AMCoNav: Asynchronous Multi-module Collaborative Framework for Embodied Visual Navigation

Jiaquan Yan[1], Fang Zhao[1]*, Yushi Chen[1], Long Wang[1], Haiyong Luo[2]**, and Dan Luo[3]

[1] Beijing University of Posts and Telecommunications, Beijing, China {yanjiaquan, zfsse, chenyushi, 92536077}@bupt.edu.cn
[2] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China yhluo@ict.ac.cn
[3] Beijing Forestry University, Beijing, China dluo_work@bjfu.edu.cn

Abstract. Embodied visual navigation requires agents with real-time perception, logical reasoning and efficient decision-making capabilities. Existing collaborative modular frameworks struggle to balance accurate decision-making and efficient online execution, leading to error propagation and inference blocking. To tackle this limitation, we propose AMCoNav, an Asynchronous Multi-module Collaborative Framework that unites real-time lightweight backbone networks with on-demand zero-shot large model modules (ZLMM) for robust and real-time embodied navigation. The framework has four tightly coupled components: (1) a Real-Time Multimodal Decision Module (RMDM) for continuous decision making; (2) Zero-shot Large Model Modules (ZLMM) for on-demand asynchronous reasoning; (3) a latency-tolerant Multimodal Shared Context (MSC) for cross-module coordination; and (4) a Bayesian Probabilistic Decision Fusion Module (BPDFM) for robust score fusion and mode switching. Experiments on three representative embodied navigation benchmarks (HM3D-OVON, SG3D, and GOAT-Bench) demonstrate that AMCoNav consistently improves success rates while maintaining real-time execution. The most significant gain appears on HM3D-OVON, where AMCoNav achieves an average 8.7% gain over prior state-of-the-art results, with 60.0%/60.0%/53.3% success rate (SR) on Val Seen/Val Seen Synonyms/Val Unseen. Ablation results show that asynchronous guidance from MSC and ZLMM and the BPDFM both contribute to performance improvement, and their combination yields the best overall results. These results show that asynchronous multi-module collaboration can improve navigation capability without sacrificing real-time responsiveness.

Keywords: Embodied Visual Navigation · Multi-module Collaboration · Vision-Language Models

* Corresponding author.
** Corresponding author.

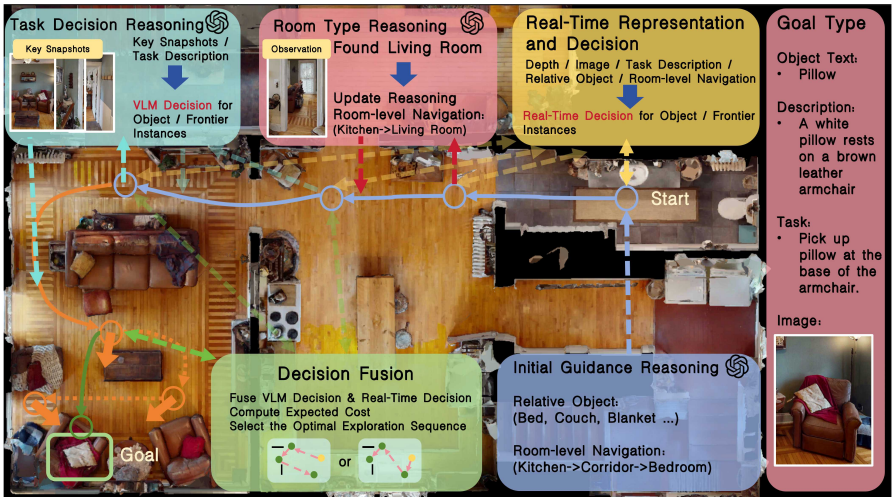


Fig. 1: AMCoNav is an embodied navigation model with multi-module asynchronous collaboration. Its core module processes multimodal inputs and makes real-time decisions. Asynchronously cooperating with a vision-language large model (VLLM) for global navigation guidance, the model finally fuses decision outputs from the real-time model and VLLM to execute navigation tasks. The whole process involves three modes: blue trajectories indicate the exploration mode, orange trajectories stand for the verification mode, and green trajectories represent the navigation mode.

1 Introduction

Embodied intelligent navigation requires models to integrate visual perception, semantic understanding, and real-time decision making [15, 41, 42]. However, end-to-end navigation methods struggle to balance these capabilities and often suffer from severe generalization bottlenecks when facing complex language instructions and dynamic unstructured environments [5, 6, 18, 26, 27, 35]. As a result, multi-module collaborative architectures have become the mainstream paradigm in embodied intelligent navigation [7, 12, 42]. In recent years, Multimodal Large Language Models (MLLMs) have further strengthened cross-modal generalization and semantic reasoning, enabling systems to integrate MLLMs into the perception, reasoning, and decision pipeline [9, 14, 16, 17, 30, 31].

Despite recent progress, existing multi-module systems still have clear limitations. Mainstream pipeline architectures decompose navigation into specialized submodules [28, 32, 42], but one-way information flow can amplify local biases and eventually trigger global failures. Redundant decision architectures improve robustness by running multiple models in parallel for fault tolerance [23, 37], yet their latency is often constrained by the slowest large-model branch, and they can suffer from inference congestion after integrating large models. Global-guided architectures reduce latency instead of online cost by using one large-model decision to produce a global plan that guides local control [21, 40], but these plans

are usually fixed and cannot be revised effectively with new observations. As a result, balancing navigation quality and real-time execution remains difficult.

To address this trade-off, we propose Asynchronous Multi-module Collaborative Framework (AMCoNav), a framework that unites real-time lightweight backbones with on-demand zero-shot large-model modules for efficient navigation. As shown in Fig. 1, the core idea of AMCoNav is to let a real-time module make rapid decisions while large-model modules perform parallel reasoning; all modules communicate through an asynchronously updated context to ensure real-time performance, and their outputs are fused for navigation in different modes. The framework has four tightly coupled core components: (i) a Real-time Multimodal Decision Module (RMDM) for object/frontier scoring; (ii) Zero-shot Large Model Modules (ZLMM) that are triggered on demand to perform room reasoning, target verification, and semantic perception in asynchronous threads; (iii) a Multimodal Shared Context (MSC) for maintaining synchronized object, spatial snapshot, and reasoning cues across modules; and (iv) a Bayesian Probabilistic Decision Fusion Module (BPDFM) for unified probability estimation and switching among exploration, verification, and navigation modes under budget constraints.

Our main contributions are fourfold: (1) We propose an asynchronous collaboration framework that integrates MLLM reasoning to enhance navigation capability while ensuring real-time performance. (2) We develop a real-time multimodal exploration model that leverages the global guidance information output by the MLLM to improve decision quality under latency constraints. (3) We design a Bayesian probabilistic fusion method (with a probability-based algorithm) to fuse real-time and MLLM outputs for robust decision-making and mode switching. (4) Extensive experiments on HM3D-OVON, SG3D, and GOAT-Bench show that AMCoNav improves performance across benchmarks while maintaining real-time operation; the largest gain is on HM3D-OVON, with an average 8.7% improvement over prior SOTA (60.0%/60.0%/53.3% SR on Val Seen/Val Seen Synonyms/Val Unseen).

2 Related Work

Pipeline Modular Architecture Pipeline modular architectures decompose navigation into one or more dedicated submodules for perception, reasoning, and decision making. Some works use CLIP [17], YOLOworld [4], or Grounded-SAM [20] for scene perception and representation, then train a separate decision model for navigation [1, 24, 42]. These systems can be limited by the reasoning capacity of models and may struggle in complex task descriptions and generalization scenarios. To enhance reasoning performance, many studies use perceptual models to extract scene information and construct textual descriptions of the scene for zero-shot large language models to perform reasoning and decision-making [2,7,13,19,28,34,43]. With the development of MLLMs, some methods directly leverage MLLMs for perception and decision-making [8,9,12,16,22,30,32]. Large models introduce substantial latency, and some approaches trade zero-shot

generalization for real-time performance by fine-tuning smaller models [25]. A broader limitation of pipeline modularity is cascading error propagation: mistakes in early perception or representation stages corrupt downstream inputs and can undermine subsequent decision making. Our method ensures real-time performance and multi-fold verification of decision-making via a multi-module asynchronous collaboration mechanism that incorporates dual verification.

Redundant Decision Architectures Redundant decision architectures improve robustness via multi decision model parallelism, combining outputs through fusion rules or evaluators to reduce single-model bias [23, 37]. FlexVLN [37] employs three specialized navigation models to generate action proposals in parallel, then uses an LLM to fuse their outputs into a final decision. CLASH [23] integrates a reactive small-model planner with a reflective large-model reasoner, and uses an uncertainty-aware collaboration mechanism to adaptively fuse their decisions. However, these methods still depend on large models for planning or output fusion, forcing the pipeline to wait for large-model responses and limiting real-time navigation. Our asynchronous collaborative design keeps large-model perception, reasoning, and decision capabilities while avoiding navigation-wide latency losses caused by blocking waits.

Global-Guided Decision Architecture Global-guided decision architectures balance performance and efficiency by separating global planning from local execution [21, 40]. LM-Nav uses an LLM to generate global landmark guidance, while a VNM handles local execution [21]. BeliefMapNav builds a global 3D belief map from large-model semantic priors and refines actions with local planning [40]. Large models typically produce coarse global guidance from initial observations, while lightweight local modules adapt actions online under those constraints. This design leverages large-model reasoning with real-time execution, but global guidance is often fixed and lacks mechanisms for revision. When the environment deviates from initial perception, static global plans can mislead local control and reduce robustness. Our method updates global guidance at key steps using new observations to mitigate this issue.

3 Method

As shown in Fig. 2, AMCoNav has four components: the Real-Time Multimodal Decision Module (RMDM), the Zero-shot Large Model Modules (ZLMM), the Multimodal Shared Context (MSC), and the Bayesian Probabilistic Decision Fusion Module (BPDFM). RMDM runs in the main control loop and produces real-time decisions. ZLMM runs asynchronously and provides complementary semantic and reasoning signals. MSC stores text, images, and high-level representations for cross-module coordination. BPDFM fuses RMDM and ZLMM scores to balance movement cost and exploration efficiency. The superscripts G, L, O, F, and K used in this paper denote Global, Local, Object, Frontier, and Key-object, respectively.

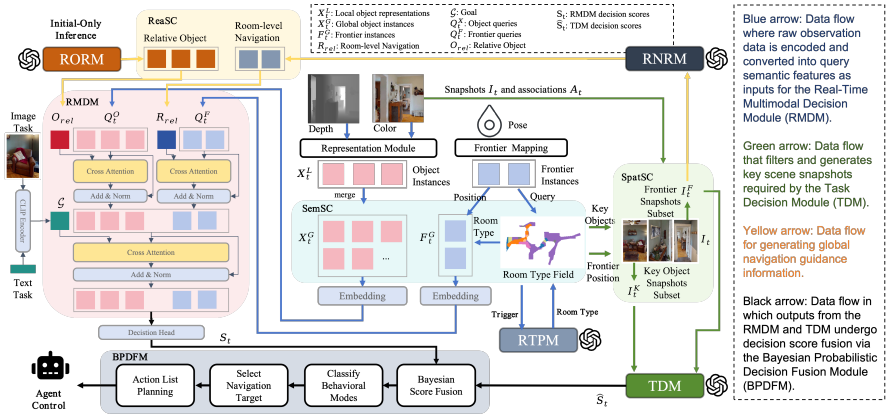


Fig. 2: The Overall Framework of AMCoNav. SemSC, and SpatSC are sub-modules of the MSC, which stores multimodal contextual information. RORM, RNRM, TDM, and RTPM are sub-modules of the ZLMM; they invoke large models in dedicated sub-threads using the MSC, raw observations, and task descriptions. The RMDM outputs real-time navigation decisions, and the BPDFM fuses RMDM and TDM outputs to select navigation coordinates and generate action sequences.

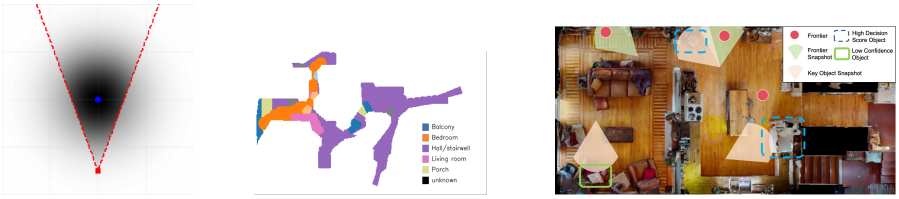
3.1 Multimodal Shared Context (MSC)

MSC is the shared state used by all modules. It stores immediate observations and delayed large-model outputs in a latency-tolerant form, so asynchronous signals can still be consumed by the real-time controller. MSC has three parts: Semantic Scene Context (SemSC), Spatial Snapshot Context (SpatSC), and Reasoning Semantic Context (ReaSC).

Semantic Scene Context (SemSC). The SemSC comprises global object instances $\mathcal{X}_t^G$ and frontier instances $\mathcal{F}_t^G$. We use the same Online Query Representation model as MTU3D [42] for 3D scene representation. For the t-th frame, the RGB and depth inputs are processed to obtain the local object representation set $\mathcal{X}_t^L$. The i-th object representation $x_{t,i}^L \in \mathcal{X}_t^L$ is $[f_{t,i}, c_{t,i}, b_{t,i}, m_{t,i}, v_{t,i}]$, where $f_{t,i} \in \mathbb{R}^C$ are representation features, $c_{t,i} \in \mathbb{R}$ is the confidence score, $b_{t,i} \in \mathbb{R}^6$ is the 3D bounding box in global coordinates, $m_{t,i} \in \mathbb{R}^M$ is the instance mask, and $v_{t,i} \in \mathbb{R}^C$ is the open-vocabulary embedding.

Local object representations $\mathcal{X}_t^L$ are merged into the global object instance set $\mathcal{X}_{t-1}^G$ via the Intersection over Union (IoU)-based Hungarian matching strategy following MTU3D [42]. The merged global representation is denoted by $x_n \in \mathcal{X}_t^G$, where n indexes objects in the scene. In addition, the latest decision score s_n^O from RMDM for each object is retained to select the key snapshot set.

Frontier instances $\mathcal{F}_t^G$ include $q_{t,m} \in \mathbb{R}^3$ and $r_{t,m}$, which denote the 3D coordinate of the m-th frontier instance and its room type at time t. Frontier points lie on the boundary between explored and unexplored regions and are



(a) Confidence Field **(b)** Room Type Field **(c)** Key Snapshot Subset Selection

Fig. 3: Schematic Diagram of Multimodal Shared Context. (a) Confidence field of a single observation; (b) global room-type field; (c) key snapshot subset selection.

updated before each decision step. We project 3D frontier points $q_{t,m}$ onto the 2D plane of the room-type field to obtain corresponding 2D projections $p_{t,m} \in \mathbb{R}^2$.

Since room-type information is acquired from the ZLMM with an inevitable time delay, we maintain two dedicated fields: a room-type field Φ_{rt} and a confidence field Ψ_{rt} for dynamic state estimation, as depicted in Fig. 3a and Fig. 3b. When the ZLMM predicts a room type r with confidence γ_{rt}, we first compute a local confidence field $\Psi_{\mathrm{rt}}^{\mathrm{local}}$ and then merge it with the global field as

$$\Psi_{\mathrm{rt}}^{\mathrm{local}}(p) = \begin{cases} 0, & d(p, p_{\mathrm{ref}}) > R \\ \gamma_{\mathrm{rt}} \cdot \exp\left(\frac{(d(p,p_{\mathrm{ref}})-R)^2}{2\sigma^2}\right), & d(p, p_{\mathrm{ref}}) < R,\ p \in \mathcal{M} \\ \min(\gamma_{\mathrm{rt}}, 0.3) \cdot \exp\left(\frac{(d(p,p_{\mathrm{ref}})-R)^2}{2\sigma^2}\right), & d(p, p_{\mathrm{ref}}) < R,\ p \notin \mathcal{M} \end{cases} \tag{1}$$

$$\Phi_{\mathrm{rt}}(p) = \begin{cases} r, & \Psi_{\mathrm{rt}}^{\mathrm{local}}(p) > \Psi_{\mathrm{rt}}(p) \\ \Phi_{\mathrm{rt}}(p), & \Psi_{\mathrm{rt}}^{\mathrm{local}}(p) \le \Psi_{\mathrm{rt}}(p) \end{cases} \tag{2}$$

$$\Psi_{\mathrm{rt}}(p) = \max\left(\Psi_{\mathrm{rt}}^{\mathrm{local}}(p), \Psi_{\mathrm{rt}}(p)\right), \tag{3}$$

where $p \in \mathbb{R}^2$ denotes an arbitrary 2D point on the room-type field, and $\Psi_{\mathrm{rt}}(p) \in [0, 1]$ represents the current confidence value stored at point p. γ_{rt} denotes the confidence score of room type r inferred by MLLMs. $p_{\mathrm{ref}} \in \mathbb{R}^2$ is the 2D reference point, obtained by projecting the 3D position 1.5 m ahead of the current view direction onto the plane of the room-type field. R denotes the influence radius (set to 3 m), $\mathcal{M}$ is the 2D region covered within 3 m in front of the camera on the room-type field, and $d(\cdot, \cdot)$ is the 2D Euclidean distance metric.

The calculation of local confidence $\Psi_{\mathrm{rt}}^{\mathrm{local}}(p)$ is scaled via distance-based Gaussian decay, with distinct upper bounds assigned to regions inside and outside the field of view; this design ensures a rapid confidence drop for areas either outside the field of view or at long distances. For each 2D point p, we retain the room type associated with the higher confidence score: $\Phi_{\mathrm{rt}}(p)$ is updated to r if $\Psi_{\mathrm{rt}}^{\mathrm{local}}(p) > \Psi_{\mathrm{rt}}(p)$, and $\Psi_{\mathrm{rt}}(p)$ is updated to the maximum of its current value and $\Psi_{\mathrm{rt}}^{\mathrm{local}}(p)$. This enables RMDM to infer room types $r_{t,m} = \Phi_{\mathrm{rt}}(p_{t,m})$ from Φ_{rt} even when ZLMM reasoning is delayed.

Spatial Snapshot Context (SpatSC). The SpatSC stores each observation snapshot I_t and its associations to multiple object instances, defined as the set of pairs $\mathcal{A}_t = \{(I_t, \{x_n \mid x_n \in \mathcal{X}_t^L\})\}$. As shown in Fig. 3c, it manages two snapshot subsets. The first is an optimal snapshot set covering frontier instances. For each frontier instance p_m, we compute the distance between p_m and each snapshot's reference point $p_{ref,t}$ and select the nearest snapshot index $t_m^* = \arg\min_t d(p_{ref,t}, p_m)$, with the corresponding snapshot denoted as $I_m = I_{t_m^*}$. The selected snapshots across all p_m form a frontier instance snapshot set $\mathcal{I}^F$. The second subset is a minimal snapshot set that covers key objects. The key-object set $\mathcal{X}^K$ is constructed by selecting SemSC objects with $c_n < 0.4$ and $s_n^O > 0.6$. We then apply a greedy selection based on $\mathcal{A}_t$, iteratively choosing the snapshot that covers the largest number of uncovered key objects, yielding a set $\mathcal{I}^K$ that covers all key objects. These subsets are used as input to the task decision module and the room-navigation reasoning module.

Reasoning Semantic Context (ReaSC). The ReaSC maintains global guidance, including relevant rooms $\mathcal{R}_{\text{rel}} = [r_{rel,0}, r_{rel,1}, r_{rel,2}, \cdots]$ and relevant objects $\mathcal{O}_{\text{rel}} = [o_{rel,0}, o_{rel,1}, o_{rel,2}, \cdots]$. They are produced by the Relevant-Object Reasoning Module and the Room-Navigation Reasoning Module, and guide the Real-Time Multimodal Decision Module. Object relevance depends only on task relevance and is not dynamically updated. At the room level, the model infers a temporal sequence of room types toward the target based on current perception and the task goal (e.g., $r_a \to r_b \to r_c \to \cdots$), which guides exploration over the next horizon. Once the agent reaches the next room in the sequence, that room is removed. If the actual layout deviates from the inferred sequence, the large-model module updates the guidance using new observations. The reasoning delay is shorter than typical room-level exploration time, so asynchronous updates remain effective.

3.2 Real-Time Multimodal Decision Module (RMDM)

RMDM is the real-time decision core and runs in the main thread. At each step, it reads object and frontier representations from SemSC, combines them with task-level guidance from ReaSC, and outputs navigation scores.

Network Architecture. Figure 2 shows the RMDM architecture. The backbone is a four-layer transformer-style decoder. We construct object queries Q_t^O from $\mathcal{X}_t^G$ and frontier queries Q_t^F from $\mathcal{F}_t^G$, then condition both on guidance $\mathcal{R}_{\text{rel}}, \mathcal{O}_{\text{rel}}$ and task goal $\mathcal{G}$ (text or image). The decoder outputs decision scores as $\mathcal{S}_t = f(Q_t^O, Q_t^F, \mathcal{R}_{\text{rel}}, \mathcal{O}_{\text{rel}}, \mathcal{G})$. Here, Q_t^O contains $f_{t,i}$ and $o_{t,i}$ for every object instance i, while Q_t^F contains the room type $r_{t,i}$ of every frontier instance m; spatial positional encodings are derived from b_t and q_t, respectively. The room type r_t in Q_t^F, $\mathcal{R}_{\text{rel}}$, $\mathcal{O}_{\text{rel}}$, and $\mathcal{G}$ are encoded by the CLIP text/image encoder and projected into a shared feature space.

In cross-attention, frontier queries attend to room guidance, and object queries attend to object guidance. Both branches then attend to task goal $\mathcal{G}$. A final spatial self-attention layer models inter-instance geometry and outputs $\mathcal{S}_t = [s_n^O, s_m^F]$ for object and frontier decisions.

Training. During training, we optimize a multi-task objective to jointly learn scene understanding, reasoning, and navigation decisions. The loss is defined as:

$$L_{\text{total}} = \sum_{h \in \mathcal{H}} \lambda_h L_h, \tag{4}$$

where $\mathcal{H} = \{\text{decision}, \text{class}, \text{rel\_object}, \text{rel\_room}, \text{room\_type}\}$, λ_h denotes the task weight, and L_h is the loss for task h. We use CE for single-label classification heads (object class and room type), and BCE for binary/multi-label heads (decision, relevant-object, and relevant-room).

The ground-truth labels are defined as follows. For class and room-type prediction, each query has one categorical label ($y^{\text{class}} \in \{1, \ldots, K\}$ and $y^{\text{room}} \in \{1, \ldots, R\}$). For relevant-object and relevant-room prediction, each query is supervised by a multi-hot binary vector ($y^{\text{ro}} \in \{0,1\}^M$ and $y^{\text{rr}} \in \{0,1\}^P$). For decision prediction, we use a single positive target index t_i: if the target object is already in the known object set, t_i is that object instance; otherwise, t_i is the frontier instance closest to the target. Detailed CE/BCE formulas are provided in the supplementary material.

3.3 Zero-Shot Large Model Modules (ZLMM)

The ZLMM leverages MLLMs to deliver robust perception, reasoning, and decision-making capabilities. With the exception of the Relevant-Object Reasoning Module (RORM), all large model invocations are executed in independent subthreads to avoid blocking the main thread. Inference latency is mitigated by the MSC, and excessive token overhead is mitigated by demand-driven invocation.

Relevant-Object Reasoning Module (RORM). This module is invoked only at the start of each task. Given the task description, it infers objects likely to be relevant near the target and their relevance score. We keep the top five objects with the highest relevance, store them in the ReaSC as relevant-object guidance $\mathcal{O}_{\text{rel}}$, and later use them as inputs to the RMDM.

Room-Navigation Reasoning Module (RNRM) We obtain frontier instance snapshots from the SpatSC, $\mathcal{I}^F$. For each snapshot, we generate a textual description of the objects outside the field of view within its 2-meter range to supplement additional information (e.g., "a bed 0.5 m to the left"). Based on $\mathcal{I}^F$ and these descriptions, the module performs room-level navigation reasoning to

predict rooms likely to be traversed en route to the target. We store the predicted rooms in the ReaSC as relevant-room guidance $\mathcal{R}_{\text{rel}}$ and later use them as inputs to the RMDM. To reduce computational overhead, this module is only invoked when a new room type is detected by Room-Type Perception Module.

Room-Type Perception Module (RTPM). Given the latest observation image, this module infers the room type ahead and outputs a confidence score, then updates room-type field Φ_{rt} in the SemSC. To reduce inference cost, it is invoked only when the average confidence within the visible region $\mathcal{M}$ falls below 0.4.

Task Decision Module (TDM). This module retrieves $\mathcal{I}^K$ and $\mathcal{I}^F$ from the SpatSC and also obtains scene descriptions from the scene context. It scores each snapshot with respect to the task goal. The scores $\hat{s}_m^F$ from $\mathcal{I}^F$ are assigned to the corresponding frontier instances. For $\mathcal{I}^K$, when TDM is prompted to verify a suspected target, it returns both a decision score $\hat{s}_n^O$ and a 2D box $\hat{b}$; we assign $\hat{s}_n^O$ to object instances that overlap with $\hat{b}$. Once the RMDM makes a decision, we rebuild the key-object set and invoke the TDM for inference. Processed snapshots are marked to avoid repeated queries. We construct $\hat{\mathcal{S}}_t$ from $\hat{s}_n^O$ and $\hat{s}_m^F$ so that it matches $\mathcal{S}_t$ in length and one-to-one correspondence. Due to inference latency and incomplete coverage of $\mathcal{I}^K$, some instances receive no scores; we set these to -1 and handle them separately during BPDFM.

3.4 Bayesian Probabilistic Decision Fusion Module (BPDFM)

BPDFM fuses RMDM and TDM scores into unified success probabilities and selects one of three modes: exploration, verification, or navigation. If both models assign low scores to all candidates, BPDFM enters exploration mode and chooses the next waypoint by minimum expected cost. If both models assign high confidence to at least one candidate, it enters navigation mode and commits to the candidate with highest fused probability. If the two models disagree, BPDFM enters verification mode and samples nearby viewpoints for additional evidence. Verification is enabled only when the step count exceeds 300 or frontier options are exhausted.

Probability Fusion. We build a calibration dataset to map raw decision scores from RMDM ($\mathcal{S}_t$) and TDM ($\hat{\mathcal{S}}_t$) to success probabilities. Nonlinear fitting yields calibrated probabilities $\mathcal{P}_t$ and $\hat{\mathcal{P}}_t$. For entries with missing TDM scores (encoded as -1), we set $\hat{\mathcal{P}}_t = 0.5$, which keeps the fused estimate close to the RMDM prior. Fitting details are provided in the Appendix. To fuse these success probabilities via Bayesian inference, we denote $p_1 \in \mathcal{P}_t$ and $p_2 \in \hat{\mathcal{P}}_t$. The fused probability is then computed as:

$$p_t^{\text{f}} = \frac{p_1 p_2}{p_1 p_2 + (1 - p_1)(1 - p_2)}. \tag{5}$$

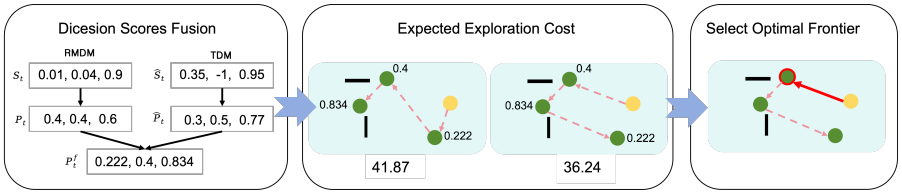


Fig. 4: Schematic Diagram of the Exploration Mode. The model fuses decision scores into probabilities, constructs a graph, computes expected exploration cost, and selects the optimal frontier as the next exploration target.

Exploration Mode. In exploration mode (Fig. 4), the objective is to minimize expected travel cost under a remaining-step budget. We build an undirected weighted graph $G = (V, E, W)$ with frontier nodes V_c, start node n_s, and current best target node n_t (highest p_t^{f}). Edge weights W encode movement cost.

An admissible trajectory is an ordered sequence $\mathcal{T} = [u_1, \ldots, u_k]$ with $u_i \in V_c$, all distinct, $1 \le k \le n$, and total cost below a remaining-step budget D:

$$\mathrm{Cost}(\mathcal{T}) = W(n_s, u_1) + \sum_{i=1}^{k-1} W(u_i, u_{i+1}) + W(u_k, n_t). \tag{6}$$

We set $D = \max(0, t_{\max} - t_{\mathrm{cur}})$, where $t_{\max}$ is the maximum allowed steps and t_{cur} is the number of steps already used. This constraint ensures the agent can return to n_t before the budget is exhausted.

Using a Bellman-style expected-cost criterion, we denote the cumulative movement cost up to node u_m as $C_m = W(n_s, u_1) + \sum_{i=1}^{m-1} W(u_i, u_{i+1})$. The expected exploration cost is then given by:

$$\mathbb{E}[\mathrm{Cost}(\mathcal{T})] = \sum_{m=1}^{k} \left[\left(\prod_{i=1}^{m-1} q_i^f \right) p_m^f C_m \right] + \left(\prod_{i=1}^{k} q_i^f \right) \left(\mathrm{Cost}(\mathcal{T}) + C_f \right), \tag{7}$$

where p_m^f is the success probability at node u_m, $q_m^f = 1 - p_m^f$, and $C_f \gg W(e)$ for all $e \in E$ denotes the penalty incurred if all nodes fail.

Exhaustive search is factorial in the number of frontier nodes, so we solve this problem with state-compressed dynamic programming (SCDP), which gives polynomial-time complexity. The controller follows the first node of the minimum-expected-cost admissible path $\mathcal{T}^*$; if no admissible path exists, it falls back to n_t.

Verification Mode. In verification mode, we sample candidate viewpoints around the target, keep those with sufficient visible free space to the target, and greedily select three geometrically separated views for diversity. The agent visits these verification viewpoints in ascending distance to the target.

Navigation mode. The controller enters navigation mode in two cases: (1) an object passes dual confirmation from RMDM and TDM, or (2) the remaining step budget is insufficient for another exploration cycle. In the second case, the agent directly commits to the object with highest fused probability.

4 Experiments

4.1 Experimental setting

Training Data Collection. We use the same base training datasets as MTU3D, such as HM3D-OVON [33], ScanRefer [3] and Multi3DRefer [38]. Ground-truth collection follows MTU3D [42] with one key change. Frontier selection uses the optimal policy instead of random sampling. This change matches BPDFM, which explicitly considers movement cost when selecting frontiers and therefore penalizes distant high-score candidates. Room-type labels are obtained with MLLMs. We first use HM3DSEM region segmentation [29] to collect unobstructed multi-view images per region, then infer room types from these views. Each candidate observation point inherits the label of its mapped region. We also use an LLM to infer task-relevant target objects, and derive relevant rooms from the room sequence along the optimal ground-truth trajectory.

Implementation Details. Training has two stages: 10 epochs on the full dataset, followed by 10 epochs of task-specific fine-tuning. We use AdamW (learning rate $1e^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$), set all loss weights to 1, and train for about 384 GPU hours on an NVIDIA A100. For simulation, we use the Stretch robot embodiment (height 1.41 m, base radius 17 cm). Inputs are 360×640 RGB images I_t, depth maps D_t, and poses P_t. The action set includes MOVE FORWARD (0.25 m), TURN LEFT (30°), and TURN RIGHT (30°). The multimodal exploration model is triggered when the agent reaches a target position, and 18 frames are sampled between consecutive targets. To emulate asynchronous execution, each agent step is set to 0.5 s, corresponding to 0.5 m/s translation speed and 60°/s rotation speed, which are within the range of typical indoor robots. The large-model module uses GPT-5.

Dataset and Benchmarks We evaluate on GOAT-Bench [10], HM3D-OVON [33], and SG3D [39]. Primary metrics are Success Rate (SR) and SPL: SR $= \frac{N_{\text{success}}}{N_{\text{total}}}$ and SPL $= \frac{1}{N_{\text{total}}} \sum_{i=1}^{N_{\text{total}}} S_i \cdot \frac{l_i}{\max(p_i, l_i)}$, where S_i indicates success, l_i is shortest-path length, and p_i is executed path length. For SG3D, we also report task SR (t-SR). We compare against representative end-to-end methods [18, 33, 36] and modular methods [8–10, 32, 42]; methods without published results on these benchmarks are omitted. MTU3D and MSGNav are our main modular references: MTU3D is real-time with learned representation and decision modules, whereas MSGNav uses YOLO-World [4], SAM [11], and CLIP [17] for perception plus an MLLM for decision inference.

Table 1: Open-vocab navigation results on HM3D-OVON [33].

Method	Val Seen		Val Seen Synonyms		Val Unseen	
	SR (↑)	SPL (↑)	SR (↑)	SPL (↑)	SR (↑)	SPL (↑)
DAgger [33]	18.1	9.4	15.0	7.4	10.2	4.7
RL [27]	39.2	18.7	27.8	11.7	18.6	7.5
BCRL [18]	20.2	8.2	15.2	5.3	8.0	2.8
DAgRL [33]	41.3	21.2	29.4	14.4	18.3	7.9
VLFM [32]	35.2	18.6	32.4	17.3	35.2	19.6
Uni-NaVid [36]	41.3	21.1	43.9	**21.8**	39.5	19.8
TANGO [43]	–	–	–	–	35.5	19.5
DAgRL+OD [33]	38.5	21.1	39.0	21.4	37.1	19.9
MTU3D [42]	55.0	**23.6**	45.0	14.7	40.8	12.1
MSGNav [8]	–	–	–	–	48.3	**27.0**
AMCoNav(Ours)	**60.0**	15.6	**60.0**	20.9	**53.3**	14.8

Table 2: Task-oriented sequential navigation results on SG3D-Nav [39].

Method	s-SR(↑)	t-SR(↑)	SPL(↑)
Embodied Video Agent [39]	14.7	3.8	10.2
SenseAct-NN Monolithic [39]	12.1	7.7	10.1
MTU3D [42]	23.8	8.0	**16.5**
AMCoNav (Ours)	**24.5**	**8.3**	11.3

4.2 Quantitative Results

Open Vocabulary Navigation. Table 1 shows that AMCoNav obtains the highest SR on all three OVON splits: 60.0% (Val Seen), 60.0% (Val Seen Synonyms), and 53.3% (Val Unseen). Relative to MTU3D, the gains are +5.0, +15.0, and +12.5 points. This improvement is consistent with BPDFM's dual-verification design, which suppresses false positives and missed detections. The trade-off is lower SPL, because verification introduces extra steps and BPDFM imposes stricter conditions for target confirmation; This trade-off is consistently observed across other tasks.

Task-oriented Sequential Navigation. In SG3D (Tab. 2), AMCoNav improves both sequence-level and task-level success (24.5% s-SR and 8.3% t-SR). The absolute gain over MTU3D is modest, suggesting that this benchmark is more sensitive to high-level reasoning quality. When real-time predictions are uncertain, BPDFM enters navigation mode less often, which limits the final success rate.

Multi-modal Lifelong Navigation. On GOAT-Bench (Tab. 3), AMCoNav reaches 55.2% SR on Val Seen and 48.8% on Val Seen Synonyms, and matches

Table 3: Multi-modal lifelong navigation results on GOAT-Bench [10].

Method	Val Seen		Val Seen Synonyms		Val Unseen	
	SR (↑)	SPL (↑)	SR (↑)	SPL (↑)	SR (↑)	SPL (↑)
Modular GOAT [10]	26.3	17.5	33.8	24.4	24.9	17.2
Modular CLIP on Wheels [10]	14.8	8.7	18.5	11.5	16.1	10.4
SenseAct-NN Skill Chain [10]	29.2	12.8	38.2	15.2	29.5	11.3
SenseAct-NN Monolithic [10]	16.8	9.4	18.5	10.1	12.3	6.8
DyNaVLM [9]	–	–	–	–	25.5	10.2
TANGO [43]	–	–	–	–	32.1	16.5
MTU3D [42]	52.2	**30.5**	48.4	**30.3**	47.2	27.7
MSGNav [8]	–	–	–	–	**52.0**	**29.6**
AMCoNav (ours)	**55.2**	25.5	**48.8**	21.8	47.2	21.5

Table 4: Time overhead of each module: average over 10 episodes on the A6000 GPU

Real-time Model			ZLMM			
Query Proposal	RMDM	BPDFM	RORM	RNRM	RTPM	TDM
209 ms	106 ms	281 ms	1.51s	14.58s	3.36s	3.99s

MTU3D on Val Unseen (47.2%). Compared with non-real-time MSGNav, the performance gap is most pronounced on the Val Unseen split. A task breakdown on Val Unseen shows strong results on Category (68%) and Language (49.2%) but weaker performance on image-centered tasks (23.6%). This pattern indicates that the current bottleneck is visual inference quality of the real-time branch; when visual confidence is low, BPDFM has fewer chances to commit to navigation mode.

Time Overhead Table 4 reports per-inference runtime for each module. In the real-time branch, the query proposal takes 209 ms, the RMDM takes 106 ms and the BPDFM takes 281 ms, which satisfies the target control frequency. In ZLMM, TDM and RTPM run one inference per observation, whereas RNRM performs joint reasoning over multiple snapshots. With a minimum inter-decision interval of 30 steps (15 s), most large-model calls finish within one decision cycle. Since RNRM provides long-term global guidance rather than per-step control, it poses no real-time concern. The latency-tolerant MSC further reduces the effect of delayed responses on control. To quantify the latency advantage of our asynchronous design, we additionally compare the practical runtime overhead between the synchronous and asynchronous versions of the framework. Specifically, we evaluate the scenario of synchronous TDM decision-making, where the agent must wait for the TDM's decision output before proceeding with each navigation step, with a simulated time cost of 0.5 s per step. Statistical results over 100 random episodes show that the synchronous mode yields an average

Table 5: Ablation experiment results on HM3D-OVON [33].

Method	Val Seen		Val Seen Synonyms		Val Unseen	
	SR (↑)	SPL (↑)	SR (↑)	SPL (↑)	SR (↑)	SPL (↑)
MTU3D	55.0	**23.6**	45.0	14.7	40.8	12.1
AMCoNav w/o BPDFM	56.7	15.8	53.3	17.5	44.2	12.5
AMCoNav	**60.0**	15.6	**60.0**	**20.9**	**53.3**	**14.8**

Table 6: Hyperparameter ablation on HM3D-OVON Val Seen (50 episodes). c_1: step threshold for entering verification; c_2: probability threshold for entering navigation mode; c_3: RTPM confidence threshold; $\overline{N}$: average RTPM invocations per episode.

Setting	c_1	c_2	c_3	SR (↑)	SPL (↑)	$\overline{N}$ (↓)
Default	300	0.4	0.8	**60.0**	13.7	132.08
Vary c_1	200	0.4	0.8	56.0	10.9	–
	400	0.4	0.8	54.0	11.5	–
	500	0.4	0.8	52.0	10.2	–
Vary c_2	300	0.4	0.6	58.0	15.1	–
	300	0.4	0.4	52.0	16.2	–
Vary c_3	300	0.6	0.8	**60.0**	13.1	158.25
	300	0.2	0.8	52.0	9.7	**124.25**
MTU3D	–	–	–	54.0	**16.6**	–

time overhead of 174.1 s per episode, while the asynchronous mode reduces this to 142.5 s per episode, representing an 18.15% reduction in runtime. This result empirically validates that the asynchronous paradigm effectively enhances the real-time responsiveness of embodied navigation. Overall, the timing results are consistent with the design goal: large-model modules add reasoning depth, while the real-time branch keeps control responsive.

4.3 Qualitative results

Ablation Experiment Given that global guidance is co-implemented by MSC and ZLLM, we streamline ablation analysis by solely evaluating the global guidance mechanism and BPDFM algorithm. Table 5 isolates and quantifies the performance impact of each core component on the HM3D-OVON benchmark. Starting from the MTU3D baseline, incorporating global guidance in isolation yields consistent gains in SR across all dataset splits. Subsequent integration of BPDFM into this global guidance framework delivers a further performance boost, achieving state-of-the-art SR across all experimental settings. These results confirm conclusively that the global guidance mechanism and probabilistic fusion strategy function as complementary, rather than redundant, components.

Table 7: Generalization across large models on HM3D-OVON Val Seen (50 episodes).

Large Model	SR (↑)	SPL (↑)
GPT-5	**60.0**	**13.7**
Doubao 2.0-pro	60.0	13.2
Doubao 2.0-mini	58.0	12.3
GPT-4o	56.0	11.0
Qwen-VL-Pro	56.0	10.7

Hyperparameter Analysis. We ablate the three key thresholds on a 50-episode subset of Val Seen (Tab. 6). The default (c_1=300, c_2=0.4, c_3=0.8) yields the highest SR. A smaller c_1 triggers verification too early, hurting exploration, while a larger c_1 starves the agent of observations and causes missed targets. Lowering c_3 relaxes the condition for entering navigation mode, trading a small SR drop for higher SPL - this serves as an SPL-oriented variant. Increasing c_2 over-triggers RTPM without SR gains, wasting inference budget.

Generalization Across Large Models. Tab. 7 validates that AMCoNav is agnostic to the underlying LLM. Stronger models (GPT-5, Doubao 2.0-pro) achieve the best SR, while all five variants remain competitive (SR $\geq$ 56%, SPL $\geq$ 10.7), confirming good scalability across model capacities.

5 Conclusion

In this paper, we presented AMCoNav, an asynchronous collaborative framework for embodied navigation. AMCoNav combines a lightweight real-time policy with on-demand large-model modules, coordinated through a shared multimodal context and a Bayesian fusion mechanism. Across benchmarks, this design improves navigation success while maintaining real-time execution. A main limitation is that the real-time model can bottleneck performance when observations are ambiguous or incomplete. Future work will introduce a confidence-based dynamic gating mechanism to bypass low-confidence real-time predictions, explore confidence-adaptive validation, and strengthen the large-model modules with task-specific score-to-probability mappings and stronger closed-loop reasoning.

6 Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China under Grant 62261042, the Beijing Natural Science Foundation under Grant 4254084, the Yibin high-level Introduction talents project under Grant 2024YG01 and the China Postdoctoral Science Foundation under Grant 2024M750200.

References

1. An, D., Wang, H., Wang, W., Wang, Z., Huang, Y., He, K., Wang, L.: ETPNav: Evolving topological planning for vision-language navigation in continuous environments. IEEE TPAMI (2024)
2. Cao, Y., Zhang, J., Yu, Z., Liu, S., Qin, Z., Zou, Q., Du, B., Xu, K.: CoGNav: Cognitive process modeling for object goal navigation with LLMs. In: ICCV. pp. 9550–9560 (2025)
3. Chen, D.Z., Chang, A.X., Nießner, M.: ScanRefer: 3D object localization in RGB-D scans using natural language. In: ECCV. pp. 202–221 (2020)
4. Cheng, T., Song, L., Ge, Y., Liu, W., Wang, X., Shan, Y.: YOLO-World: Real-time open-vocabulary object detection. In: CVPR. pp. 16901–16911 (2024)
5. Eftekhar, A., Hendrix, R., Weihs, L., Duan, J., Caglar, E., Salvador, J., Herrasti, A., Han, W., VanderBil, E., Kembhavi, A., et al.: The one ring: A robotic indoor navigation generalist. arXiv preprint arXiv:2412.14401 (2024)
6. Goetting, D., Singh, H.G., Loquercio, A.: End-to-end navigation with vision-language models: Transforming spatial reasoning into question-answering. In: International Conference on Neuro-symbolic Systems. Proceedings of Machine Learning Research, vol. 288, pp. 22–35. PMLR (2025)
7. Huang, C., Mees, O., Zeng, A., Burgard, W.: Visual language maps for robot navigation. In: IEEE Int. Conf. Robot. Autom. (ICRA). pp. 10608–10615 (2023)
8. Huang, X., Zhao, S., Wang, Y., Lu, X., Zhang, W., Qu, R., Li, W., Wang, Y., Wen, C.: MSGNav: Unleashing the power of multi-modal 3D scene graph for zero-shot embodied navigation. In: CVPR. pp. 37154–37163 (2026)
9. Ji, Z., Lin, H., Gao, Y.: DyNaVLM: Zero-shot vision-language navigation system with dynamic viewpoints and self-refining graph memory. arXiv preprint arXiv:2506.15096 (2025)
10. Khanna, M., Ramrakhya, R., Chhablani, G., Yenamandra, S., Gervet, T., Chang, M., Kira, Z., Chaplot, D.S., Batra, D., Mottaghi, R.: GOAT-Bench: A benchmark for multi-modal lifelong navigation. In: CVPR. pp. 16373–16383 (2024)
11. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W., et al.: Segment anything. In: ICCV. pp. 4015–4026 (2023)
12. Kuang, Y., Lin, H., Jiang, M.: 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)
13. Lin, B., Nie, Y., Zai, K.L., Wei, Z., Han, M., Xu, R., Niu, M., Han, J., Zhang, H., Lin, L., et al.: EvolveNav: Empowering LLM-based vision-language navigation via self-improving embodied reasoning. IEEE TPAMI (2026)
14. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Jiang, Q., Li, C., Yang, J., Su, H., et al.: Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In: ECCV. pp. 38–55 (2024)
15. Liu, Y., Chen, W., Bai, Y., Liang, X., Li, G., Gao, W., Lin, L.: Aligning cyber space with physical world: A comprehensive survey on embodied AI. IEEE/ASME Trans. Mechatronics (2025)
16. Nie, D., Guo, X., Duan, Y., Zhang, R., Chen, L.: WMNav: Integrating vision-language models into world models for object goal navigation. In: IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS). pp. 2392–2399 (2025)
17. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML. pp. 8748–8763 (2021)

18. Ramrakhya, R., Batra, D., Wijmans, E., Das, A.: PIRLNav: Pretraining with imitation and RL finetuning for objectnav. In: CVPR. pp. 17896–17906 (2023)
19. Rana, K., Haviland, J., Garg, S., Abou-Chakra, J., Reid, I., Suenderhauf, N.: SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning. arXiv preprint arXiv:2307.06135 (2023)
20. Ren, T., Liu, S., Zeng, A., Lin, J., Li, K., Cao, H., Chen, J., Huang, X., Chen, Y., Yan, F., et al.: Grounded SAM: Assembling open-world models for diverse visual tasks. arXiv preprint arXiv:2401.14159 (2024)
21. Shah, D., Osiński, B., Levine, S., et al.: LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In: Conf. Robot Learn. (CoRL). pp. 492–504 (2023)
22. Shi, X., Li, Z., Lyu, W., Xia, J., Dayoub, F., Qiao, Y., Wu, Q.: SmartWay: Enhanced waypoint prediction and backtracking for zero-shot vision-and-language navigation. In: IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS). pp. 16923–16930 (2025)
23. Wang, L., He, Z., Li, J., Xia, R., Hu, M., Yao, C., Liu, C., Tang, Y., Chen, Q.: CLASH: Collaborative large-small hierarchical framework for continuous vision-and-language navigation. arXiv preprint arXiv:2512.10360 (2025)
24. Wang, Z., Li, X., Yang, J., Liu, Y., Jiang, S.: GridMM: Grid memory map for vision-and-language navigation. In: ICCV. pp. 15625–15636 (2023)
25. Wei, M., Wan, C., Peng, J., Yu, X., Yang, Y., Feng, D., Cai, W., Zhu, C., Wang, T., Pang, J., et al.: Ground slow, move fast: A dual-system foundation model for generalizable vision-and-language navigation. arXiv preprint arXiv:2512.08186 (2025)
26. Wijmans, E., Essa, I., Batra, D.: VER: Scaling on-policy RL leads to the emergence of navigation in embodied rearrangement. NeurIPS **35**, 7727–7740 (2022)
27. Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., Batra, D.: DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In: ICLR (2020)
28. Wu, P., Mu, Y., Wu, B., Hou, Y., Ma, J., Zhang, S., Liu, C.: VoroNav: Voronoi-based zero-shot object navigation with large language model. In: ICML (2024)
29. Yadav, K., Ramrakhya, R., Ramakrishnan, S.K., Gervet, T., Turner, J., Gokaslan, A., Maestre, N., Chang, A.X., Batra, D., Savva, M., et al.: Habitat-matterport 3D semantics dataset. In: CVPR. pp. 4927–4936 (2023)
30. Yang, Y., Yang, H., Zhou, J., Chen, P., Zhang, H., Du, Y., Gan, C.: 3D-Mem: 3D scene memory for embodied exploration and reasoning. In: CVPR. pp. 17294–17303 (2025)
31. Yang, Z., Li, L., Lin, K., Wang, J., Lin, C., Liu, Z., Wang, L.: The dawn of LMMs: Preliminary explorations with GPT-4V(ision). arXiv preprint arXiv:2309.17421 (2023)
32. Yokoyama, N., Ha, S., Batra, D., Wang, J., Bucher, B.: VLFM: Vision-language frontier maps for zero-shot semantic navigation. In: IEEE Int. Conf. Robot. Autom. (ICRA). pp. 42–48 (2024)
33. Yokoyama, N., Ramrakhya, R., Das, A., Batra, D., Ha, S.: HM3D-OVON: A dataset and benchmark for open-vocabulary object goal navigation. In: IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS). pp. 5543–5550 (2024)
34. Yu, B., Kasaei, H., Cao, M.: L3MVN: Leveraging large language models for visual target navigation. In: IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS). pp. 3554–3560 (2023)

35. Zeng, K., Zhang, Z., Ehsani, K., Hendrix, R., Salvador, J., Herrasti, A., Girshick, R., Kembhavi, A., Weihs, L.: PoliFormer: Scaling on-policy RL with transformers results in masterful navigators. arXiv preprint arXiv:2406.20083 (2024)
36. Zhang, J., Wang, K., Wang, S., Li, M., Liu, H., Wei, S., Wang, Z., Zhang, Z., Wang, H.: Uni-NaVid: A video-based vision-language-action model for unifying embodied navigation tasks. arXiv preprint arXiv:2412.06224 (2024)
37. Zhang, S., Qiao, Y., Wang, Q., Guo, L., Wei, Z., Liu, J.: FlexVLN: Flexible adaptation for diverse vision-and-language navigation tasks. IEEE TMM **27**, 6307–6318 (2025)
38. Zhang, Y., Gong, Z., Chang, A.X.: Multi3DRefer: Grounding text description to multiple 3D objects. In: ICCV. pp. 15225–15236 (2023)
39. Zhang, Z., Zhu, Z., Li, J., Li, P., Wang, T., Liu, T., Ma, X., Chen, Y., Jia, B., Huang, S., et al.: Task-oriented sequential grounding and navigation in 3D scenes. arXiv preprint arXiv:2408.04034 (2024)
40. Zhou, Z., Hu, Y., Zhang, L., Li, Z., Chen, S.: BeliefMapNav: 3D voxel-based belief map for zero-shot object navigation. In: NeurIPS (2025)
41. Zhu, F., Zhu, Y., Lee, V., Liang, X., Chang, X.: Deep learning for embodied vision navigation: A survey. arXiv preprint arXiv:2108.04097 (2021)
42. Zhu, Z., Wang, X., Li, Y., Zhang, Z., Ma, X., Chen, Y., Jia, B., Liang, W., Yu, Q., Deng, Z., et al.: Move to understand a 3D scene: Bridging visual grounding and exploration for efficient and versatile embodied navigation. In: ICCV. pp. 8120–8132 (2025)
43. Ziliotto, F., Campari, T., Serafini, L., Ballan, L.: TANGO: Training-free embodied AI agents for open-world tasks. In: CVPR. pp. 24603–24613 (2025)