2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

CMAR-search: Commonsense and Memory Augmented Reasoning for Object Search in Dynamic Interactive Environments

Kaiyao Liao[1†], Qingfeng Li[1†], Xinlei Zhang[1], Chen Chen[2], Qing Sun[2], Jianwei Niu[1*]

***Abstract*— Dynamic interactive object search in large-scale human environments presents substantial challenges for existing methods. Current scene representations like 3D Scene Graphs (3DSG) only provide coarse-grained spatial segmentation and cannot identify functional areas such as storage or leisure areas. Without functional area understanding, existing methods are constrained to exhaustive sequential exploration at large scales, resulting in inefficient search behaviors—particularly in open-layout environments with numerous interactive objects such as drawers and cabinets. Moreover, these methods lack adaptability to environmental dynamics such as object relocations. To address these limitations, this paper proposes CMAR-search, a novel framework built upon Commonsense and Memory Augmented Reasoning (CMAR). Our approach leverages commonsense about area functionalities and aggregates environmental memory to construct a Functional 3D Scene Graph (F3DSG), which organizes the environment into functional areas with their associated containers. Through this structured representation, CMAR enables hierarchical action planning at both macro-area and micro-container levels, empowering the system to efficiently identify and inspect semantically relevant areas for effective object search. Notably, CMAR continuously integrates real-time perception, accumulated memory, and commonsense to dynamically relocalize objects in changing environments. Extensive experiments in simulation and real-world settings demonstrate that CMAR-search significantly surpasses state-of-the-art baselines in both success rate and search efficiency for object search in dynamic interactive environments.**

I. INTRODUCTION

In human-centered environments, interactive object search is a fundamental embodied AI task where a robot must locate target objects through navigation and physical interaction such as opening cabinets and drawers. This problem becomes particularly challenging in large-scale, dynamic settings that demand long-horizon reasoning and semantic understanding of spatial layouts. Current approaches typically extract environmental information into 3D Scene Graphs (3DSG) that structurally represent rooms and objects [1], [2], [3], [4], [5] and utilize Large Language Models (LLMs) for high-level reasoning of robotic actions [6], [7], [8], [9]. However, as environmental complexity increases, these methods encounter several critical issues: (1) In open-layout home environments, existing scene representation methods struggle to effectively segment functional areas, making it difficult to narrow down possible object search ranges; (2) their reliance on atomic actions—such as opening only one container at a time—considerably slows down search processes when handling multiple containers; and (3) they lack robust adaptation to environmental dynamics, showing limited integration of historical memory and object commonsense for efficient object relocation, which significantly restricts their real-world applicability.

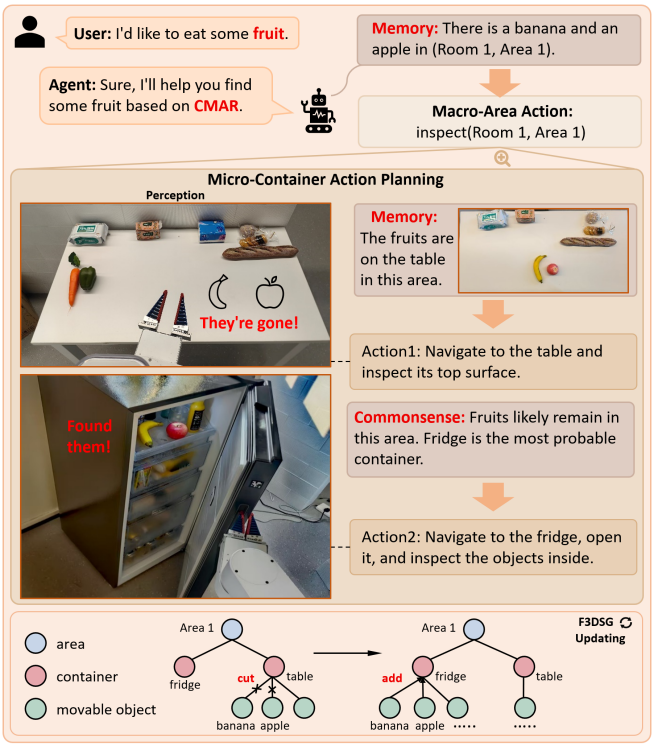

Fig. 1. Our CMAR-search framework empowers the agent to perform hierarchical action planning using the F3DSG, enabling rapid object relocation in dynamic environments.

Inspired by Kahneman's dual-system theory of cognition (System 1 for intuitive processing and System 2 for deliberate reasoning) [10], this paper proposes a novel framework centered on Commonsense and Memory Augmented Reasoning (CMAR). Our approach addresses the above limitations through two core innovations. First, in decision granularity, our Functional 3D Scene Graph (F3DSG), constructed using object-functional commonsense and spatial relationships, enables LLMs to perform macro-level planning using functional areas as units. The system directs LLMs to generate comprehensive search commands for entire areas at once, significantly reducing the number of LLMs calls and mitigating error accumulation. At the micro-level, these commands are decomposed into optimized sub-steps for inspect-

[1]School of Computer Science and Engineering, Beihang University
[2]Zhejiang Key Laboratory of Industrial Big Data and Robot Intelligent Systems, Hangzhou Innovation Institute of Beihang University
† These authors contributed equally to this work
* Corresponding author

ing all containers. Our CMAR-based sequence—considering distance, semantic similarity, and risk factors—determines the optimal execution order, further enhancing search efficiency. Second, in dynamic response, the CMAR mechanism synergistically retrieves historical memory and object-functional commonsense to infer optimal search strategies. This framework allows the robot to systematically explore new environments and adapt to changes in a human-like manner, leading to significant improvements in success rate and efficiency in dynamic interactive object search tasks.

To summarize, our main contributions are as follows:

- This paper proposes CMAR-search, a novel framework for dynamic interactive object search, which introduces CMAR to enable adaptation to environmental dynamics (e.g., object relocations) by synergistically leveraging real-time perception, historical memory, and commonsense, significantly improving search robustness in changing environments.
- The framework constructs an F3DSG that organizes environments into functional areas with their associated containers, enabling hierarchical action planning at both macro-area and micro-container levels to efficiently identify and inspect semantically relevant areas, reducing unnecessary exploration in large-scale environments.
- Extensive experiments in both simulated and real-world environments demonstrate that the proposed method significantly outperforms state-of-the-art baselines in success rate and efficiency for dynamic interactive object search tasks.

II. RELATED WORKS

A. 3DSG-based Environmental Representation

3DSG as a sparse environmental representation can abstract hierarchical, object-centric models of environments from dense perceptual data, thereby providing a powerful interface for high-level reasoning and task planning in robotics [4], [5], [11]. Existing research has constructed hierarchical graphs such as "building–floor–room–object" and enhanced their representational capacity by incorporating object-object relationship edges [12], [13], [14], integrating Voronoi graphs for navigation support [15], or distinguishing between static and dynamic objects [5]. There are also some methods that try to implement dynamic updates for 3DSG [2], [16]. However, most of these methods rely on a pre-built, static scene graph representation, which inherently lacks the ability to dynamically evolve based on interactive experiences (e.g., inspecting container interiors) and fails to excavate the semantically critical layer of "functional areas" essential for task planning. In contrast, our work aims to construct a functional and dynamically updatable 3D scene graph, whose core innovation lies in leveraging LLMs to infer functional area segmentation beyond purely geometric space, thereby laying the foundation for efficient interactive object search.

B. LLM-driven Robotic Planning

LLMs have been widely used to generate behavioral sequences for robots, achieving remarkable progress in tabletop manipulation and vision-language navigation (VLN) tasks [17], [18]. Some studies have attempted to apply LLMs to more challenging mobile manipulation tasks in home environments, such as generating plans through information retrieval for task matching [18], fine-tuning LLMs to encode object relationships [19], or iteratively pruning scene graphs [5]. MoMa-LLM [1] encodes scene graphs and instructions into natural language inputs for LLMs to generate next-step actions, enabling a certain level of interactive environmental information utilization. However, existing methods generally suffer from limited abstraction levels and single-grained instruction generation, making it difficult to adapt to the complex and diverse interactive reasoning demands in large-scale environments. This study significantly enhances the reasoning efficiency and generalization capability of LLMs in interactive tasks by introducing a hierarchically abstracted action space and structured task representation.

C. Object Search

Traditional object search methods include heuristic approaches based on frontier exploration [20] and reinforcement learning models [21]. With the increasing application of scene graphs, graph neural networks (GNNs) have been used to search for objects with relational constraints within graphs [22] or to cope with dynamic changes [23], [24]. Recent studies have employed LLMs to assess semantic similarity for guiding search [25], [26], but these are still limited to non-interactive navigation tasks. Schmalstieg et al. [27] explicitly proposed the interactive object search task, emphasizing operations on containers to find interior objects. MoMa-LLM [1] further supports interactive search using scene graphs but lacks adaptability to environmental changes. We propose the CMAR mechanism, integrating real-time perception, historical memory, and functional commonsense, to enable efficient and robust interactive object search in dynamic environments.

III. METHOD

Our proposed CMAR-search framework tackles object search in dynamic interactive environments, as shown in Figure 2. We assume access to ground truth perception for semantic masks, depth, localization, and handle detection, focusing on the reasoning aspect. The method comprises three components: (1) construction of F3DSG that incorporates functional area segmentation and container-movable object hierarchies; (2) a hierarchical action space for macro-area and micro-container planning; and (3) a dynamic response mechanism leveraging real-time perception, memory, and commonsense.

A. F3DSG Construction

We first acquire foundational scene information through perception and a door-based room separation approach. This provides the structural basis for hierarchical representations

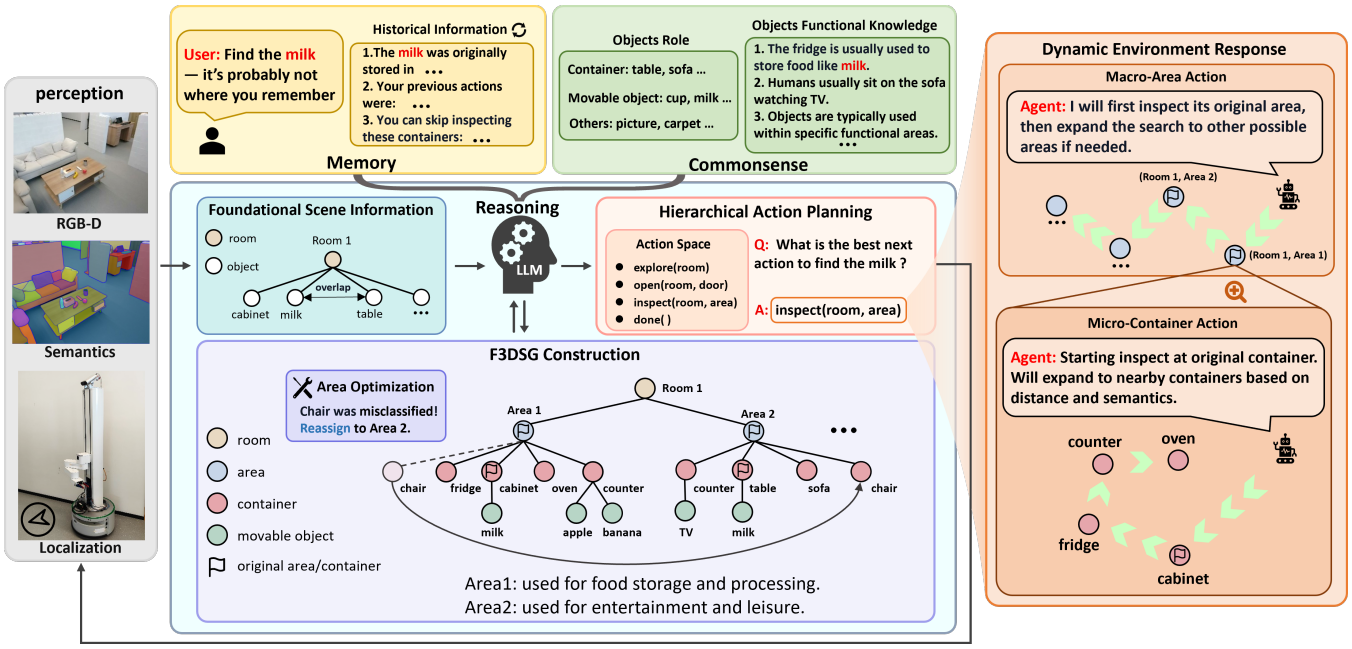

Fig. 2. Our CMAR-search framework continuously accumulates environmental memory and leverages commonsense to augment a hierarchical reasoning process across three stages: First, memory and commonsense augment the construction of a F3DSG that organizes the environment into functional areas with their associated containers; Second, F3DSG enables hierarchical action planning at both macro-area and micro-container levels, efficiently identifying and inspecting semantically relevant areas; Third, when environmental dynamics occur (e.g., object relocations), the framework augments relocalization by integrating historical memory with commonsense to prioritize original areas, then inferring alternative search paths based on functional semantics when necessary.

of rooms and objects, along with Voronoi graphs suitable for robot navigation [1]. Building upon this structural foundation, we further enrich the scene representation through function-aware reasoning to construct the F3DSG using the following method.

Container Layer and Movable Object Layer: We classify all objects in the scene (set $\mathcal{O}$) into containers $\mathcal{C}$ and movable objects $\mathcal{M}$ using LLMs. Two objects are considered to have a potential support relationship, and the pair is added to the set $\mathcal{I}$ if they satisfy: (1) overlap on the XY-plane, and (2) are close or overlapping on the Z-axis. If a pair (c, m) satisfies Eq. (1), it forms a valid container-movable object pair and is included in the set $\mathcal{V}$. All other containers not satisfying this relationship are classified as isolated containers $\mathcal{C}_i$, while remaining movable objects are classified as other objects $\mathcal{S}$.

$$V = \{(c, m) \in I \mid c \in \mathcal{C} \wedge m \in \mathcal{M}\} \tag{1}$$

Area Layer: We input all containers' names, spatial coordinates, and their carried movable objects (using `None` for isolated containers $\mathcal{C}_i$) to the LLMs. The LLMs then perform area clustering based on both spatial proximity (distance metrics) and semantic relationships, infers each area's functionality, and provides explicit reasoning for this division. The prompt template for this LLMs interaction is shown in Figure 3. Then, we propose an optimization method for LLM-generated area partitions. First, we reduce the number of areas by merging single-container areas to decrease the complexity of subsequent actions planning. Second, we correct positional classification errors by reassigning containers with abnormal spatial distributions to more appropriate areas. The formal implementation follows these rules:

$$\Phi_1(c) \coloneqq (|A_i(c)| > 1 \wedge \exists c' \in A_i(c), d(c, c') < \sigma) \tag{2a}$$

$$\Phi_2(c) \coloneqq (|A_i(c)| = 1 \wedge \forall c', d(c, c') \geq \sigma) \tag{2b}$$

$$A_o(c) = \begin{cases} A_i(c) & \text{if } \Phi_1(c) \vee \Phi_2(c) \\ \arg\min\limits_{A \in \mathcal{A}} \min\limits_{c' \in A} d(c, c') & \text{otherwise} \end{cases} \tag{2c}$$

Where Φ_1 and Φ_2 are conditions for preserving initial assignments: Φ_1 applies to containers in multi-container areas, and Φ_2 to all areas. $A_i(c)$ is the initial area of container c, $A_o(c)$ the optimized one, and $|A_i(c)|$ the number of containers in c's area. $\mathcal{A}$ is the set of all areas, $d(c, c')$ the Euclidean distance between c and c', and σ the distance threshold ($5\,\mathrm{m}$ in simulation, $3\,\mathrm{m}$ in real-world). The argmin operator selects the area $A \in \mathcal{A}$ containing c's nearest neighbor. To eliminate the influence of container processing order on optimization results, we adopt a two-phase optimization strategy: the first phase collects optimization decisions for all containers, while the second phase executes all decisions simultaneously. Post-optimization, the function of multi-container areas remains unchanged despite individual container additions or removals. Single-container areas will be deleted if merged into adjacent multi-container areas. When multiple single-container areas merge, the area with the smallest index is retained and its function is set to None. We find that a single iteration is sufficient to achieve optimal outcomes. After optimization, we associate objects in S to their nearest

Systems: You are a reliable agent equipped with rich human-like commonsense.

User: you need to group the following containers into distinct areas based on their locations and semantic relationships.
Output Format:
Area 1:
- Containers: [list of container names]
- Function: [description of the area s function] area (e.g., "This area is used for cooking").
- Reasoning: [explanation of why these containers belong together]
Containers in Room 1:

Name	Coordinate	Carried Movable Object
armchair_1	(50, 212)	pen_1
armchair_2	(49, 188)	None
coffee_table_1	(49, 201)	pen_2
coffee_table_2	(72, 195)	notebook_1
fridge_1	(142, 206)	None
kitchen_counter_1	(200, 209)	banana_1
kitchen_counter_2	(210, 199)	hand_towel_1
cabinet_1	(140, 200)	orange_1, yogurt_1

Agent:
Area 1:
- Containers: **[armchair_1, armchair_2, coffee_table_1, coffee_table_2]**
- Function: This area is used for seating and casual activities like writing or reading.
- Reasoning: These containers are close to each other (positions around (50, 200)) and include objects like pens and a notebook, suggesting a seating or lounging area.
Area 2:
- Containers: **[fridge_1, kitchen_counter_1, kitchen_counter_2, cabinet_1]**
- Function: This area is used for food storage and appliance placement.
- Reasoning: These containers are near the fridge (positions around (140-210, 200-238)) and include objects like a banana and a hand towel, suggesting a food storage and appliance area.

Fig. 3. Area Partitioning Prompt: Based on the container names, spatial coordinates, and carried movable objects, the LLMs are instructed to perform area clustering and infer the functional attributes of each area.

object's corresponding area.

B. Hierarchical Action Planning

To enable the LLMs to fully leverage CMAR for efficient task planning, we encode the semantic and geometric information from the scene graph into structured text representations [1]. Our framework features a hierarchical action space that operates at both macro and micro levels, significantly reducing decision complexity and improving search efficiency.

We adopt the method from MoMa-LLM [1] to construct a bird's-eye-view (BEV) occupancy map and generate a Voronoi graph for navigation based on free space. Frontiers are defined as the boundaries between explored and unexplored regions in the BEV map, guiding the robot's active environmental exploration. For each object in the scene, its closest Voronoi node is also computed to facilitate robot navigation and manipulation. After successfully navigating or opening a container, the robot gains full observability of all objects carried by it. Based on this perceptual capability, we design a macro action space as follows:

- *explore(*`room_name`*)*: Navigates to an unexplored frontier point in the specified room.
- *open(*`room_name`*,* `door_name`*)*: Navigates to a door, opens it, and enters the space behind.
- *inspect(*`room_name`*,* `area_name`*)*: Navigates to each container in the area following the inspection sequence for observation. If the container is closed and operable, performs an opening operation after navigation.
- *done()*: Terminate the episode when either of the following conditions is met: (1) successful target object localization, or (2) consecutive action execution failures. Throughout operation, the system maintains continuous real-time object detection, immediately terminating the current task upon visual acquisition of the target.

Systems: You are a robot in an unexplored house. Your task is to find a sugar-jar. You have the following actions available that you can use to achieve this task:
1. explore(room_name): explore the unknown space near one of the rooms that is not fully explored yet.
2. open(room_name, door): go to this door and open it.
3. inspect(room_name, area): reveal the carried objects of all containers in the area.
4. done(): call when the task is completed.
Output Response Format:
Analysis: describe where you could find the objects of interest and what actions you need to execute to get there.
Reasoning: justify why the next action is important to solve the task.
Command: function call.

User: you have found the following rooms and areas in the house so far:
Room 1:
Area 1: This area is used for seating and casual activities like writing or reading.
Containers: [armchair_1, armchair_2, coffee_table_1, coffee_table_2]
Movable objects: [pen_1, pen_2, notebook_1] others: [picture_1, floor_lamp_1]
Area 2: This area is used for food storage and appliance placement.
Containers: [fridge_1, kitchen_counter_1, kitchen_counter_2, cabinet_1]
Movable objects: [apple_1, orange_1, hand_towel_1] others: [None]
Your 1 previous actions were: explore(Room 1) – success.
These areas you can inspect: [(Room 1, Area 1)(very close), (Room 1,Area 2)(near)].
These rooms have unexplored space:[Room 1].
These rooms contain closed doors that might open up new space:[Room 1].
What is the best next action to complete the task as efficiently as possible?
Remember:
1. Respond with a function call and the parameter must be selected from the given list.
2. If some actions failed repeatedly, they may not be possible.

Agent:
Analysis: The sugar-jar is likely to be found in a storage area or kitchen, we can inspect the Area2 in Room 1.
Reasoning: inspecting the storage area in Room 1 is the most efficient next step because it is nearby and contains containers that could hold the sugar-jar.
Command: inspect(Room 1, Area 2)

Fig. 4. Macro Action Planning Prompt: Provide information about inspectable areas to guide the LLMs in determining whether there are areas that should be inspected or whether exploration-related instructions should be executed.

We maintain a container blacklist system where, after completing the inspection of a particular area, all containers in that area are added to the blacklist. This mechanism directly supports our hierarchical approach in the inspect action, which decomposes a macro-area command into optimized micro-container operations. For each container within the target area, we compute a composite selection score that balances path distance, semantic relevance, and search risk:

$$\tilde{D}_i = 1 - \frac{D_i - D_{\min}}{D_{\max} - D_{\min}} \tag{3a}$$

$$\tilde{P}_i = \frac{P_i - P_{\min}}{P_{\max} - P_{\min}} \tag{3b}$$

$$S_i = \alpha \tilde{D}_i + \beta \tilde{P}_i + \gamma \prod_{k \neq i} (1 - P_k) \tag{3c}$$

$$c^* = \arg\max_{c_i \in \mathcal{R}} S_i \tag{3d}$$

where D_i denotes the A* path distance, initially computed from the robot's current position to container c_i's nearest Voronoi node and subsequently calculated between consecutive containers' Voronoi nodes. P_i represents the SBERT-derived [28] cosine similarity between container c_i's name and the target object description. $D_{\max}$ and $D_{\min}$ are the

maximum and minimum A* distances among remaining containers $\mathcal{R}$, defined as all containers excluding blacklisted containers and previously selected containers. $P_{\max}$ and $P_{\min}$ are the extreme SBERT similarity values within $\mathcal{R}$; while $\tilde{D}_i$ and $\tilde{P}_i$ are their normalized counterparts in $[0, 1]$. The term $\prod_{k \neq i}(1 - P_k)$ represents the joint probability that the target is not present in any other remaining container except c_i, serving as a risk factor that encourages exploration of containers which, if skipped, might lead to complete search failure. S_i is the composite selection score; α, β, γ are weighting coefficients ($\alpha = 0.4, \beta = 0.4, \gamma = 0.2$); with c^* being the optimal container selected at each step.

We employ a hierarchical "room-area" scene representation to organize information, where each area is described in detail including its functional attributes, containers, movable objects, and other objects. To improve planning efficiency, the system automatically filters out areas where all containers are blacklisted. Since areas are dynamically constructed, our action history only retains two types of operational records: *explore* and *open*. For LLMs decision-making, we constrain the action selection space with prompts like "These areas you can inspect:" , while providing proximity information between the robot's current position and each area. This distance is calculated using the A* path distance between the robot and the nearest container in each area. Additionally, the system incorporates other output constraints including chain-of-thought reasoning, with complete prompt examples shown in Figure 4.

C. Dynamic Environment Response

We propose a human-object-interaction-inspired heuristic target search method to address the challenge of object search in partially explored, dynamic environments. Unlike traditional settings where the environment is assumed to be static and fully known, we focus on more realistic scenarios where objects may be relocated after the initial exploration, and the robot must efficiently re-find them based on limited prior knowledge. Motivated by the observation that movable objects in daily life typically remain within specific functional areas (e.g., a TV remote often moves between the table and sofa in the relaxation area), we develop a hierarchical processing pipeline using CMAR that operates at both area and container levels to enable efficient and robust target relocalization.

Area Level: The robot leverages its environmental memory to first locate the target object's initial area within the F3DSG and navigates to that area. For multiple candidate areas, the system prioritizes inspection based on the A* path distance from the robot to the nearest container in each area. If the target is not found in these initial areas, the system employs the LLMs to infer potential relocation areas based on commonsense knowledge of human object usage habits.

Container Level: When the target area contains original container(s) where the object was previously stored, our system prioritizes inspecting these containers. During inspection, if any changes in movable objects placed on containers are detected, the F3DSG is dynamically updated to reflect the current scene configuration. For the remaining containers beyond the initial ones, the inspection order follows the optimization criteria defined in Eq. (3) ensure efficient search.

IV. EXPERIMENTS

A. Experiment Setup

Baselines. We employ DeepSeek-V3 as our language model for its practical performance-cost balance. We compare our approach against heuristic-based, recent learning-based, and language-based methods.

Random: Uniform random selection among available actions (frontiers and closed objects).

Greedy: Selects the closest actionable target via A* planning in the explored map.

ESC-Interactive: Extends ESC by scoring both frontiers and operable objects using object-room co-occurrences from a fine-tuned language model [25].

HIMOS: Hierarchical RL approach adapted to large scenes by restricting navigation to target/articulated objects [27].

Unstructured LLM: Provides raw JSON scene graphs to LLM without structured encoding (adapted from SayPlan [5]).

MoMa-LLM: Current state-of-the-art using dynamic scene graphs with LLM-grounded room segmentation and open-vocabulary classification [1].

MoMa-LLM w/ Hydra: Variant using Hydra's geometric constriction method [3] instead of door-based segmentation.

Metrics. We use the following metrics to evaluate the performance of our method in scene understanding and multiple dynamic interactive object search tasks.

Accuracy: The proportion of correctly predicted containers among all actual containers, measuring overall prediction correctness.

Precision: The proportion of predicted containers that are true containers, measuring the reliability of predictions.

Recall: The proportion of actual containers that are successfully predicted, evaluating the ability to identify all containers.

F1-Score: The harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives.

Voronoi Graph Purity: A measure of the alignment between Voronoi graph components and ground-truth room segmentation, defined as:

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{k} \max_{j} |\omega_k \cap c_j| \tag{4}$$

where $\Omega = \omega_1, \omega_2, \ldots, \omega_K$ denotes the Voronoi components, $\mathbb{C} = c_1, c_2, \ldots, c_J$ represents the ground-truth rooms, and N is the total number of nodes. The $\max$ operation identifies the best-matching room for each component [1].

Room Classification Accuracy: The accuracy of predicting room types based on object distributions within Voronoi components, compared against ground-truth labels [1].

Object Interactions (OI): The average number of opening operations performed by the agent per episode.

TABLE I

CONTAINER CLASSIFICATION EVALUATION

Approach	Accuracy ↑	Precision ↑	Recall ↑	F1-Score ↑
OpenIN	0.648	0.972	0.648	0.778
Ours	**0.957**	**0.982**	**0.957**	**0.969**

TABLE II

ROOM SEGMENTATION EVALUATION

Approach	Voronoi graph purity ↑	Room Classification Accuracy ↑
Hydra	0.562	-
MoMa-LLM	0.615	0.276
Ours	**0.911**	**0.593**

TABLE III

INTERACTIVE OBJECT SEARCH RESULTS

Model	SR ↑	SPL ↑	OI ↓	DT ↓
Random	93.1	50.2	5.7	32.9
Greedy	85.7	50.9	8.1	22.3
ESC-Interactive	95.4	62.7	4.1	19.6
HIMOS	93.7	48.5	4.8	35.9
Unstructured LLM	86.3	59.4	3.6	18.5
MoMa-LLM w/ Hydra	92.0	61.9	<u>2.7</u>	12.9
MoMa-LLM	97.7	63.6	3.9	18.2
Ours	**98.9**	**79.0**	2.9	<u>9.8</u>
MoMa-LLM w/ F3DSG	94.9	76.1	**2.1**	**8.9**
Ours w/o area optimization	**98.9**	75.1	3.2	12.8
Ours w/ random order	**98.9**	<u>77.8</u>	3.0	10.6
Ours w/ greedy order	<u>98.3</u>	74.9	3.1	11.5

Distance Traveled (DT): The average distance in meters traveled by the agent per episode from its initial position until the target object is located.

Success Rate (SR): The proportion of episodes in which the agent successfully finds the target object.

Success Weighted by Path Length (SPL): A metric that evaluates efficiency by comparing the traveled path to the shortest possible path, weighted by success. It does not account for the cost of object interactions [29].

B. Simulation Experiments

We instantiate the task in the iGibson simulator [30] using a Fetch robot, following the iGibson challenge's data split with eight training scenes for module development and prompt engineering, and seven test scenes. In the scene, objects are categorized as movable and non-movable. During scene initialization, movable objects are placed on non-movable objects that meet volume requirements and exhibit certain semantic relevance [1]. In the dynamic environment response phase, only movable objects are selected as targets, and their new positions are constrained to other semantically and physically compatible non-movable objects.

Scene Understanding: We evaluated the performance of the proposed method in scene understanding across all test scenarios. Using all non-movable objects in the dataset as ground truth for the container category (after excluding special objects such as pictures and windows), we compared the container/non-container classification results obtained by our method and the OpenIN's approach [2]. As shown in TABLE I, our method achieves superior performance across all four evaluation metrics, validating the accuracy of our container-layer construction.

Since there is no standardized evaluation method for our area segmentation (Area Layer construction), we evaluate our approach's scene understanding capability by comparing it with two room segmentation baselines: Hydra's geometry-based method [3] that detects narrow passages for segmentation, and MoMa-LLM's semantic approach [1] that uses door positions as partitioning boundaries. Our method adopts a two-stage approach: first, we perform initial room segmentation using door positions similar to MoMa-LLM; then, for rooms that are potentially under-segmented (containing over five containers, typically indicating open-layout spaces), we adapt our Area Layer construction method with modified prompts to further subdivide them into smaller rooms and infer their categories based on spatial proximity and semantic relationships of containers. Each Voronoi node is then connected to its nearest container area node, constructing the separated Voronoi graph for segmentation and classification evaluation. As TABLE II demonstrates, our approach surpasses both baselines in Voronoi purity and area classification accuracy, effectively solving the under-segmentation problem that baseline methods encounter in open-layout spaces.

Interactive Object Search: The task involves locating target objects in completely unexplored interactive environments. For each test scene, we evaluate the agent through 25 procedurally generated episodes with randomized initial positions, target objects, and object distributions. As shown in TABLE III, our method demonstrates significant improvements over all baseline approaches, achieving a 24.2% higher SPL while maintaining competitive SR. Although our hierarchical action planning requires opening all operable containers during area inspection, the precision of our functional area judgment confines this process to a minimal set of candidate regions. This design yields fewer total object interactions than most baselines, demonstrating a superior balance between comprehensive search and operational efficiency. To further validate the superiority of our scene representation, we substituted MoMa-LLM's scene graph with our F3DSG while keeping other components unchanged. This replacement not only maintained the high SR but also achieved significant improvements in SPL, OI, and DT, unequivocally demonstrating that F3DSG enhances exploration efficiency. In ablation studies, we removed the area optimization module and replaced the inspection order with random and greedy strategies (always selecting the nearest container at each step). These variants exhibit degraded performance in SPL, OI, and DT.

Dynamic Environment Response: We design two tasks to evaluate the proposed method's adaptability in dynamic environments. Novel Target Search (NTS): a target object

TABLE IV

NOVEL TARGET SEARCH RESULTS

Model	Level 1		Level 2		Level 3		Average	
	SR	SPL	SR	SPL	SR	SPL	SR	SPL
MoMa-LLM	70.5	61.3	72.4	63.6	55.2	51.8	66.0	58.9
Ours	**97.1**	77.9	**95.2**	**80.8**	**91.4**	**81.2**	**94.6**	**80.0**
MoMa-LLM w/ F3DSG	85.7	75.5	74.3	65.6	61.9	53.2	74.0	64.8
Ours w/o area optimization	94.3	**78.9**	91.4	79.6	88.6	76.4	91.4	78.3
Ours w/ random order	96.2	76.1	**95.2**	78.7	**91.4**	72.8	94.3	75.9
Ours w/ greedy order	**97.1**	74.3	**95.2**	79.8	**91.4**	71.6	**94.6**	75.2

TABLE V

TARGET RELOCALIZATION RESULTS

Model	Level 1		Level 2		Level 3		Average	
	SR	SPL	SR	SPL	SR	SPL	SR	SPL
MoMa-LLM	81.0	71.2	85.7	76.8	84.8	72.2	83.8	73.4
Ours	**98.1**	**84.9**	**98.1**	**86.4**	97.1	**88.3**	**98.1**	**86.5**
MoMa-LLM w/ F3DSG	88.6	73.0	88.6	75.5	86.7	72.0	88.0	73.5
Ours w/o area optimization	96.2	82.4	97.1	77.2	**98.1**	87.8	97.1	82.5
Ours w/ random order	97.1	79.2	97.1	78.9	**98.1**	88.1	97.4	82.1
Ours w/ greedy order	**98.1**	78.1	**98.1**	81.2	96.2	82.8	97.5	80.7

with variable locations is randomly generated in a partially known scene, testing the system's ability to discover and localize new objects under partial prior knowledge. Target Relocalization (TR): a known object is selected from the robot's memory, its position is changed, and the robot must relocate it, assessing its perception of environmental changes and memory updating. To comprehensively evaluate robustness, we define three levels based on the known information of interactive container contents: Level 1 (Fully Unknown), where container contents are completely unknown; Level 2 (Semi-Known), where half of the container contents are known; and Level 3 (Fully Known), where all container contents are known, simulating scenarios with varying degrees of observable content. For experimental control, we assume full exploration of non-interactive parts of the scene, though our method applies to any exploration level.

For each scene and level, we conduct 15 episodes with randomized initial knowledge. We use MoMa-LLM [1] as the baseline, where for the TR task, the robot first navigates to the original target location and upon detecting a change, applies MoMa-LLM to reason about the subsequent action. Results for NTS and TR are shown in TABLE IV and TABLE V, respectively. Our method demonstrates substantial improvements over the baseline, achieving 43.3% higher SR and 35.8% higher SPL in NTS tasks, and 17.1% higher SR and 17.8% higher SPL in TR tasks. Failure analysis reveals that MoMa-LLM lacks systematic environment organization, relying on navigating to objects potentially related to the target, leading to excessive attempts and significantly reduced SR and SPL. In some cases, it prematurely terminates search upon concluding the target is unreachable. Replacing its scene representation with F3DSG improves performance, but limitations in the action space still prevent thorough inspection of specific areas. Our method leverages the CMAR mechanism to effectively improve the efficiency of both NTS and TR by fully utilizing memory and object usage commonsense. Ablation studies further validate the contribution of each component to the overall performance.

TABLE VI

REAL-WORLD EXPERIMENTS RESULTS

Model	IOS		NTS		TR	
	OI	DT	OI	DT	OI	DT
MoMa-LLM	1.8	12.6	1.5	10.5	1.9	12.1
Ours	**1.1**	**8.3**	**0.9**	**5.3**	**0.8**	**6.7**

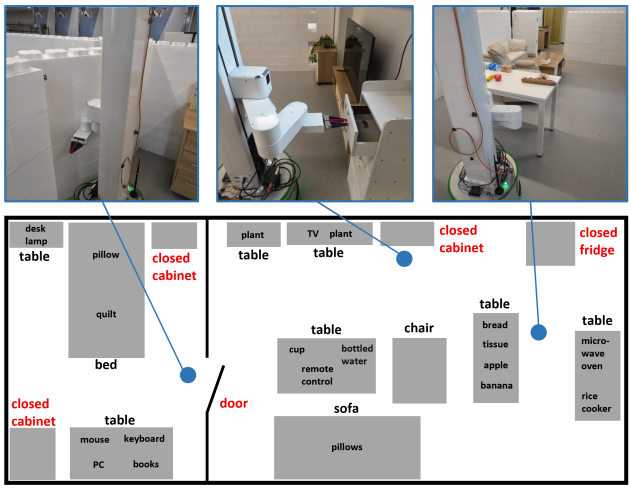

Fig. 5. Real-world household experimental environment. Interactive objects are labeled in red.

C. Real-World Experiments

We construct a real-world household environment with two distinct rooms, as illustrated in Figure 5: an open-layout room combining the kitchen and living room functions (with no internal doors), and a separate bedroom. The visible regions of the scene were first mapped using an existing method [12] to extract object information. In the Interactive Object Search (IOS) task, the robot was incrementally provided with map information from areas covered by its camera during operation. For both NTS and TR tasks, the system is granted complete information about non-interactive parts of the scene (equivalent to Level 1 in simulation, where container contents are completely unknown).

We perform 10 episodes for each task category, including open-vocabulary instructions such as "I am hungry, find me some food" and "I am thirsty, find me a drink". An episode is considered successful if any object relevant to the instruction enters the robot's field of view. MoMa-LLM [1] is employed as the baseline method. Owing to the constrained scale of the environment and limited number of objects, both our method and the baseline achieve perfect success rates across all tasks. We therefore evaluate exploration efficiency using two metrics: the number of Object Interactions and the Distance Traveled. As summarized in the accompanying TABLE VI, our method consistently outperforms the baseline in all three types of tasks. The advantage is particularly evident in NTS and IOS tasks, where our CMAR-search framework effectively utilizes prior knowledge to efficiently adapt to real-world dynamic variations.

V. CONCLUSIONS

We propose CMAR-search, a novel framework for dynamic interactive object search that augments robotic reasoning through the synergistic integration of memory and commonsense. Our method constructs an F3DSG using LLM-based functional area inference and a hierarchical action space to enable efficient macro-area and micro-container planning. The CMAR mechanism integrates real-time perception, historical memory, and object-functional commonsense to dynamically adapt to environmental changes. Extensive experiments in simulation and real-world settings demonstrate that our approach outperforms baselines in scene understanding, success rate, efficiency, and robustness, providing a scalable solution for long-horizon interactive search tasks in dynamic human environments.

ACKNOWLEDGMENT

This work was supported by the Shenzhen Science and Technology Program [Grant No. CJGJZD20240729141702003] and the Beijing Institute of Computer Technology and Applications [Grant No. 2023WDZC02002]. The authors would also like to thank Shenzhen Pudu Technology Co., Ltd. for the support and resources provided, which were essential to the completion of this research. Finally, the authors acknowledge the use of GPT-4 for language polishing and grammatical corrections during the preparation of this manuscript.

REFERENCES

[1] D. Honerkamp, M. Büchner, F. Despinoy, T. Welschehold, and A. Valada, "Language-grounded dynamic scene graphs for interactive object search with mobile manipulation," *IEEE Robotics and Automation Letters*, 2024.

[2] Y. Tang, M. Wang, Y. Deng, Z. Zheng, J. Deng, S. Zuo, and Y. Yue, "Openin: Open-vocabulary instance-oriented navigation in dynamic domestic environments," *IEEE Robotics and Automation Letters*, vol. 10, no. 9, pp. 9256–9263, 2025.

[3] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3d scene graph construction and optimization," *Robotics: Science and Systems XVIII*, 2022.

[4] A. Werby, C. Huang, M. Büchner, A. Valada, and W. Burgard, "Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation," in *Robotics: Science and Systems XX*, RSS2024, Robotics: Science and Systems Foundation, July 2024.

[5] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. D. Reid, and N. Sünderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable task planning," in *Conference on Robot Learning*, 2023.

[6] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[7] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," *arXiv preprint arXiv:2306.15724*, 2023.

[8] B. Li, P. Wu, P. Abbeel, and J. Malik, "Interactive task planning with language models," *arXiv preprint arXiv:2310.10645*, 2023.

[9] Z. Ni, X. Deng, C. Tai, X. Zhu, Q. Xie, W. Huang, X. Wu, and L. Zeng, "Grid: Scene-graph-based instruction-driven robotic task planning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13765–13772, IEEE, 2024.

[10] K. Frankish, "Dual-process and dual-system theories of reasoning," *Philosophy Compass*, vol. 5, no. 10, pp. 914–926, 2010.

[11] E. Greve, M. Büchner, N. Vödisch, W. Burgard, and A. Valada, "Collaborative dynamic 3d scene graphs for automated driving," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11118–11124, IEEE, 2024.

[12] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull, "Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5021–5028, 2023.

[13] H. Yin, X. Xu, Z. Wu, J. Zhou, and J. Lu, "Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation," *Advances in neural information processing systems*, vol. 37, pp. 5285–5307, 2024.

[14] H. Yin, X. Xu, L. Zhao, Z. Wang, J. Zhou, and J. Lu, "Unigoal: Towards universal zero-shot goal-oriented navigation," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19057–19066, 2025.

[15] P. Wu, Y. Mu, B. Wu, Y. Hou, J. Ma, S. Zhang, and C. Liu, "Voronav: Voronoi-based zero-shot object navigation with large language model," *arXiv preprint arXiv:2401.02695*, 2024.

[16] Q. Li, X. Zhang, C. Chen, H. Zhao, and J. Niu, "Interaction-driven updates: 3d scene graph maintenance during robot task execution," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11933–11939, 2025.

[17] J. Gu, E. Stefani, Q. Wu, J. Thomason, and X. E. Wang, "Vision-and-language navigation: A survey of tasks, methods, and future directions," in *Annual Meeting of the Association for Computational Linguistics*, 2022.

[18] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2998–3009, 2023.

[19] G. Chalvatzaki, A. Younes, D. Nandha, A. T. Le, L. F. Ribeiro, and I. Gurevych, "Learning to reason over scene graphs: a case study of finetuning gpt-2 into a robot language model for grounded task planning," *Frontiers in Robotics and AI*, vol. 10, p. 1221739, 2023.

[20] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pp. 146–151, IEEE, 1997.

[21] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," *arXiv preprint arXiv:2004.05155*, 2020.

[22] M. Lingelbach, C. Li, M. Hwang, A. Kurenkov, A. Lou, R. Mart'in-Mart'in, R. Zhang, L. Fei-Fei, and J. Wu, "Task-driven graph attention for hierarchical relational object navigation," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 886–893, 2023.

[23] A. Kurenkov, M. Lingelbach, T. Agarwal, C. Li, E. Jin, R. Zhang, L. Fei-Fei, J. Wu, S. Savarese, and R. Martín-Martín, "Modeling dynamic environments with scene graph memory," in *Adaptive Agents and Multi-Agent Systems*, 2023.

[24] Z. Ying, X. Yuan, B. Yang, Y. Song, Q. Xu, F. Zhou, and W. Sheng, "Rp-sg: Relation prediction in 3d scene graphs for unobserved objects localization," *IEEE Robotics and Automation Letters*, vol. 9, pp. 1412–1419, 2024.

[25] K.-Q. Zhou, K. Zheng, C. Pryor, Y. Shen, H. Jin, L. Getoor, and X. E. Wang, "Esc: Exploration with soft commonsense constraints for zero-shot object navigation," in *International Conference on Machine Learning*, 2023.

[26] J. Chen, G. Li, S. Kumar, B. Ghanem, and F. Yu, "How to not train your dragon: Training-free embodied object goal navigation with semantic frontiers," *ArXiv*, vol. abs/2305.16925, 2023.

[27] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada, "Learning hierarchical interactive multi-object search for mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 8, pp. 8549–8556, 2023.

[28] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Conference on Empirical Methods in Natural Language Processing*, 2019.

[29] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[30] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. Vainio, C. Gokmen, G. Dharan, T. Jain, *et al.*, "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," *arXiv preprint arXiv:2108.03272*, 2021.