OpenIN: Open-Vocabulary Instance-Oriented Navigation in Dynamic Domestic Environments

Yujie Tang, Meiling Wang, Yinan Deng, Zibo Zheng, Jingchuan Deng, Sibo Zuo, and Yufeng Yue, *Member, IEEE*

***Abstract*—In daily domestic settings, frequently used objects like *cups* often have unfixed positions and multiple instances within the same category, and their carriers also frequently change. As a result, it becomes challenging for a robot to efficiently navigate to a specific instance. To tackle this challenge, the robot must capture and update scene changes and plans continuously. However, current object-navigation approaches primarily focus on the semantic level and lack the ability to dynamically update the scene representation. In contrast, this paper captures the relationships between frequently used objects and their static carriers. It constructs an open-vocabulary Carrier-Relationship Scene Graph (CRSG) and updates the carrying status during robot navigation to reflect the dynamic changes of the scene. Based on CRSG, we further propose an instance navigation strategy that models the navigation process as a Markov Decision Process. At each step, decisions are informed by the Large Language Model's commonsense knowledge and visual-language feature similarity. We designed a series of long-horizon navigation tasks for frequently used everyday items in the Habitat simulator. The results demonstrate that by updating the CRSG, the robot can navigate efficiently to moved targets. Additionally, we conducted extensive experiments on a real robot, demonstrating the effectiveness of our method and exploring its limitations.**

***Index Terms*—Instance Navigation, Carrier-Relationship Scene Graph, Dynamic Scenes.**

I. Introduction

As ONE of the fundamental tasks of embodied AI, object navigation [1] has garnered widespread attention from researchers. Imagine a daily environment in which a robot is tasked with efficiently navigating to any object, whether it is static furniture or a frequently used item with changing positions, such as *a black cup*. This poses several requirements for the navigation algorithm: 1. Open-vocabulary [2] recognition and instruction, 2. Accurate instance differentiation, 3. Memorizing and updating object states, and 4. Effective navigation strategies. However, there is still a significant gap in achieving this goal.

Received 6 January 2025; accepted 30 June 2025. Date of publication 23 July 2025; date of current version 4 August 2025. This article was recommended for publication by Associate Editor S. Thakar and Editor A. Banerjee upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 62473050, Grant 92370203, and Grant 62233002. *(Corresponding author: Yufeng Yue.)*

Yujie Tang, Meiling Wang, Yinan Deng, Jingchuan Deng, Sibo Zuo, and Yufeng Yue are with the School of Automation, Beijing Institute of Technology, Beijing 100081, China (e-mail: yueyufeng@bit.edu.cn).

Zibo Zheng is with the School of Mechanical Engineering, University of Nottingham Ningbo China, Ningbo 315100, China.

The project page can be found here: https://OpenIN-nav.github.io.

Digital Object Identifier 10.1109/LRA.2025.3592071

Many existing object navigation methods [1], [3], [4], [5], [6], [7], [8] are limited to closed-set navigation, leading to poor performance with unknown objects. With the advancement of visual language models (VLM) [9] and multi-modal large language models (MLLM) [10], open-vocabulary detection is achievable. However, most object navigation methods [11], [12], [13] support only one instruction type, which limits flexibility. In contrast, humans use diverse instructions, such as demands or instance descriptions. To meet this requirement, the proposed method supports multiple instruction types, including demands, semantics, and instance descriptions (e.g., color or object relationships). In addition, instance-level navigation requires precise target discrimination, but most methods lack dedicated target identification modules, resulting in lower accuracy. To address this, we designed a multimodal text-image input, combining text feature similarity, RGB feature similarity, and image-level similarity from a MLLM to enhance the accuracy of instance-level object navigation.

Moreover, in daily scenarios, the location of frequently used instance objects (e.g., *a blue cup*) is dynamic (e.g., from *the living room* to *the kitchen*), and their carriers can change (e.g., *on a coffee table* or *dining table*), making efficient navigation to specific instances challenging. This requires scene memory and dynamic updates, which many existing methods struggle to handle effectively [1], [3], [4], [5], [6], [7], [8], [11], [12], [13], [14]. We capture the carried-by relationships between commonly used instance objects and their carriers, constructing a dynamic Carrier-Relationship Scene Graph (CRSG) to memorize and update these relationships over time. In addition, we update the existence confidence of carriers, as well as other spatial relationships (e.g., *next to*) between objects, ensuring that the scene graph remains consistent and informative during long-term navigation.

Finally, the positional variability of commonly used instances requires an efficient navigation strategy. Based on the CRSG, we designed an instance-oriented navigation strategy that models the object search process as a Markov Decision Process (MDP) [15]. At each navigation step, the robot navigates toward a candidate target or an unexplored carrier object. Specifically, candidate targets are identified via visual language features and ranked based on factors like navigation costs. If no candidate target is available, we rank the exploration order of carriers based on their existence confidence and commonsense knowledge from the large language model (e.g., *a cup is more likely to be placed on the table*), enabling efficient instance navigation.

In summary, our contributions are as follows.

- We present an open-vocabulary, instance-oriented navigation system that supports multi-type object navigation

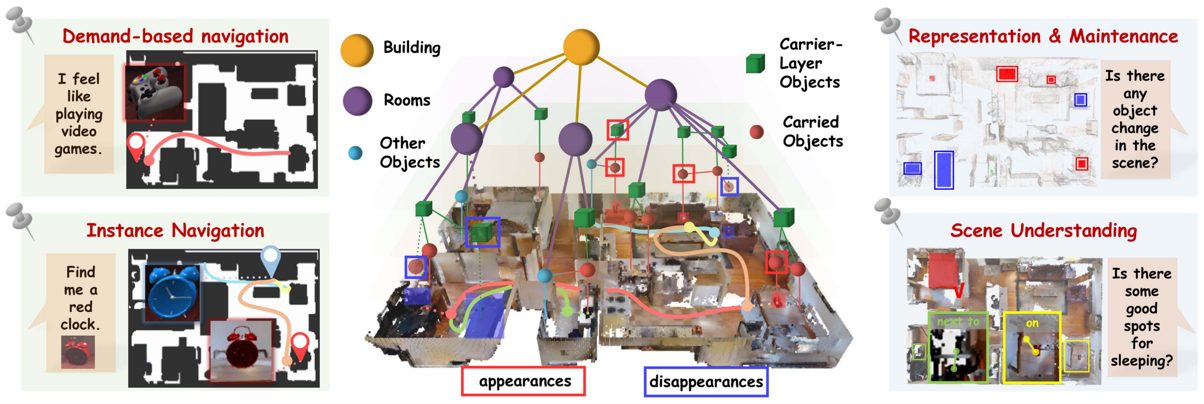

Fig. 1. OpenIN supports multi-type (demand, semantic, and instance-level) object navigation. Additionally, it enables maintaining the latest scene representations through a dynamic carrier-relationship scene graph, along with scene understanding.

instructions, enabling effective navigation to everyday instances with variable positions. (see Fig. 1).

- We present an adaptable carrier relationship scene graph (CRSG) that primarily describes the dynamic carried-by relationships between objects.
- We design a CRSG-based navigation strategy, utilizing visual language features and commonsense knowledge from the MLLM to inform decision making.
- Extensive qualitative and quantitative experiments in the simulation and real world demonstrate that updating the CRSG contributes to efficient navigation in tasks involving long sequences of moved instances.

II. Related Work

A. Open Vocabulary Mapping

With the advent of vision language models like CLIP [16] and its variants, scene reconstruction has moved beyond fixed classes [17], [18], [19], [20], [21], and expanded to open-vocabulary [2], [22]. Clip-fields [9] integrates CLIP and SBERT features into the neural implicit map, enabling open-vocabulary map queries and navigation for a robot. Vlmap [23] projects CLIP features top-down onto a 2D grid to enable zero-shot language navigation. Conceptgraph [22] and Hovsg [24] constructed instance-level point cloud maps with embedded CLIP features and scene graph representing certain relationships between objects, which facilitates more detailed and precise object retrieval.

These maps provide crucial support for applications such as open-vocabulary object queries, scene understanding, and robot navigation. However, they generally lack dynamic update capabilities. We construct a dynamic carrier-relationship scene graph (CRSG) that describes different dynamic relationships between objects (e.g., *carried by*), and continuously update it during navigation. This enables more efficient navigation to everyday instances.

B. Object Navigation

Object navigation [1], [3], [4], [5], [6], [7], [8], [11], [12], [13], [14], [25], as one of the key tasks in the field of embodied AI, primarily involves navigating to a specified semantic object or instance within a scene. [1], [3], [4], [5], [6], [7], [8] mainly perform object navigation within closed-set classes. [22], [24], [26] constructed an open-vocabulary instance map of the scene based on VLMs, enabling open-set instance navigation. [11], [12], [13], [14] perform open-vocabulary object navigation using the frontier exploration method. However, they [11], [12], [13], [14], [22], [24], [26] cannot capture and update the dynamics of instances in everyday environments and typically lack an instance discrimination module, which makes efficient navigation to dynamic instances challenging. In contrast, we build a dynamic CRSG that captures instance changes. Moreover, by considering multiple factors, such as visual language features and image similarity, we achieve more accurate instance identification and navigation. The approach most similar to ours is GOAT [27], which also implements memory capabilities for the latest scene and supports multi-type navigation command inputs. However, for navigating to a displaced everyday object, we designed a navigation strategy based on CRSG, while GOAT selects the closest unexplored region for exploration.

III. Scene Graph Construction and Updating

In a daily environment, when given a navigation command, an intuitive answer is that the robot queries the offline map to determine the endpoint and navigates there. If the target is a daily item (e.g., *a cup*) that is being carried, the robot evaluates whether the item remains in its original location based on current observations. If not, the robot initiates a strategic exploration process. We define this challenge as a **displaced instance exploration task** within an everyday setting. An overview of the framework is provided in Fig. 2.

A. Offline Carrier-Relationship Scene Graph (CRSG)

We first construct an offline open-vocabulary instance map $\mathcal{M}$ using the pre-collected RGB-D data of the scene. Unlike Conceptgraph [22], where each instance object $O_i \in \boldsymbol{O}$ ($\mathbf{O}$ is the set of all objects) is represented by a CLIP feature, we enhance each instance by adding a caption description list $\boldsymbol{cap}_i$, generated using the Tokenize Anything model [28], as well as a text feature $\boldsymbol{T\_F}_i$ encoded using the SBERT model [29]. A Carrier-Relationship Scene Graph (CRSG) is then constructed below.

Building and Room layer: Existing works, such as [24], have proposed various methods for room segmentation. We segment rooms by first projecting wall and door point clouds onto the 2D

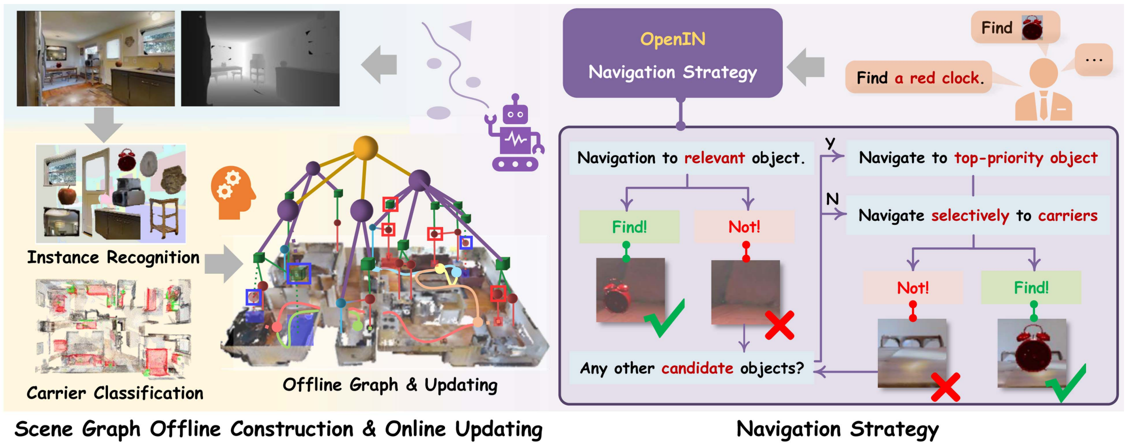
Instance Recognition
Carrier Classification
Offline Graph & Updating
Scene Graph Offline Construction & Online Updating
OpenIN
Navigation Strategy
Find
Find a red clock.
...
Navigation to relevant object.
Y
Navigate to top-priority object
N
Navigate selectively to carriers
Find!
Not!
Not!
Find!
Any other candidate objects?
Navigation Strategy

Fig. 2. The OpenIN framework consists of two main modules. The Scene Graph Offline Construction and Online Updating module builds an offline scene graph capturing carrier-carried relationships and updates carried objects or carriers, as well as geometry-aware edges between objects. The Navigation Strategy module executes active navigation according to user instructions and the proposed strategy.

plane and densifying them. Inspired by the line search method in autonomous navigation [30], we initialize the boundary tracing based on door orientation and perform clockwise search along walls to detect closed loops, which define individual rooms. The objects within each loop are then assigned to the corresponding room. The resulting rooms are then combined to form the overall building layer.

Carrier layer: We calculate the similarity between the text features $\boldsymbol{T\_F}_i$ of each object O_i and the SBERT encoded text feature $\tilde{\boldsymbol{T}}$ for "*furniture for holding objects*". The mathematical expression for this is as follows, where $sim(\cdot,\cdot)$ denotes the cosine similarity function.

$$sim(\boldsymbol{T\_F}_i, \tilde{\boldsymbol{T}}) = \frac{\boldsymbol{T\_F}_i \cdot \tilde{\boldsymbol{T}}}{||\boldsymbol{T\_F}_i|| \cdot ||\tilde{\boldsymbol{T}}||} \tag{1}$$

Next, we select the objects in $\boldsymbol{O}$ whose similarity scores exceed a specified threshold.

For each selected object O_i, we extract the k most frequent captions from the corresponding $\boldsymbol{cap}_i$, where k is a configurable parameter (default: 3). These captions are then input into an MLLM (GPT-4o in our experiments) using a specially designed prompt $\mathcal{P}$ to identify candidate carrier-type objects.

Among these candidates, we further filter objects based on geometric criteria such as size and ground contact to obtain the final carrier-layer objects, denoted as $\bar{\boldsymbol{O}}$. We also collect the category set of these carriers to guide future updates. Each carrier is also assigned a **confidence score** p, representing its **likelihood of existence**, which is continuously updated based on observation consistency and used to prioritize exploration.

Carried layer and other objects: For any non-carrier-layer object $O_i \in (\boldsymbol{O} - \bar{\boldsymbol{O}})$, to determine whether it is carried by a carrier-layer object $O_j \in \bar{\boldsymbol{O}}$, we define a binary function $\boldsymbol{h}(O_j, O_i)$ based on their spatial relationship. Specifically, $\boldsymbol{h}(O_j, O_i) = 1$ if the following conditions are satisfied:

- *Vertical positioning:* The bottom height of O_i is lower than the top of O_j plus 0.1 m, and the vertical center of O_i is at least 0.02 m above the bottom of O_j.
- *Spatial overlap (XY plane):* The projection of O_i's center onto the XY plane falls within a 0.02 m margin of O_j's XY bounding box.
- *Size constraint:* O_i does not exceed 1.2 m in either width or length, reflecting the typical size of carried objects.
- *Proximity in 3D space:* The minimum Euclidean distance between the center of O_i and any point in O_j's point cloud is less than 0.3 m.

For any $O_j \in \bar{\boldsymbol{O}}$, we define the set of objects $\boldsymbol{C}(O_j)$ carried by O_j as follows:

$$\boldsymbol{C}(O_j) = \{O_i | \boldsymbol{h}(O_j, O_i) = 1, O_i \in (\boldsymbol{O} - \bar{\boldsymbol{O}}), O_j \in \bar{\boldsymbol{O}}\} \tag{2}$$

All carried objects form the carried layer, represented by the object set $\mathcal{C}$ in (3). The carried-layer objects can be updated by adding or removing them based on the robot's latest environmental observations, as detailed in III-B.

$$\mathcal{C} = \bigcup_{O_j \in \bar{\boldsymbol{O}}} \boldsymbol{C}(O_j) \tag{3}$$

Except for the carrier-layer objects $\bar{\boldsymbol{O}}$ and carried-layer objects $\mathcal{C}$, the remaining objects are considered other objects, which can also be queried using SBERT features.

Other Object Relationships: Since the CRSG is grounded in geometry-aware instance point clouds, we leverage GPT-4o to automatically generate functions that take object geometric attributes as input and determine common spatial relationships, thereby enriching the CRSG with more semantic edges.

B. Online CRSG Updating While Navigating

As the robot navigates, it captures RGB and depth images from the environment and updates the scene graph online.

Matching and Updating carrier-layer objects: The observed RGB images are processed through CropFormer [31], Tokenize Anything [28] and SBERT [29] to obtain instance masks, captions and SBERT features, respectively. To identify observed carrier-layer objects $\boldsymbol{O}^{cr}_{match}$, the robot compares each newly observed object against existing carriers $\bar{\boldsymbol{O}}$. The comparison considers factors such as the object size, the distance between center positions, and SBERT feature similarities. Besides, if any newly observed object belongs to the pre-collected set of carrier categories, is not close to any existing carrier in the CRSG, and satisfies the geometric criteria of a carrier, it is added as a **new carrier-layer object**. What's more, to determine

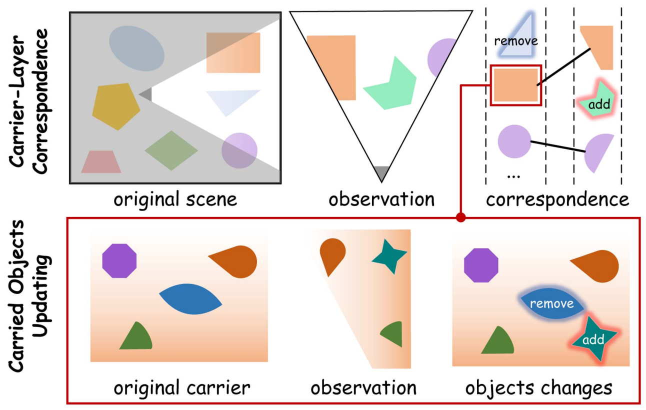

Fig. 3. This diagram illustrates how the CRSG is updated during navigation from the **top-down view**. **Carrier-layer objects** are first compared and matched with currently observations. For each matched carrier, **newly detected carried objects** are compared with previously carried ones on it based on size, spatial position, and SBERT feature similarity. The carriers and carried objects are then **added, removed, or retained** accordingly to reflect the updated scene.

whether any carrier has disappeared, each carrier's 3D center is projected onto the RGB observations, and its category is matched against the captions of detected bounding boxes at that location. If a carrier is expected to match but does not (i.e., it should be visible given its projected position and no occlusion), its existence confidence $p \in [0, 1]$ is decreased accordingly. A carrier is **removed from the CRSG** once its confidence falls below a predefined threshold. For a carrier, its p is updated in (4).

$$p_T = \alpha * m_T + (1 - \alpha) * p_{T-1} \tag{4}$$

where p_T and p_{T-1} denote the confidence at the **expected observation time step** T and $T-1$, respectively. $\alpha = 0.5$ is the smoothing factor, and m_t is a binary indicator that equals 1 if a matching detection is found at T, and 0 otherwise.

Addition and Removal of Carried Objects: For currently observed instances, $h(\cdot, \cdot)$ in (2) is used to determine whether they are being carried by $\boldsymbol{O}^{cr}_{match}$. The newly identified carried objects are then compared with the previously recorded carried objects on $\boldsymbol{O}^{cr}_{match}$. The comparison criteria include object's size, the distance between their center positions, and the SBERT feature similarity score. After the comparison, the carried objects on $\boldsymbol{O}^{cr}_{match}$ are updated accordingly: they are either added, removed, or left unchanged. Notably, certain carrier-layer objects are often only partially observed. Therefore, for each observed carrier-layer object, we calculate the distance between its partial point cloud and the carried objects recorded in the CRSG; if the distance exceeds a threshold, the carried object's state is not updated. Fig. 3 illustrates a schematic of the CRSG updating. In addition, spatial relationships such as *next to*, *below*, and *in between* are dynamically updated as the associated objects are added or removed from the scene graph.

IV. Navigation Strategy for an Instance Object

Before navigation begins, we assume that an initial CRSG has been constructed as described in Section III-A, which serves as the initialization for the planner.

Let the **input navigation command** for the target object be either a $\boldsymbol{text}$, an $\boldsymbol{image}$, or both. For solely an $\boldsymbol{image}$, it is passed through GPT-4o to obtain a textual description of the target object, also denoted as $\boldsymbol{text}$ for simplicity. Then, $\boldsymbol{text}$ is encoded using the SBERT model. The resulting feature is compared with the SBERT features of each object in the CRSG using cosine similarity, similar to (1). The object with the highest similarity score is chosen as a candidate target, O_{ct}.

We model the exploration of an instance object as a fixed-policy Markov decision process (MDP) below.

State Space $\mathcal{S}$: In the current step t, we define:
1. the robot's position L_t,
2. the set of unexplored carrier-layer objects CR_t,
3. the set of candidate target objects CT_t on CR_t,
4. the flag of finding the target or not $F_t \in \{0, 1\}$.
The state variable S_t is defined in (5).

$$S_t = (L_t, CR_t, CT_t, F_t) \in \mathcal{S} \tag{5}$$

In the initial state $S_0 = (L_0, CR_0, CT_0, F_0)$, L_0 is the initial position of the robot, $CR_0 = \boldsymbol{O}$, and $CT_0 = \{O_{ct}\}$.

Action Space $\mathcal{A}$: At each time step t, the agent selects an action $a_t \in \mathcal{A}$ based on the state S_t following a policy $\pi(\cdot)$:

$$a_t = \pi(S_t) \tag{6}$$

The action space $\mathcal{A}$ is defined as follows:

- $Explore(O_j)$: explore a carrier-layer object $O_j \in CR_t$.
- $Goto(O_i)$: navigate to a candidate target $O_i \in CT_t$.
- $Stop$: end the task if it is completed or considered failed.

Policy $\pi(\cdot)$: Given current state $S_t = (L_t, CR_t, CT_t, F_t)$,
1. if $F_t = 1$ or $CR_t = \emptyset$, then $a_t = Stop$.
2. If $F_t = 0$ and $CT_t \neq \emptyset$, we prioritize and select a candidate object to proceed with. Specifically, let $CT_t = \{O_1, \ldots, O_k\}$. Some additional variables are stored: the textual similarities $SS_t = \{\boldsymbol{ss}_1, \ldots, \boldsymbol{ss}_k\}$ between $\boldsymbol{text}$ and CT_t, the distances $D_t = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_k\}$ between L_t and CT_t, and the average depth values $\tilde{D}_t = \{\tilde{\boldsymbol{d}}_1, \ldots, \tilde{\boldsymbol{d}}_k\}$ when CT_t are observed by the robot's camera. The priority rating of any $O_i \in CT_t$ corresponding to $\boldsymbol{ss}_i$, $\boldsymbol{d}_i$ and $\tilde{\boldsymbol{d}}_i$, is evaluated in (7) and (8), where the parameters are set to $\omega_1 = 5$, $\omega_2 = 1$, $\tilde{d} = 0.3$ m, $\alpha = 10$, and $\beta = 0.1$ in the experiments.

$$P\_R(O_i) = \omega_r \cdot \frac{\omega_1 \cdot \boldsymbol{ss}_i \cdot f(\tilde{\boldsymbol{d}}_i)}{1 + \omega_2 \cdot \boldsymbol{d}_i} \tag{7}$$

$$f(\tilde{\boldsymbol{d}}_i) = \begin{cases} \exp(\alpha(\tilde{\boldsymbol{d}}_i - \tilde{d})) & \tilde{\boldsymbol{d}}_i < \tilde{d} \\ \exp(-\beta(\tilde{\boldsymbol{d}}_i - \tilde{d})) & \tilde{\boldsymbol{d}}_i \geq \tilde{d} \end{cases} \tag{8}$$

where $P\_R(O_i)$ increases with $\boldsymbol{ss}_i$, as higher similarity implies higher target likelihood. Moreover, $f(\tilde{\boldsymbol{d}}_i)$ is used to model the confidence level of $\boldsymbol{ss}_i$ based on $\tilde{\boldsymbol{d}}_i$. Specifically, in (8), when $\tilde{\boldsymbol{d}}_i$ is less than a small distance $\tilde{d}$, we assume that the confidence of the front-end detection drops sharply, as being too close to the detected object may cause observation distortion or incompleteness. Conversely, when $\tilde{\boldsymbol{d}}_i$ exceeds this distance, we assume that the confidence decreases more gradually. Considering that in daily life, moved objects are more likely to remain within the same room, we slightly adjust the weighting strategy to reflect this common prior. Specifically, we set $\omega_r = 1$ if the two objects are in the same room, and $\omega_r = 0.8$ otherwise. Note that this adjustment is not a hard constraint, and our method fully supports object navigation across different rooms. The robot will navigate to the object with the maximum P_R and explore for the target.

3. If $F_t = 0$, $CT_t = \emptyset$, and $CR_t \neq \emptyset$, the MLLM selects one of the carrier objects $O_j \in CR_t$ and the robot executes the action $a_t = Explore(O_j)$. Specifically, the captions of each carrier object in CR_t are extracted and provided as input to the MLLM, along with the image or caption of the target object. Leveraging the LLM's commonsense understanding of carrier-carried relationships (e.g., "*a cup is unlikely to be placed on a toilet*"), the MLLM identifies the most plausible carrier. To improve reliability, each unexplored carrier is further weighted by its existence confidence score p, biasing the selection toward carriers that are both semantically appropriate and likely to exist based on past observations.

State Transition Process: If $a_t = Explore(O_j)$ (where $O_j \in CR_t$) or $a_t = Goto(O_i)$ (where $O_i \in CT_t$, and for simplicity, we assume O_i **is carried by** O_j), during the robot's movement, CR_t and CT_t are updated as follows:

The robot first removes from CR_t the navigated carrier object O_j, along with any nearby carrier objects within a small radius r that lack candidate targets based on the latest observations. Meanwhile, CT_t is updated by adding newly discovered candidates (whose SBERT feature similarities to the target exceed a threshold σ_1).

1. If $a_t = Explore(O_j)$, we update:

$$CR_{t+1} = CR_t \setminus (\{O_j\} \cup \text{ nearby carriers without candidates}) \tag{9a}$$

$$CT_{t+1} = CT_t \cup \text{new candidates} \tag{9b}$$

2. If $a_t = Goto(O_i)$, CR_{t+1} is updated in the same way as described in (9a), and we also update:

$$CT_{t+1} = CT_t \setminus \{ct\} \cup \text{new candidates} \tag{10}$$

In addition, CR_{t+1} incorporates newly discovered carriers and removes disappeared ones. Accordingly, CT_{t+1} incorporates candidates supported by the new carriers and discarding those associated with disappeared carriers.

Target Determination Criteria: In either case, the input $\boldsymbol{text}$ (and $\boldsymbol{image}$) is compared against the objects carried by O_j to measure similarity. First, we calculate the SBERT feature similarity sim_{sbert} like (1) between $\boldsymbol{text}$ and the carried objects. If the input includes an image, more comparisons are made: **1.** A GPT-4o-based image-to-image comparison is conducted, yielding a similarity metric sim_{GPT} indicating whether the objects are the same. **2.** The RGB histogram feature similarity sim_{RGB} between the input $\boldsymbol{image}$ and the image of the carried object, denoted as $\boldsymbol{image}_c$, as follows.

For each channel (R, G, B), the pixel values ranging from 0 to 255 are divided into K intervals, and the number of pixels in each interval is counted. The histograms for each channel in an image are denoted as hist_R, hist_G and hist_B. The histogram for each channel is calculated below:

$$\text{hist}_C[i] = \sum_{(x,y)\in image} \mathbb{I}\left(i \cdot \frac{256}{K} \leq C(x,y) < (i+1) \cdot \frac{256}{K}\right) \tag{11}$$

where $C \in \{R, G, B\}$, i denotes the i-th interval and $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 if the condition is true and 0 otherwise.

After merging the three histograms and normalizing, we obtain the final histograms of $\boldsymbol{image}$ and $\boldsymbol{image}_c$ denoted as $\text{hist}_{\boldsymbol{image}}$ and $\text{hist}_{\boldsymbol{image}_c}$. Finally, the calculation of sim_{RGB} is based on the definition of $sim(\cdot,\cdot)$ as presented in (1).

TABLE I
SUCCESS RATE OF OBJECT QUERY ON THE OFFLINE MAP

Method	scene_1	scene_2	scene_3	scene_4	scene_5	average
Vlmap [23]	7/14	8/19	7/17	9/17	6/18	43.5%
Conceptgraph [22]	9/14	10/19	12/17	6/17	8/18	52.9%
Ours	**13/14**	**15/19**	**15/17**	**14/17**	**13/18**	**82.4%**

The determination of whether a carried object is the target object relies on a comprehensive evaluation of the values sim_{sbert}, sim_{GPT} and sim_{RGB}.

Based on our empirical findings, we define the following criteria to confirm target discovery:

- If either the textual similarity $sim_{sbert} < 0.5$ or the RGB similarity $sim_{RGB} < 0.4$, the object is not considered to be the target, and the robot continues exploring.
- Otherwise, the object is considered the target if: $(sim_{sbert} + sim_{RGB})/2 > 0.7$, or $(sim_{sbert} + sim_{GPT})/2 > 0.7$.

If the criteria are met, $F_{t+1} = 1$ to indicate task completion.

V. EXPERIMENTAL RESULTS

By conducting extensive simulations and real world experiments, we investigate the following key questions:

1. Do the carried-by relationship and text description features improve the accuracy of instance queries? (Section V-A)?
2. Does the dynamic update of the CRSG contribute to more efficient instance navigation (Section V-B)?
3. Is our CRSG-based navigation strategy effective in navigating to moved instances (Section V-C)?

Metrics: We report Success Rate (SR) and Success weighted by inverse Path Length (SPL). SPL measures the efficiency of a robot's path by comparing it to the shortest route from the starting point to the target object. If the robot fails to reach the target, the SPL is zero. Otherwise, SPL is the ratio of the shortest path length to the robot's actual path length, with higher values indicating better performance.

A. Multi-Type Query on the Offline Map

Baselines: **Vlmap** [23] and **Conceptgraph** [22] construct offline maps embedded with visual-language features for object queries and robot navigation, and are compared in terms of the accuracy of multi-type queries.

A total of 85 queries, covering different types of navigation instructions (semantic, instance, and demand-driven), were conducted across 5 scenes in Gibson [32], with each type accounting for 17.65%, 49.41%, and 32.94% of the total, respectively. The experimental results are presented in Table I, where the query success rate of **Ours** averages 82.4% and is the highest in all five scenes. Because in instance queries like "*a cup on the table*", **Ours**' CRSG records the carrying relationship between *cup* and *table*, allowing for precisely locating the instance. In contrast, **Vlmap** [23] and **Conceptgraph** [22] may identify the *table* instead of the *cup*. Additionally, **Ours** incorporates text features of caption descriptions for each instance in CRSG, enabling better differentiation between similar objects, such as *black cup* and *white cup*, and demonstrating superior performance in querying instances of specific colors. If the query command does not involve any relationship description, we retrieve objects solely based on SBERT feature similarity. Meanwhile, **Vlmap** projects CLIP features from 3D space to 2D grids, which can cause the

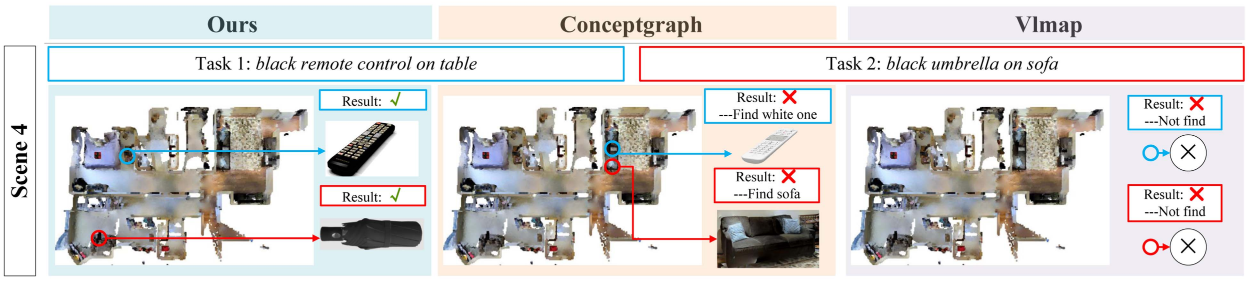

Fig. 4. Partial query Results on the Offline Map. Leveraging both carrier-carried relationships and SBERT-based textual similarity, **Ours** achieves more accurate instance retrieval compared to other methods.

TABLE II
SR(I) AND TASKS_SR(I) IN DIFFERENT SCENES FOR A SERIES OF LONG-HORIZON DAILY INSTANCES NAVIGATION TASKS (BEST SHOWN IN **BOLD**)

Scenes		**2**					**3**					**4**					**5**				
Methods		**Vlfm**	**Ofm-0.4**	**Ofm-0.55**	**Ofm-0.7**	**Ours**	**Vlfm**	**Ofm-0.4**	**Ofm-0.55**	**Ofm-0.7**	**Ours**	**Vlfm**	**Ofm-0.4**	**Ofm-0.55**	**Ofm-0.7**	**Ours**	**Vlfm**	**Ofm-0.4**	**Ofm-0.55**	**Ofm-0.7**	**Ours**
Object ID	**1**	40 40	20 20	0 0	20 20	**80 80**	20 20	0 0	20 20	40 40	**60 60**	0 0	0 0	0 0	40 40	**100 100**	0 0	**40 40**	20 20	**40 40**	**40 40**
	2	40 0	20 0	40 0	20 0	**100 80**	0 0	20 0	**80 20**	20 0	60 **20**	20 0	40 0	20 0	0 0	**100 100**	0 0	40 0	60 0	**80** 20	60 **40**
	3	20 0	40 0	40 0	40 0	**80 60**	40 0	0 0	0 0	20 0	**100 20**	0 0	40 0	20 0	0 0	**80 80**	0 0	80 0	60 0	80 20	**100 40**
	4	20 0	20 0	60 0	40 0	**80 40**	40 0	40 0	60 0	60 0	**100 20**	0 0	20 0	20 0	0 0	**100 80**	20 0	20 0	40 0	40 **20**	**60 20**
	5	0 0	20 0	40 0	20 0	**100 40**	0 0	60 0	40 0	60 0	**80 20**	0 0	20 0	20 0	0 0	**80 60**	0 0	40 0	40 0	0 0	**60** 0

loss of CLIP features for small objects. Additionally, its limited queryable semantic categories lead to inferior performance. We illustrate partial query results in Fig. 4, demonstrating that **Ours** excels at distinguishing between objects within the same category and accurately querying specific carried instances.

B. Long-Horizon Navigation for Everyday Instances

We conducted long-horizon navigation experiments in 4 everyday environments from the Gibson dataset [32], using the Habitat simulator [33]. Each task comprises exactly 5 sequential instance targets that the agent is required to reach in order. Each target is either a **displaced object**-whose position has changed relative to the initial CRSG, or **a new object** that was not present in the initial CRSG. For each scene, we designed 5 such long-horizon tasks, resulting in a total of 20 tasks across all four scenes. Specifically, in each scene, we placed commonly used items (e.g., *black cup*, *blue clock*) along with distractor objects of the same category (e.g., *black cup*, *white cup*) to test accurate navigation to specific instances, and constructed an offline CRSG. Before the navigation experiments, the positions of these items were randomly altered to reflect the variability in their locations. The robot was instructed to navigate to these instances sequentially.

Baselines: The latest online exploratory open-vocabulary navigation methods, **Vlfm** [13] and **Ofm** [25] are selected as baselines. For **Ofm** [25], it detects object semantics during navigation and records these semantics on a 2D grid map if the detection confidence exceeds a specific threshold. Due to the threshold's impact on navigation results, we conducted experiments with three thresholds: 0.4, 0.55 (official implementation), and 0.7, referred to as **Ofm-0.4**, **Ofm-0.55**, and **Ofm-0.7**, respectively. Besides, a variant, **Ours-w/o-u**, relying on the initial CRSG for each object navigation task, was compared with **Ours** in terms of SPL across all four scenes.

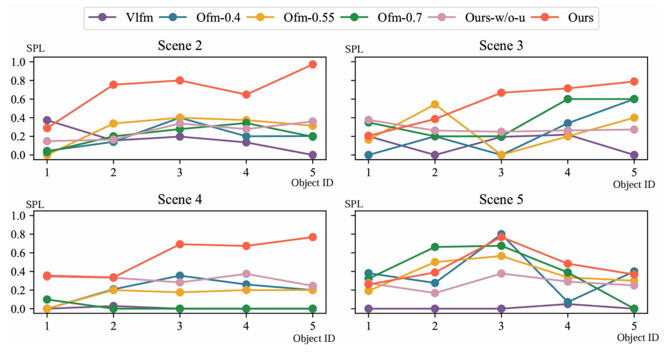

Fig. 5. SPL of different methods for long-horizon object navigation. The SPL of **Ours** generally increases as the object ID grows, reflecting the **positive impact of CRSG updates on navigation efficiency**. In Scene 5, the SPL of **Ours** exhibits a rise followed by a decline due to the observation blind spots and targets that are difficult to accurately distinguish, which lowers the SR and consequently reduces the SPL.

The navigation results for each scene are shown in Table II (SR(i) and Tasks_SR(i)) and Fig. 5 (SPL). SR(i) denotes the average navigation success rate for the i-th object. Tasks_SR(i) denotes the proportion of long-horizon tasks in which the agent successfully reached the first i subgoals in sequence without any failure. Table II shows that **Ours** achieves the highest SR and Tasks_SR(i), thanks to its ability to identify instance-level targets. By supporting image input and integrating text features, RGB features, and MLLM-based (GPT-4o) discrimination, **Ours** enables accurate navigation to the target instance. In contrast, both **Vlfm** and **Ofm** rely solely on text input. Specifically, **Vlfm** only supports navigation to general semantic categories, such as *bottle*, rather than more specific instances like *yellow bottle*, resulting in poor performance in correctly navigating to the target instance.

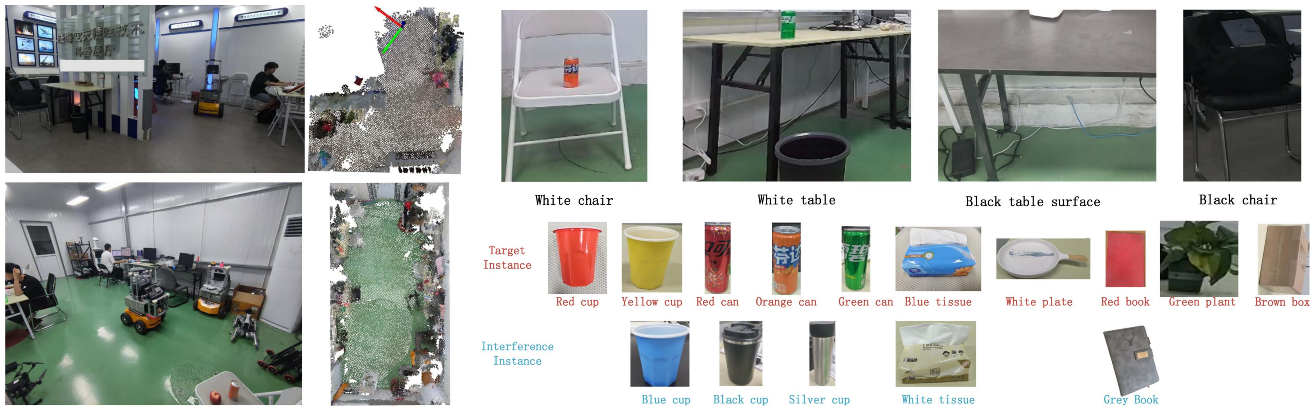

Fig. 6. Different scenes, carrying surfaces, and target instances in the real-world experiments.

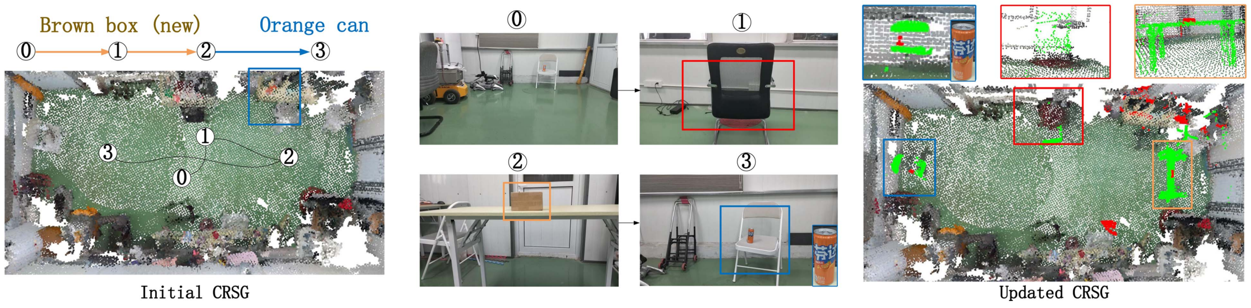

Fig. 7. A successful example of sequential navigation to two targets. The first target, a *brown box*, is a new object absent from the initial CRSG. In the initial CRSG, the **blue bounding box** marks the *orange can*'s initial position. In the updated CRSG, the **red bounding box** indicates a disappeared table, and the **yellow bounding box** shows a newly appeared table carrying the *brown box*.

In Fig. 5, the SPL of **Ours** steadily improve with the increasing object ID, as the robot captures changes in the scene and updates the CRSG throughout the long-horizon navigation task. As a result, the robot often completes subsequent navigation tasks without further exploration. This highlights the importance of CRSG updates and showcases our scene maintenance capabilities. In contrast, **Ours-w/o-u** executes each single-object navigation based on the initial CRSG, so it requires exploring the position of each target rather than efficient point-to-point navigation, resulting in consistently low SPL. **Vlfm** lacks scene maintenance, which prevents its SPL from improving as the long-horizon task progresses. Although **Ofm** has scene memory capabilities, its performance is limited by low success rates, leading to an SPL typically lower than **Ours**. Notably, in Scene 5, the SPL of **Ours** exhibits a rise followed by a decline, primarily due to observation blind spots and challenges in distinguishing visually ambiguous targets. These factors lead to low success rates, which in turn reduces the SPL. Some methods also exhibit similar trends.

C. Ablation Study

1) Different Criteria for Goal Determination: We conducted ablation experiments on long-horizon navigation tasks in scene 4 to assess the necessity of using GPT-4o, text features, and RGB histogram features to confirm navigation to the correct target. The results, shown in Table III, highlight that **Ours** consistently achieves the highest success rate. The absence of any component-**w/o GPT-4o**, **w/o text**, or **w/o RGB**-results in lower performance, demonstrating the importance of all three components for accurate target navigation.

TABLE III
ABLATION STUDY 1: SR(I) (%) / TASKS_SR(I) (%) USING DIFFERENT CRITERIA FOR GOAL DETERMINATION

Object	w/o GPT-4o	w/o text	w/o RGB	Ours
1	66.7 / 66.7	**83.3 / 83.3**	**83.3 / 83.3**	**83.3 / 83.3**
2	66.7 / 66.7	50.0 / 50.0	50.0 / 50.0	**83.3 / 83.3**
3	**66.7** / 50.0	**66.7** / 33.3	**66.7** / 33.3	**66.7 / 66.7**
4	**83.3** / 50.0	50.0 / 33.3	50.0 / 33.3	**83.3 / 66.7**
5	50.0 / 00.0	25.0 / 25.0	25.0 / 25.0	**75.0 / 50.0**

TABLE IV
ABLATION STUDY 2: SPL BY DIFFERENT NAVIGATION STRATEGIES

Metric	only-carriers_Random	only-carriers_LLM	Ours
SPL	0.205	0.309	**0.342**

2) Navigation Strategies: We also conducted single daily instance navigation experiments in three scenes to evaluate the impact of various modules in our navigation strategy on navigation efficiency. **only-carriers_Random** navigates to a randomly selected carrier object for exploration, without considering candidate target objects. **only-carriers_LLM** builds on this by selecting the next carrier object based on MLLM's recommendations. As shown in Table IV, **Ours** achieves the highest SPL, followed by **only-carriers_LLM**, demonstrating that navigating to candidate target objects and selecting carrier objects using MLLM's commonsense knowledge improves navigation efficiency.

D. Real-World Experiments

We conducted real-world experiments in two scenarios involving different carrying surfaces and target instances, as shown in Fig. 6. Everyday items were placed on tabletops to reflect realistic conditions. Experiments were carried out using an Autolabor robot, equipped with a Livox Mid-360 LiDAR for SLAM-based global pose estimation and an Azure Kinect DK for RGB-D data acquisition. Path planning and obstacle avoidance were handled using the ROS move_base package.

We identified three primary failure modes during real-world experiments. **First**, low-reflectivity surfaces and small objects often cause depth missing, leading to update failures. **Second**, motion-induced blur in RGB-D inputs leads to inconsistent object captions and impaired recognition. These issues do not arise in simulation, where the depth and RGB data are clean. To address this, we plan to explore a GPT-based text caption similarity comparison approach to enable more robust instance matching. **Third**, occlusions or blind spots can prevent target objects from being updated to the CRSG.

Despite these challenges, our system achieved success in various test cases. As illustrated in Fig. 7, we present an example of successfully navigating to two sequential target instances. Real-time CRSG updates significantly improved the robot's efficiency in navigating to the second target.

VI. Conclusion

This paper has proposed an open-vocabulary navigation method for frequently used everyday items, leveraging a dynamic carrier-relationship scene graph (CRSG). Specifically, we first construct the CRSG to capture the dynamic relationships between carrier-level objects and the objects they carry. Next, a navigation strategy based on the CRSG is developed to navigate to frequently used items, modeling the object search process as a Markov Decision Process. At each navigation step, the CRSG is dynamically updated based on the robot's observations of the environment. The robot then decides to navigate toward candidate target objects or unexplored carrier objects, guided by visual-language feature similarities and commonsense knowledge from the MLLM. Both simulations and real-world experiments demonstrate that our method efficiently navigates to objects that are subject to positional changes, even in the presence of distractors from the same category.

References

[1] J. Zhang et al., "3D-aware object goal navigation via simultaneous exploration and identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 6672–6682.

[2] Y. Deng et al., "Opengraph: Open-vocabulary hierarchical 3D graph representation in large-scale outdoor environments," *IEEE Robot. Automat. Lett.*, vol. 9, no. 10, pp. 8402–8409, 2024.

[3] M. Chang et al., "Semantic visual navigation by watching youtube videos," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 4283–4294.

[4] J. Ye et al., "Auxiliary tasks and exploration enable objectnav," 2021, *arXiv:2104.04112*.

[5] D. S. Chaplot et al., "Object goal navigation using goal-oriented semantic exploration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 4247–4258.

[6] H. Luo, A. Yue, Z. -W. Hong, and P. Agrawal, "Stubborn: A strong baseline for indoor object navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 3287–3293.

[7] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, "PONI: Potential functions for objectgoal navigation with interaction-free learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18890–18900.

[8] A. Rajvanshi et al., "Saynav: Grounding large language models for dynamic planning to navigation in new environments," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2024, pp. 464–474.

[9] N. M. M. Shafiullah et al., "Clip-fields: Weakly supervised semantic fields for robotic memory," 2022, *arXiv:2210.05663*.

[10] Z. Liang et al., "A survey of multimodel large language models," in *Proc. 3rd Int. Conf. Comput. Artif. Intell. Control Eng.*, 2024, pp. 405–409.

[11] J. Chen et al., "How to not train your dragon: Training-free embodied object goal navigation with semantic frontiers," *Robot.: Sci. Syst.*, 2023.

[12] K. Zhou et al., "ESC: Exploration with soft commonsense constraints for zero-shot object navigation," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 42829–42842.

[13] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, "VLFM: Vision-language frontier maps for zero-shot semantic navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 42–48.

[14] V. S. Dorbala, J. F. Mullen, and D. Manocha, "Can an embodied agent find your 'cat-shaped mug'? LLM-based zero-shot object navigation," *IEEE Robot. Automat. Lett.*, vol. 9, no. 5, pp. 4083–4090, May 2024.

[15] F. Garcia et al., "Markov decision processes," *Markov Decis. Processes Artif. Intell.*, pp. 1–38, 2013.

[16] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.

[17] Y. Tang, M. Wang, Y. Deng, Y. Yang, Z. Lan, and Y. Yue, "Multi-view robust collaborative localization in high outlier ratio scenes based on semantic features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 11042–11047.

[18] Y. Tang, M. Wang, Y. Yang, Z. Lan, and Y. Yue, "Robust large-scale collaborative localization based on semantic submaps with extreme outliers," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 4, pp. 2649–2660, Aug. 2024.

[19] X. Tian et al., "SAME: Ground-air collaborative semantic active mapping and exploration," in *Proc. Int. Conf. Unmanned Syst.*, 2024, pp. 1923–1930.

[20] Y. Deng, M. Wang, Y. Yang, and Y. Yue, "HD-CCSOM: Hierarchical and dense collaborative continuous semantic occupancy mapping through label diffusion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 2417–2422.

[21] Y. Tang, M. Wang, Y. Deng, Y. Yang, and Y. Yue, "SSGM: Spatial semantic graph matching for loop closure detection in indoor environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 9163–9168.

[22] Q. Gu et al., "ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 5021–5028.

[23] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 10608–10615.

[24] A. Werby et al., "Hierarchical open-vocabulary 3D scene graphs for language-grounded robot navigation," *Robot. Sci. Syst.*, 2024.

[25] Y. Kuang et al., "OpenFMNav: Towards open-set zero-shot object navigation via vision-language foundation models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2024, pp. 338–351.

[26] J. Huang et al., "IVLMap: Instance-aware visual language grounding for consumer robot navigation," 2024, *arXiv:2403.19336*.

[27] M. Chang et al., "Goat: Go to any thing," in *Proc. Conf. Robot. Sci. Syst.*, 2024.

[28] T. Pan et al., "Tokenize anything via prompting," in *Proc. Eur. Conf. Comput. Vis.*, 2023, pp. 330–348.

[29] N. Reimers and G. Iryna, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proc. 2019 Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process.*, 2019, p. 3982.

[30] T. Qin et al., "The maze method for extracting boundary lines." 2021. [Online]. Available: https://zhuanlan.zhihu.com/p/391392970

[31] L. Qi et al., "High-quality entity segmentation," in *Proc. 2023 IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 4024–4033.

[32] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9068–9079.

[33] M. Savva et al., "Habitat: A platform for embodied AI research," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9339–9347.