Appendix B summarizes this selection process.

3.5 Locate–Verify–Ground Target-Approach Procedure

LVG first selects and verifies a target observation waypoint, then estimates a target anchor. We denote its current target reference by H_t:

$$H_t = \begin{cases} \varnothing, & \text{the target candidate has not been verified,} \\ \boldsymbol{w}_t, & \text{the target observation waypoint has been established,} \\ \boldsymbol{o}_t, & \text{the target anchor has been established.} \end{cases} \tag{6}$$

Here $\boldsymbol{w}_t$ and $\boldsymbol{o}_t$ denote the target observation waypoint and target anchor, respectively, and both belong to $\mathbb{R}^3$. VerifierVLM acceptance sets H_t to $\boldsymbol{w}_t$, and a valid target anchor replaces it with $\boldsymbol{o}_t$.

Target approach first selects and verifies a target observation waypoint. The agent turns toward the target direction k_t^{tar} provided by ObserverVLM. The depth module generates an indexed candidate set $\mathcal{C}_t$ within the traversable region of that view and projects the candidates onto the RGB image I_t^{loc}. LocatorVLM selects the index of a candidate near the target from the annotated image:

$$i_t^{\mathrm{loc}} = \mathrm{LocatorVLM}(I_t^{\mathrm{loc}}), \qquad i_t^{\mathrm{loc}} \in \{0, \ldots, |\mathcal{C}_t|\}. \tag{7}$$

In Eq. (7), $i_t^{\mathrm{loc}} = 0$ indicates that LocatorVLM does not select a candidate. For $i_t^{\mathrm{loc}} > 0$, VerifierVLM checks whether the target is visible and the selected candidate is a suitable approach position. If verification succeeds, the system transforms the selected candidate into world coordinates to obtain the target observation waypoint $\boldsymbol{w}_t$, and updates H_t to $\boldsymbol{w}_t$. Otherwise, it clears the current candidate and terminates the current *target approach*.

After the candidate is verified, the system moves toward $\boldsymbol{w}_t$ in *approach*. In the next unannotated RGB observation, GrounderVLM localizes the target region and provides a reference point within

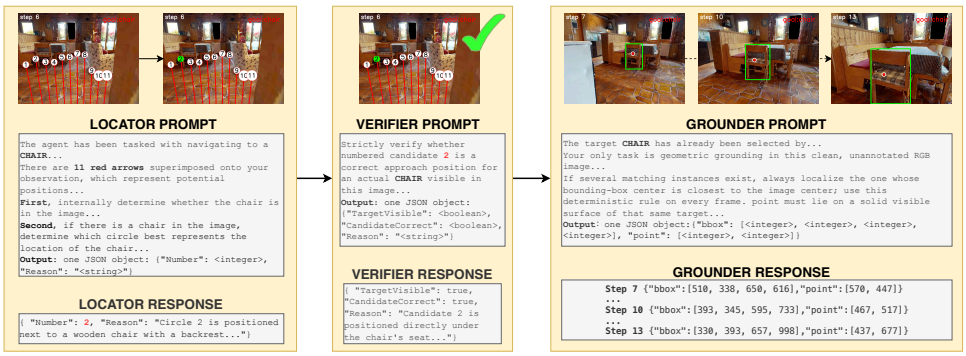

Figure 4: Locate–Verify–Ground in *target approach*. LocatorVLM selects Candidate 2 from the depth-derived candidate waypoints projected onto the target view, and VerifierVLM confirms that the target is visible and Candidate 2 is a suitable approach position. GrounderVLM then localizes the target region and an interior reference point in subsequent unannotated observations, enabling depth-based estimation and update of the 3D target anchor.

it. The depth module extracts a depth-continuous set of target pixels Ω_t from that point. Their 3D positions define one target-anchor observation:

$$\widehat{\boldsymbol{o}}_t = \mathcal{T}_t\left(\underset{(u,v)\in\Omega_t}{\mathrm{cmed}} \Pi_K^{-1}(u, v, D_t(u, v))\right). \tag{8}$$

In Eq. (8), Π_K^{-1} back-projects pixel (u, v) using depth $D_t(u, v)$ and camera intrinsics K. The operator cmed takes the coordinate-wise median of the resulting 3D points, and $\mathcal{T}_t$ maps it to the world coordinate frame. If GrounderVLM returns no valid target region or Ω_t contains insufficient valid depth, then $\widehat{\boldsymbol{o}}_t = \varnothing$. The system turns toward H_t and retries in the next observation; a second failure ends the current *target approach*. Otherwise, recent matched anchor observations are fused by a coordinate-wise median to obtain $\boldsymbol{o}_t$. The system then sets $H_t = \boldsymbol{o}_t$ and switches to *fine approach*. Appendix C gives the validity, association, and fusion rules. Figure 4 illustrates the waypoint verification and subsequent target-region localization.

During *fine approach*, the system adjusts its orientation and approaches the target surface with short-range actions. Let $\boldsymbol{p}_t$ denote the current agent position and d_{nav} the distance on the navigation plane. The stopping condition is

$$\textsc{Stop} \iff d_{\mathrm{nav}}(\boldsymbol{p}_t, H_t) \le \delta_o. \tag{9}$$

Here δ_o is the stopping-distance threshold.

The system also terminates the current target approach if neither of two consecutive translation actions reduces the agent's remaining distance to H_t by at least 0.1 m. If target approach terminates before navigation succeeds and a target position has already been established, the system creates a target-memory Ghost from H_t, allowing the corresponding direction to be prioritized in subsequent *target search*.

4 Experiments

4.1 Datasets and Evaluation Metrics

Datasets. We evaluate on the ObjectNav validation sets of HM3D Ramakrishnan et al. (2021) and Matterport3D (MP3D) Chang et al. (2017) in Habitat Savva et al. (2019). HM3D v0.1 and MP3D follow the public ObjectNav benchmark splits used by SG-Nav Yin et al. (2024). HM3D v0.1 contains 20 validation scenes and 2,000 episodes over 6 target categories, while MP3D contains 11 validation scenes and 2,195 episodes over 21 target categories. HM3D v0.2 uses the Habitat Navigation Challenge 2023 validation split, which contains 36 scenes and 1,000 episodes Yadav et al. (2023a).

Evaluation metrics. We report success rate (SR) and success weighted by path length (SPL). For N episodes, $\mathrm{SR} = \frac{1}{N}\sum_{i=1}^{N} S_i$, where $S_i \in \{0, 1\}$ indicates whether episode i succeeds. SPL measures both task success and path efficiency and is defined as $\mathrm{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{\ell_i}{\max(p_i, \ell_i)}$, where ℓ_i is the reference shortest geodesic distance from the initial location to the nearest valid target instance and p_i is the agent's actual path length.

Complete implementation and hyperparameter settings are provided in Appendix E.

4.2 Comparison with Existing Methods

Table 1 compares GA-Nav with learning-based and training-free zero-shot ObjectNav methods on MP3D, HM3D v0.1, and HM3D v0.2. Baseline entries use results reported in their cited papers, and the table distinguishes training-free (TF) and zero-shot (ZS) settings. GA-Nav uses Qwen3-VL-32B-Instruct, and its results are obtained on the complete validation sets of all three datasets.

Table 1: Comparison with existing ObjectNav methods on HM3D and MP3D. TF and ZS denote training-free and zero-shot settings, respectively. A dash indicates a result not reported in the original paper. Bold and underlined values denote the best and second-best results among training-free zero-shot methods, respectively.

Method	TF	ZS	MP3D		HM3D v0.1		HM3D v0.2	
			SR (%)↑	SPL (%)↑	SR (%)↑	SPL (%)↑	SR (%)↑	SPL (%)↑
Habitat-Web Ramrakhya et al. (2022)	×	×	31.6	8.5	41.5	16.0	–	–
OVRL Yadav et al. (2023b)	×	×	28.6	7.4	–	–	–	–
ZSON Majumdar et al. (2022)	×	✓	15.3	4.8	25.5	12.6	–	–
PixNav Cai et al. (2024)	×	✓	–	–	37.9	20.5	–	–
PSL Sun et al. (2024)	×	✓	18.9	6.4	42.4	19.2	–	–
SGM Zhang et al. (2024)	×	✓	37.7	14.7	60.2	30.8	–	–
VLFM Yokoyama et al. (2024)	×	✓	36.4	17.5	52.5	30.4	62.6	31.0
Chain-of-Search Chen et al. (2026)	×	✓	37.6	17.6	55.9	29.1	–	–
CoW Gadre et al. (2023)	✓	✓	9.2	4.9	–	–	–	–
ESC Zhou et al. (2023)	✓	✓	28.7	14.2	39.2	22.3	–	–
L3MVN Yu et al. (2023)	✓	✓	–	–	50.4	23.1	36.3	15.7
TopV-Nav Zhong et al. (2026)	✓	✓	35.2	16.4	53.0	29.8	–	–
SG-Nav Yin et al. (2024)	✓	✓	40.2	16.0	54.0	24.9	49.6	25.5
VLMNav Goetting et al. (2025)	✓	✓	–	–	50.4	21.0	–	–
InstructNav Long et al. (2025)	✓	✓	–	–	58.0	20.9	–	–
UniGoal Yin et al. (2025)	✓	✓	41.0	16.4	54.5	25.1	–	–
MFNP Zhang et al. (2025)	✓	✓	41.1	15.4	<u>58.3</u>	26.7	–	–
WMNav Nie et al. (2025)	✓	✓	<u>45.4</u>	17.2	58.1	31.2	<u>72.2</u>	<u>33.3</u>
PanoNav Jin et al. (2026)	✓	✓	–	–	43.5	23.7	–	–
EIK-Nav Jiang et al. (2026)	✓	✓	37.1	<u>17.4</u>	54.8	<u>31.5</u>	–	–
GA-Nav (Ours)	✓	✓	**49.5**	**24.9**	**58.5**	**35.6**	**73.1**	**39.1**

Among the training-free zero-shot methods in Table 1, GA-Nav ranks first in SR and SPL on all three datasets. Relative to the best previous result for each metric, its SR/SPL gains are $4.1/7.5$ percentage points on MP3D, $0.2/4.1$ points on HM3D v0.1, and $0.9/5.8$ points on HM3D v0.2.

Among these baselines, WMNav Nie et al. (2025) also combines panoramic observations, persistent memory, and separate search and approach stages. Compared with WMNav, GA-Nav improves SR/SPL by $4.1/7.7$ percentage points on MP3D, $0.4/4.4$ points on HM3D v0.1, and $0.9/5.8$ points on HM3D v0.2. The two methods differ in how memory enters planning: WMNav uses predicted target likelihoods stored in a Curiosity Value Map to select directions, while NGTM uses candidate execution states to restrict available directions. Their approach procedures also differ. WMNav selects a target location from navigable positions, whereas LVG verifies an observation waypoint and then estimates a target anchor from RGB-D for approach and stopping.

When task-trained policies are included, GA-Nav still ranks first in SR and SPL on MP3D and HM3D v0.2. On HM3D v0.1, it achieves the highest SPL among the listed methods, exceeding SGM by 4.8 points, while its SR is 1.7 points lower Zhang et al. (2024).

4.3 Comparison under a Common VLM Backbone

Because navigation systems use VLMs with different capabilities, comparisons of published results reflect both the navigation method and its backbone. Table 2 therefore compares GA-Nav, VLMNav, UniGoal, InstructNav, and ReMemNav with Qwen3-VL-4B on HM3D v0.2. The four baseline results are taken from ReMemNav Wu et al. (2026).

With Qwen3-VL-4B, GA-Nav obtains an SR of 65.7% and an SPL of 29.4%. It exceeds the four baselines in Table 2 by 9.5–24.2 SR points and 7.6–17.7 SPL points.

Table 2: Comparison with Qwen3-VL-4B on HM3D v0.2. Bold and underlined values denote the best and second-best results, respectively.

Method	SR (%)↑	SPL (%)↑
VLMNav	41.5	11.7
UniGoal	43.3	21.7
InstructNav	53.5	21.1
ReMemNav	<u>56.2</u>	<u>21.8</u>
GA-Nav (Ours)	**65.7**	**29.4**

4.4 Module Ablation

We ablate the cross-step persistence of NGTM, candidate verification, and target anchoring on all 1,000 HM3D v0.2 episodes with Qwen3-VL-32B-Instruct. The variants use the same episode IDs and form a $2 \times 2 \times 2$ design. In the single-step NGTM setting, the system clears historical Nodes, Ghosts, and edges before each macro-level planning step while retaining the topology and candidate constraints from the current observation. The other two factors enable or disable VerifierVLM and GrounderVLM with *fine approach*. When both are disabled, the system approaches only the target observation waypoint selected by LocatorVLM.

Table 3: Module and memory ablations on HM3D v0.2 using Qwen3-VL-32B-Instruct. "Single-step NGTM" retains only the topology constructed from the current observation at each planning step. VerifierVLM verifies the target observation waypoint. GrounderVLM denotes target grounding and 3D-anchor-based *fine approach*. Bold and underlined values denote the best and second-best results, respectively.

	Memory	VerifierVLM	GrounderVLM	SR (%)↑	SPL (%)↑
a	Single-step NGTM	×	×	63.1	31.5
b	Single-step NGTM	✓	×	63.1	30.6
c	Single-step NGTM	×	✓	64.1	33.8
d	Single-step NGTM	✓	✓	65.6	33.8
e	Cross-step NGTM	×	×	70.7	36.0
f	Cross-step NGTM	✓	×	<u>71.9</u>	36.5
g	Cross-step NGTM	×	✓	71.6	<u>38.5</u>
h	Cross-step NGTM	✓	✓	**73.1**	**39.1**

Across all four combinations of target verification and anchoring, clearing the historical topology reduces SR by 7.5–8.8 percentage points and SPL by 4.5–5.9 percentage points. With cross-step NGTM retained, removing VerifierVLM reduces SR/SPL by $1.5/0.6$ points, while removing GrounderVLM and *fine approach* reduces them by $1.2/2.6$ points. Removing both reduces SR/SPL by $2.4/3.1$ points. Cross-step NGTM therefore has the largest measured effect in this fixed pipeline, while candidate verification and target anchoring provide additional gains.

5 Conclusion

GA-Nav maintains persistent Ghosts for depth-derived candidate waypoints and updates their states from executed motion before the next VLM decision. Across four combinations of target verification and anchoring, retaining cross-step NGTM improves SR by 7.5–8.8 percentage points and SPL by 4.5–5.9 percentage points over the corresponding single-step NGTM variants. LVG adds verified waypoint selection and 3D target anchoring for target approach. The full system achieves 73.1% SR and 39.1% SPL on HM3D v0.2 and ranks first in SR and SPL among the training-free zero-shot methods in Table 1 on all three benchmarks.

AI USE STATEMENT

In this study, we used generative AI to assist with code generation, literature retrieval, and manuscript drafting, polishing, and formatting. The authors reviewed and tested all AI-generated code and verified the AI-assisted manuscript content against the implementation, experimental records, and cited sources. The authors take responsibility for the final content of this paper.

REFERENCES

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. ETPNav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(7):5130–5145, 2025.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. ObjectNav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

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

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. In *2017 International Conference on 3D Vision (3DV)*, pp. 667–676, 2017.

Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological SLAM for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12875–12884, 2020.

Hanrui Chen, Liqi Yan, Qifan Wang, Jianhui Zhang, Fangli Guan, and Pan Li. Chain-of-Search: Parameter-efficient reasoning for zero-shot object navigation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(3):1721–1729, 2026.

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

Xinru Cui, Qiming Liu, Zhe Liu, and Hesheng Wang. Frontier-enhanced topological memory with improved exploration awareness for embodied visual navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 296–313, 2024.

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

Dylan Goetting, Himanshu Gaurav Singh, and Antonio Loquercio. End-to-end navigation with vision-language models: Transforming spatial reasoning into question-answering. In *Proceedings of the International Conference on Neuro-symbolic Systems*, volume 288, pp. 22–35, 2025.

Botao Jiang, Haonan Luo, Zihang Wang, Jiahe Li, Hang Li, Jin Zheng, and Xiao Bai. EIK-Nav: Boosting zero-shot object navigation with explicit and implicit knowledge. *Pattern Recognition*, 174:112945, 2026.

Qunchao Jin, Yilin Wu, and Changhao Chen. PanoNav: Mapless zero-shot object navigation with panoramic scene parsing and dynamic memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18351–18359, 2026.

Yuxuan Kuang, Hai Lin, and Meng Jiang. OpenFMNav: Towards open-set zero-shot object navigation via vision-language foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 338–351, 2024.

Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. InstructNav: Zero-shot system for generic instruction navigation in unexplored environment. In *Proceedings of the 8th Conference on Robot Learning*, volume 270, pp. 2049–2060, 2025.

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

Dujun Nie, Xianda Guo, Yiqun Duan, Ruijun Zhang, and Long Chen. WMNav: Integrating vision-language models into world models for object goal navigation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2392–2399, 2025.

Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M. Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-Matterport 3D dataset (HM3D): 1000 large-scale 3D environments for embodied AI. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-Web: Learning embodied object-search strategies from human demonstrations at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5173–5183, 2022.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9339–9347, 2019.

Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Proceedings of the 7th Conference on Robot Learning*, volume 229, pp. 2683–2699, 2023.

Xinyu Sun, Lizhao Liu, Hongyan Zhi, Ronghe Qiu, and Junwei Liang. Prioritized semantic learning for zero-shot instance navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 161–178, 2024.

Feng Wu, Wei Zuo, Wenliang Yang, Jun Xiao, Yang Liu, and Xinhua Zeng. ReMemNav: Memory-based decision correction and target verification for zero-shot object navigation. *arXiv preprint arXiv:2603.26788*, 2026.

Pengying Wu, Yao Mu, Bingxian Wu, Yi Hou, Ji Ma, Shanghang Zhang, and Chang Liu. VoroNav: Voronoi-based zero-shot object navigation with large language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 53757–53775, 2024.

Karmesh Yadav, Jacob Krantz, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Jimmy Yang, Austin Wang, John Turner, Aaron Gokaslan, Vincent-Pierre Berges, Roozbeh Mottaghi, Oleksandr Maksymets, Angel X. Chang, Manolis Savva, Alexander Clegg, Devendra Singh Chaplot, and Dhruv Batra. Habitat challenge 2023, 2023a. URL `https://aihabitat.org/challenge/2023/`.

Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. In *Workshop on Reincarnating Reinforcement Learning at ICLR*, 2023b.

Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pp. 146–151, 1997.

Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. SG-Nav: Online 3D scene graph prompting for LLM-based zero-shot object navigation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 5285–5307, 2024.

Hang Yin, Xiuwei Xu, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. UniGoal: Towards universal zero-shot goal-oriented navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19057–19066, 2025.

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

Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3MVN: Leveraging large language models for visual target navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3554–3560, 2023.

Lingfeng Zhang, Hao Wang, Erjia Xiao, Xinyao Zhang, Qiang Zhang, Zixuan Jiang, and Renjing Xu. Multi-Floor zero-shot object navigation policy. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6416–6422, 2025.

Sixian Zhang, Xinyao Yu, Xinhang Song, Xiaohan Wang, and Shuqiang Jiang. Imagine before go: Self-supervised generative map for object goal navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16414–16425, 2024.

Linqing Zhong, Chen Gao, Zihan Ding, Yue Liao, Huimin Ma, Shifeng Zhang, Xu Zhou, and Si Liu. TopV-Nav: Unlocking the top-view spatial reasoning potential of MLLM for zero-shot object navigation. *International Journal of Computer Vision*, 134(9), 2026. Article 409.

Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. ESC: Exploration with soft commonsense constraints for zero-shot object navigation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 42829–42842, 2023.

A COMPLETE CONTROL ALGORITHM

Algorithm 1 is organized around the target decision of ObserverVLM. When no target approach is active, ObserverVLM first examines the panorama. A missing target direction enters Branch 1 for topology-guided search, whereas a valid target direction enters Branch 2 for LVG target approach. Once Branch 2 establishes a target hypothesis, subsequent steps remain in that branch without another panoramic observation until the approach stops or exits.

Algorithm 1: Complete GA-Nav control loop
Input: goal category g, macro-step budget T, RGB-D and pose observations.
State: NGTM graph $\mathcal{G}$, pending Ghost transition, target hypothesis H, and target-control state q.

1. Initialize $\mathcal{G}$ and H as empty, clear the pending Ghost transition, and set $q = \textit{inactive}$.
2. At each macro step $t = 1, \ldots, T$, acquire the current RGB-D observation and pose, then determine the entry:
   (a) if H is active, skip panoramic observation and continue in Branch 2;
   (b) otherwise, acquire 12 panoramic views, call ObserverVLM on the six labeled directions, resolve the pending Ghost from measured motion, update the current Node and traversed edge, and associate the direction candidates;
   (c) if ObserverVLM returns no valid target direction, enter Branch 1; otherwise, enter Branch 2 with its first valid target direction.
3. **Branch 1: ObserverVLM returns no valid target direction.**
   (a) Construct the allowed direction set from the updated NGTM, then call PlannerVLM to select a direction and subtask within that set.
   (b) Align the camera with the selected direction, generate projected local actions from depth, and let ActorVLM select one action under the subtask.
4. **Branch 2: ObserverVLM returns a valid target direction, or H remains active from the preceding step.**
   (a) **If entered from ObserverVLM:** select its first target direction and skip PlannerVLM. After camera alignment, generate projected local actions and indexed navigable candidates for LocatorVLM. Validate and associate the selected candidate, then call VerifierVLM if the waypoint has not been established. An accepted waypoint activates H, sets $q = \textit{approach}$, and schedules GrounderVLM for the next observation.
   (b) **If entered with an active H:** retain the current view and target subtask. Call GrounderVLM on the clean RGB image, obtain a 3D anchor from the validated depth component, and update the measured progress toward H.
   (c) **Update target control.** A valid anchor sets $q = \textit{fine approach}$, and the first unresolved grounding call sets $q = \textit{retry}$. Verification rejection clears H. A second unresolved grounding call or two low-progress translations stores the accepted target position as a target-memory Ghost when available, then clears H and sets $q = \textit{inactive}$.
   (d) **Dispatch one action.** Apply the first matching rule:
      i. if the LVG stop conditions hold, output STOP;
      ii. if $q = \textit{retry}$, rotate toward the stored waypoint or anchor;
      iii. if a continued target approach has become inactive, output a no-op so that the next step begins with panoramic observation;
      iv. if $q \in \{\textit{approach}, \textit{fine approach}\}$, move toward the waypoint or anchor with the corresponding translation cap;
      v. otherwise, let ActorVLM select a projected local action. This case occurs when a newly detected target direction does not establish a waypoint.
5. If either branch used ActorVLM, mark the selected Ghost as *pending* for a translational action and store its pre-action position and distance. Record a failed attempt when ActorVLM returns a non-translational or invalid selection for that Ghost.
6. Execute the selected action. Terminate on STOP. Otherwise, the next step continues Branch 2 if H remains active and returns to panoramic observation if H is inactive.



Figure 5: Complete timeline of a successful toilet-navigation episode on HM3D v0.1. Each panel shows one macro step, and the arrows indicate the execution order. Steps 0–10 show target search. At Step 11, ObserverVLM detects the target direction, LocatorVLM selects Candidate 6, and VerifierVLM accepts it. Steps 12–20 show target grounding and approach, and Step 21 terminates the episode successfully.

B NGTM Details

B.1 Candidate Construction and Stored State

NGTM constructs one topological candidate for each of the six panoramic directions when depth provides a navigable ray. Within a directional view, the depth module samples 120 rays over ± 1.5 times the camera half field of view. A ray with range r and bearing θ is ranked by $\min(r, 1.7) - 0.2|\theta|$, which favors long paths near the view center. The selected candidate is placed at $\min(0.8r, 1.7)$ m along that ray. Candidates shorter than 0.6 m are discarded. The candidate is transformed to world coordinates using the pose recorded for that panoramic view.

A Node stores its world position, visit count, parent Node, and the scene description selected when the location was reached. A Ghost stores its world position, parent Node, all source Nodes from which it has been observed, the latest scene description, its execution state, and attempt and failure counts. Ghost states are *unexplored*, *pending*, *reached*, and *failed*. A target-memory Ghost additionally stores a target flag. Node–Node edges record observed or traversed connections, while Node–Ghost edges follow the state of their Ghost.

Scene descriptions are used before association to remove directions that explicitly contain one of the following non-traversability phrases: `closed door`, `solid wall`, `blocked path`, `cannot be explored`, `not navigable`, `no passage`, or `no way forward`. Other descriptions are stored with live Ghosts and included in the topology summary given to PlannerVLM. They do not enter the Node or Ghost identity test.

B.2 Node and Ghost Association

Association follows a fixed Node-first order. A candidate first matches an existing Node when their world-coordinate distance is below 0.50 m. The current Node is excluded from this test. If no Node lies within this radius, NGTM compares the candidate direction with Nodes connected to the current

Node by traversed edges. A connected Node matches when the angular difference is at most $40°$; the match with the smallest angular difference is selected, with distance used to break a tie.

If no Node matches, the candidate is compared with existing Ghosts other than those already marked *reached*. A Ghost within 0.75 m is reused. A matched *failed* Ghost blocks the current direction. For any other matched Ghost, NGTM replaces its scene description with the current one, adds the current Node to its source-Node list, and records the current Node–Ghost association. If no Ghost matches, a new *unexplored* Ghost is created at the candidate position. Thus identity association uses position and traversed adjacency, while scene descriptions affect traversability filtering and PlannerVLM context.

B.3 Execution-Updated Ghost State

When ActorVLM selects a translational action associated with a Ghost, NGTM changes that Ghost from *unexplored* to *pending*, increments its attempt count, and stores the pre-action position and distance d_0. At the next panoramic observation, the measured post-action position gives the translation m, remaining distance d_1, and progress $\Delta d = d_0 - d_1$. The update follows Eq. (3). In the reported configuration, $\rho_{\text{reach}} = 0.65$ m, $\epsilon_m = 0.25$ m, and $N_{\text{fail}} = 2$.

A pending Ghost becomes *reached* if $d_1 \leq 0.65$ m or $\Delta d \geq \max(0.25, 0.5d_0)$. Otherwise, a failed attempt is recorded when $m < 0.25$ m or $\Delta d \leq 0$. The Ghost returns to *unexplored* after its first failed attempt and becomes *failed* after the second. Motion that makes positive progress without satisfying the reached condition returns the Ghost to *unexplored* without increasing its failure count. If ActorVLM returns a non-translational or invalid selection for the chosen Ghost, that selection is recorded as a failed attempt. Once a Ghost is reached, the post-action position is registered as a Node, and the resulting Node–Node connection is recorded as traversed.

B.4 Allowed Directions and Backtracking

NGTM maps the six panoramic directions to option numbers $\{1, 3, 5, 7, 9, 11\}$. Directions marked by a non-traversability description or a *failed* Ghost are excluded. If a current direction matches a target-memory Ghost, that option alone is passed to PlannerVLM. Otherwise, Ghost directions remain available. When an *unexplored* Ghost is visible, a Node direction whose traversed-edge count has reached two is removed before planning.

If the current scan contains no Ghost direction, breadth-first search over traversed Node–Node edges finds the first hop toward the nearest historical Node that still owns an *unexplored* or *pending* Ghost. If no such path is found, NGTM uses the parent Node; if that is unavailable, it chooses the visible Node edge with the smallest traversal count. When no graph direction can be executed, all six panoramic options are exposed as a visual fallback. PlannerVLM must return an option within the resulting set. An out-of-set or unparsable response is replaced by the first allowed option and a neutral search subtask.

C LVG Details

C.1 Call Schedule and Waypoint Verification

LVG begins when ObserverVLM returns `target_visible=true` together with at least one valid panoramic direction. The first returned direction fixes the macro direction, so PlannerVLM is not called in that step. The camera turns to this view, and the depth module projects numbered navigable positions onto the RGB image for LocatorVLM.

LocatorVLM returns a numbered position or 0. A positive result is accepted only when the number exists in the rendered candidate set. The corresponding polar candidate is transformed into a world-coordinate target observation waypoint. If an active waypoint already exists, a new observation is associated only when its planar distance from the stored waypoint is at most 1.0 m. Up to the four preceding associated observations and the current observation are fused by a coordinate-wise median.

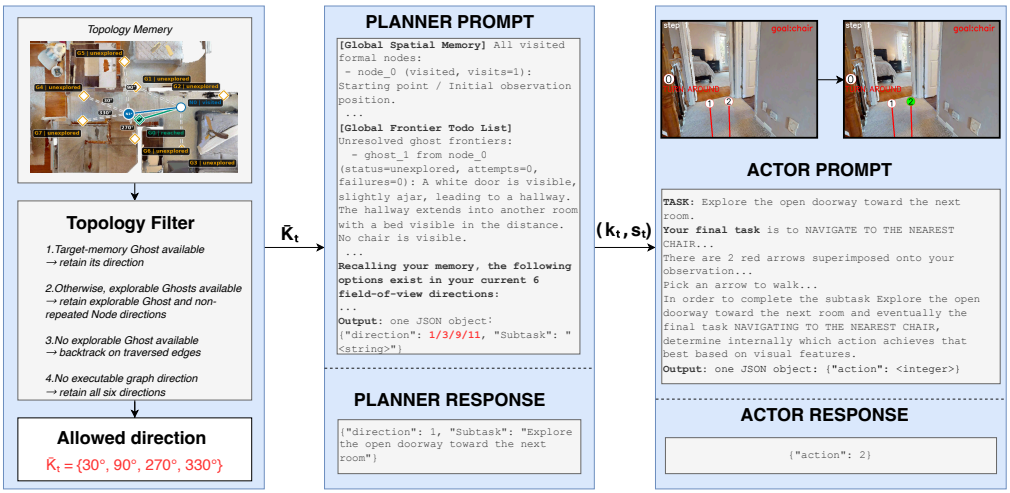

Figure 6: Topology-constrained direction and action selection during *target search*. NGTM constructs the allowed direction set from the online topology. PlannerVLM jointly selects a direction and a subtask from the panoramic observation and topology summary. After the camera aligns with the selected direction, ActorVLM selects a depth-feasible action for the subtask.

VerifierVLM receives the same annotated image and the Locator-selected index. It can only accept or reject that index. Verification is called when the candidate number is valid, the candidate is spatially consistent with the active waypoint, and the current target hypothesis has not already been verified. A rejected candidate clears the target hypothesis. Once accepted, verification remains valid for the lifetime of that target hypothesis and is not repeated at every approach step.

LocatorVLM and GrounderVLM are not called in the same macro step. The first accepted target observation waypoint triggers an *approach* action of at most 0.40 m and schedules GrounderVLM for the next observation. This separation gives GrounderVLM a clean, unannotated RGB image after the agent has moved toward the selected observation waypoint.

C.2 Depth Component and Target Anchor

GrounderVLM returns exactly a bounding box and a reference point. Qwen3-VL uses coordinates normalized to $[0, 1000]$ on both image axes. The output is accepted only if it contains exactly the keys `bbox` and `point`, the box has positive area and lies within the coordinate range, and the point lies inside the box. The coordinates are then mapped to the 640×480 depth image.

Within the predicted box, the depth at the reference point is used as the seed when valid. Otherwise, the seed is the median valid depth in a 9×9 neighborhood; if that neighborhood contains no valid depth, the nearest valid pixel in the box is used. For seed depth z_s, the depth tolerance is $\min(0.50, \max(0.20, 0.10 z_s))$ m. Pixels within this tolerance undergo a 3×3 morphological closing and 8-connected-component labeling. LVG selects the component containing the reference point, or the nearest component when the point itself has invalid depth. Components with fewer than eight pixels are rejected.

Every selected pixel is back-projected into the camera frame using the camera intrinsics. The coordinate-wise median of these 3D points is then transformed to the world frame using the depth-sensor pose to obtain one target-anchor observation. An observation is associated with the active target anchor when their planar distance is at most 0.75 m. The fused anchor is the coordinate-wise median of the current observation and up to four preceding associated observations.

C.3 APPROACH, RETRY, AND STOPPING

The target controller has four internal states: *inactive*, *approach*, *fine approach*, and *retry*. After waypoint verification, *approach* rotates toward the observation waypoint and translates by at most 0.40 m. A valid target-anchor observation changes the state to *fine approach*; translation is then capped at 0.25 m. GrounderVLM is called on subsequent target-approach observations to update the anchor.

If GrounderVLM or depth processing does not yield a valid anchor, LVG enters *retry* and rotates once toward the stored observation waypoint or anchor. A second unresolved grounding attempt ends direct target approach. The same exit occurs after two translational actions whose reduction in remaining target distance is less than 0.10 m. When a verified target position exists at exit, it is stored as a target-memory Ghost. The anchor is used when available; otherwise, the observation waypoint is stored. A target-memory Ghost within 1.0 m is reused instead of creating a duplicate.

In the full method, STOP requires an established target anchor, an accepted target waypoint, the *fine approach* state, and a planar agent-to-anchor distance of at most 0.90 m. If these conditions are not met, control remains in target approach or returns to panoramic observation through the rules above.

D COMPLETE VLM PROMPTS

The six module prompts are reproduced below. Angle brackets denote runtime substitutions. <TARGET> and <TARGET_UPPER> hold the goal name with the indicated capitalization. The remaining fields contain the current topology, subtask, candidate counts, and allowed options. <SEPARATOR> is a space when the subtask already ends in punctuation and ". " otherwise. Line wrapping is only for typesetting. ObserverVLM, PlannerVLM, LocatorVLM, VerifierVLM, and GrounderVLM receive the listed text as a user message. ActorVLM additionally receives the system instruction shown before its task prompt.

D.1 OBSERVERVLM

```
The agent has been tasked with navigating to a <TARGET_UPPER>. The agent has sent you the
     panoramic image describing your surrounding environment, each image contains a label
     indicating the relative rotation angle(30, 90, 150, 210, 270, 330) with red fonts. Your
     job is to provide a brief semantic description for each direction, judging whether this
     direction is worth exploring and what objects are visible. To help you describe the
     layout of your surrounding, please follow my instructions: (1) Describe if there is a
     way to move to another area (corner, open door, hallway, etc.). (2) Describe if the
     target <TARGET> or any related clues are visible. (3) Identify dead ends or closed doors
     that cannot be explored. Note you CANNOT GO THROUGH CLOSED DOORS, and you DO NOT NEED TO
     GO UP OR DOWN STAIRS. Return only one JSON object with an explicit target decision and
     all six descriptions. It must contain lowercase "target_visible" with a JSON boolean
     value, lowercase "target_directions" with an array of visible-target angles, and
     lowercase "directions" with a JSON object whose exact string keys are "30", "90", "150",
     "210", "270", and "330", each mapped to a description string. Only set
     target_visible=true when the actual <TARGET> is physically visible, not merely likely to
     exist in that room.
```

D.2 PLANNERVLM

```
Your ultimate task is to NAVIGATE TO THE NEAREST <TARGET_UPPER>.
You are currently standing at the visited node: <CURRENT_NODE>. You see a panoramic view with
     6 direction markers (Option Numbers 1, 3, 5, 7, 9, 11).
Previous subtask: <PREVIOUS_SUBTASK>.

[Global Spatial Memory] All visited formal nodes:
<NODE_HISTORY>

[Global Frontier Todo List] Unresolved ghost frontiers:
<FRONTIER_HISTORY>

Recalling your memory, the following options exist in your current 6 field-of-view directions:
<CURRENT_TOPOLOGY>

Please make a decision based on the above [Node/Ghost Node Information] and image content:
1. Prioritize exploring the [Unexplored Ghost Node] or [Known Ghost Node] whose semantic
     description is most relevant to finding the <TARGET_UPPER>.
2. Generally avoid historic [Visited Node Re-entry] paths that have been "traversed 1 or more
     times", encouraging extension into unknown areas.
3. [BREAKOUT STRATEGY]: If there is no valid Ghost Frontier, select a [Visited Node Re-entry]
     to return toward the nearest unresolved historical frontier. A deterministic graph
     fallback will enforce this when needed.
4. The panoramic target detector did not provide an actionable target direction. Treat
     target-related descriptions only as clues for choosing the next exploration subgoal; do
     not declare that the target has been found.
Output only one JSON object containing a lowercase "direction" key with a bare integer option
     number from <ALLOWED_OPTIONS> and an exact-case "Subtask" key with a string value.
Example: if you select Direction Angle <EXAMPLE_ANGLE>° (Option <EXAMPLE_OPTION>), output
     {"direction": <EXAMPLE_OPTION>, "Subtask": "Explore the open doorway toward the next
     room"}. This example demonstrates format only; do not copy Option <EXAMPLE_OPTION>
     unless you actually select it. Use the numeric Option number for direction; never output
     an angle label such as "Angle <EXAMPLE_ANGLE>".
```

D.3 ACTORVLM

System instruction

```
You are an embodied robotic assistant, with an RGB image sensor. You observe the image and
     instructions given to you and output a textual response, which is converted into actions
     that physically move you within the environment. You cannot move through closed doors.
```

Task prompt

```
TASK: <SUBTASK><SEPARATOR>Your final task is to NAVIGATE TO THE NEAREST <TARGET_UPPER>, and
     get as close to it as possible. There are <NUM_ACTION_ARROWS> red arrows superimposed
     onto your observation, which represent potential actions. These are labeled with a
     number in a white circle, which represent the location you would move to if you took
     that action. <TURNAROUND_NOTE>In order to complete the subtask <SUBTASK> and eventually
     the final task NAVIGATING TO THE NEAREST <TARGET_UPPER>, determine internally which
     action achieves that best based on visual features. Return exactly one JSON object with
     no text outside it. The object must contain the lowercase "action" key with a bare
     integer value. Format example only: if you select the visible arrow labeled 2, output
     {"action": 2}. Do not copy the example number unless you actually select that visible
     label. If action 0 is displayed and you choose to turn around, output {"action": 0}. The
     action value must be a bare integer copied from a white-circle label; never output
     action words such as 'turn_around', 'go_to', or '0 TURN AROUND'. Note you CANNOT GO
     THROUGH CLOSED DOORS, and you DO NOT NEED TO GO UP OR DOWN STAIRS
```

The <TURNAROUND_NOTE> field is empty during the cooldown. Otherwise it is replaced by the following sentence:

```
NOTE: You are ALREADY facing the intended macro-direction. Pick an arrow to walk. ONLY choose
     action 0 (turn around) if ALL path arrows are physically blocked by solid walls.
```

D.4 LocatorVLM

```
The agent has been tasked with navigating to a <TARGET_UPPER>. The agent has sent you an
    image taken from its current location. There are <NUM_LOCATOR_CANDIDATES> red arrows
    superimposed onto your observation, which represent potential positions. These are
    labeled with a number in a white circle, which represent the location you can move to.
    First, internally determine whether the <TARGET> is in the image, and make sure the
    object you see is ACTUALLY a <TARGET>. Return Number 0 if there is no <TARGET>, or if
    you are not sure. Note a chair must have a backrest and a chair is not a stool. Note a
    chair is NOT a sofa (couch), which is NOT a bed. Second, if there is a <TARGET> in the
    image, determine which circle best represents the location of the <TARGET>. The position
    must be close enough that a person standing there could easily touch the <TARGET>. Give
    the number and a reason. If none of the circles represent a valid position for the
    <TARGET>, return Number 0. Return only one JSON object. It must contain the exact-case
    key "Number" with a bare integer value and the exact-case key "Reason" with a brief
    string value. Examples: Target absent: {"Number": 0, "Reason": "The requested object is
    not visible."}. Target visible and circle 2 is a valid approach position: {"Number": 2,
    "Reason": "Circle 2 is a navigable position next to the target."}. Target visible but no
    numbered position is a valid approach point: {"Number": 0, "Reason": "The target is
    visible, but no numbered position is a valid approach point."}. Number must be a bare
    integer. Never output values such as "circle 2", "Number 2", or "none". Do not copy the
    example Number; choose only a numbered position that actually exists in the current
    image.
```

D.5 VerifierVLM

```
Strictly verify whether numbered candidate <TARGET_CANDIDATE_INDEX> is a correct approach
    position for an actual <TARGET_UPPER> visible in this image. The target may be partially
    occluded or near an image edge. There are <NUM_LOCATOR_CANDIDATES> numbered navigable
    positions, but you MUST NOT select a different number. If TargetVisible=false,
    CandidateCorrect MUST also be false. Return only one JSON object containing exact-case
    keys "TargetVisible" and "CandidateCorrect" with JSON boolean values, and "Reason" with
    a brief string value.
```

D.6 GrounderVLM

```
The target <TARGET_UPPER> has already been selected by the preceding semantic stage. Your
    only task is geometric grounding in this clean, unannotated RGB image. Do not output
    visibility, category, positivity, or explanation fields. If several matching instances
    exist, always localize the one whose bounding-box center is closest to the image center;
    use this deterministic rule on every frame. point must lie on a solid visible surface of
    that same target, away from holes, background, floor, and occluding objects.
    bbox=[x1,y1,x2,y2] must tightly surround it. If the selected instance cannot be
    localized in this frame, return null for both values. Return exactly one JSON object
    with only the lowercase keys bbox and point: {"bbox":[x1,y1,x2,y2],"point":[x,y]}. For
    an unavailable result return {"bbox":null,"point":null}. Do not add any other key or
    text.
```

D.7 Response Handling

All responses are parsed as a single dictionary after normalizing JSON booleans and `null`. LocatorVLM output 0, a nonexistent index, or an unparsable response yields no waypoint. VerifierVLM accepts a candidate only when both `TargetVisible` and `CandidateCorrect` are true. GrounderVLM must satisfy the geometry checks in Appendix C. PlannerVLM output is constrained to its allowed option set, as described in Appendix B. An invalid ActorVLM index invokes the default forward action of 0.20 m.

E Reproduction Configuration

Evaluation follows the task definition in Section 3.1. Each episode contains at most 40 macro steps. A macro step comprises the current observation acquisition, model inference, and one policy action output. In-place rotations for panoramic acquisition and directional alignment are included in this process, and the entire procedure counts as one macro decision. An episode is successful when the agent terminates within $d_{\text{succ}} = 1.0$ m of the goal.

The simulated agent has a radius of 0.18 m and a height of 0.88 m. The RGB-D camera has a resolution of 640×480, a horizontal field of view of $79°$, and a pitch of -0.25 rad. Each panoramic

scan acquires 12 RGB-D observations at 30° intervals through in-place rotations. ObserverVLM and PlannerVLM use the six directions at $\{30, 90, 150, 210, 270, 330\}^\circ$. The scan rotations change the camera orientation but do not contribute to translational path length. After direction selection, the camera aligns with the selected view and outputs the polar action for that macro step.

The Node and Ghost merge radii are 0.50 m and 0.75 m, respectively. In Eq. (3), we set $\rho_{\text{reach}} = 0.65$ m, $\epsilon_m = 0.25$ m, and $N_{\text{fail}} = 2$. The repeated-traversal filtering threshold for Node–Node edges is $N_{\text{rep}} = 2$. The maximum translations in *target search*, *approach*, and *fine approach* are 1.70, 0.40, and 0.25 m, respectively. The association radii for target observation waypoints and 3D anchors are 1.0 m and 0.75 m, respectively. The current target approach terminates when two consecutive translations reduce the remaining distance by less than 0.1 m. In Eq. (9), we set $\delta_o = 0.90$ m.

The common-backbone comparison on HM3D v0.2 uses Qwen3-VL-4B-Instruct. The main experiments and module ablation use Qwen3-VL-32B-Instruct. All module-ablation variants use the same episode IDs, sensor and action configurations, and evaluation settings. All experiments use random seed 48. VLM decoding uses temperature 0, top-$p = 1$, and at most 500 output tokens.

F Model Sensitivity

To examine the effect of the VLM backbone on navigation performance, we compare five models from the Qwen2.5-VL and Qwen3-VL families within the complete GA-Nav system. All six VLM modules use the same backbone in each experiment, while the remaining method configuration and evaluation settings are held fixed.

Table 4: Model sensitivity of the complete method on HM3D v0.2. Bold and underlined values denote the best and second-best results, respectively.

VLM Backbone	SR (%)↑	SPL (%)↑
Qwen2.5-VL-3B	49.3	11.3
Qwen2.5-VL-7B	58.2	20.8
Qwen3-VL-4B	65.7	<u>29.4</u>
Qwen3-VL-8B	<u>71.5</u>	28.6
Qwen3-VL-32B	**73.1**	**39.1**

As shown in Table 4, scaling Qwen2.5-VL from 3B to 7B improves SR and SPL by 8.9 and 9.5 percentage points, respectively. Within the Qwen3-VL family, the 32B model achieves the highest SR of 73.1% and SPL of 39.1%, exceeding the 4B model by 7.4 and 9.7 percentage points. Model scale does not affect the two metrics in lockstep: Qwen3-VL-8B improves SR over the 4B model by 5.8 percentage points but reduces SPL by 0.8 percentage points, showing that a larger model does not necessarily yield higher path efficiency.

Model family also affects navigation performance. Despite having fewer parameters, Qwen3-VL-4B improves SR and SPL over Qwen2.5-VL-7B by 7.5 and 8.6 percentage points, indicating that parameter count alone does not explain the performance differences. Meanwhile, the complete method with Qwen3-VL-4B still outperforms all four existing methods in the common-backbone comparison in Table 2. VLM capability therefore affects the performance ceiling of GA-Nav, while the method retains its relative advantage with a smaller backbone.