Language-Exclusive Mobile Manipulation for Efficient Object Search in Indoor Environments

Liding Zhang[1*], Zeqi Li[1*], Kuanqi Cai[1*], Zhenshan Bing[1], Alois Knoll[1]

***Abstract*— Enabling robots to efficiently search for and identify objects in complex, unstructured environments is critical for diverse applications ranging from household assistance to industrial automation. However, traditional scene representations typically capture only static semantics and lack interpretable contextual reasoning, limiting their ability to guide object search in completely unfamiliar settings—where every object's relational information is vital. To address this challenge, we propose a language-enhanced hierarchical navigation framework that tightly integrates semantic perception and spatial reasoning. Our method, Goal-Oriented Dynamically Heuristic-Guided Hierarchical Search (GODHS), leverages large language models (LLMs) to dynamically infer scene semantics through common sense knowledge. To ensure reliability, we employ a structured prompt design within a language model chain, imposing logical constraints on LLM inputs and outputs. For mobile manipulators, we further introduce a hybrid motion planner that combines polar angle sorting with distance prioritization, minimizing travel while maintaining directional consistency when selecting among multiple candidate poses. Comprehensive evaluations in Isaac Sim demonstrate that our approach significantly increases object search success rates in novel environments and outperforms conventional semantic navigation strategies in terms of path efficiency. Website and Video are available at: https://drapandiger.github.io/GODHS/.**

I. Introduction

When humans search for objects in unfamiliar environments, they rely on a hierarchical understanding of semantic information to rapidly localize potential object placements [1]. Inspired by this ability, our approach emulates the human strategy of leveraging semantic cues—e.g., "pillows often lie on beds"—to guide and constrain the search process. Rather than exhaustively scanning the entire space, humans focus on carriers that are most likely to hold the target, significantly reducing the search effort. Building on this human-inspired perspective, recent advances in mobile manipulation have equipped robots with near human-level perceptual and actuation capabilities—often referred to as "eye-hand-foot" coordination—thanks to integrated depth cameras and versatile end-effectors. Yet despite these technical improvements, most robotic systems still employ spatially driven strategies that overlook the semantic insights humans naturally rely on. Consequently, in a novel environment where the goal is to find a pillow, these systems tend to exhaustively scan every accessible corner rather than leveraging common-sense knowledge—e.g., that pillows typically rest on beds.

Most existing navigation systems also exhibit two major shortcomings. First, although they may incorporate basic semantic labels (e.g., "table" or "chair"), they often fail to reason about dynamic relationships between known objects and unknown targets, limiting their applicability in truly novel scenarios. Second, vision-language models (VLMs) have proven capable of aligning natural language with image features, yet they typically lack the broader world commonsense knowledge that large language models (LMs) can offer. In principle, LMs combined with semantic segmentation could replicate much of a VLM's functionality while delivering enhanced real-world contextual reasoning.

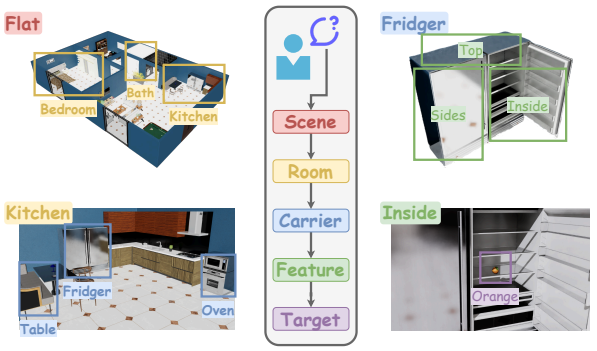

Fig. 1: The **GODHS framework** divides each scene into five strict levels: Scene, Room, Carrier, Feature, and Item. For instance, after entering the ***flat***, the robot uses mapping and semantic segmentation to determine and prioritize the types of rooms through a LM. For the first room ***kitchen***, the LM classifies and prioritizes carriers such as furniture or devices. For the first carrier ***fridge***, the LM identifies and prioritizes the feature regions worth searching. Finally, for the first feature ***inside***, the robot searches for the presence of the target object ***orange***.

To address these gaps, this paper introduces a *Goal-Oriented Dynamically Heuristic-Guided Hierarchical Search* (**GODHS**) framework, inspired by the way humans structure search behaviors. Building on the notion that people mentally decompose a search task—starting with the room (e.g., bedroom), moving to the carrier object (e.g., bed), then examining specific features (e.g., top surface), and finally verifying the target—GODHS employs large language models (LLMs) to orchestrate a five-level hierarchy: *scene*, *room*, *carrier*, *feature*, and *item*. As illustrated in Fig. 1, each level narrows the search scope through an LLM-driven process we term *Chain-of-Refinement* stepwise.

Implementing our hierarchical pipeline presents three interrelated inherent systemic challenges that GODHS directly addresses through corresponding innovations:

- **Dynamic Semantic Reasoning** The first challenge arises from the need to infer the likely locations of unknown targets based on known carriers (e.g., "cup on a table"). Our solution is the **Hierarchical Language Reasoning** framework, which formulates object search as a structured, five-tier decision process. By applying

[1]L. Zhang, Z. Li, K. Cai, Z. Bing and A. Knoll are with the Department of Informatics, Technical University of Munich, Germany. liding.zhang@tum.de

*These authors contributed equally to this work.

The authors acknowledge the financial support by the Bavarian State Ministry for Economic Affairs, Regional Development and Energy (StMWi) for the Lighthouse Initiative KI.FABRIK (Phase 1: Infrastructure and the research and development program under grant no. DIK0249).

Corresponding author: Zhenshan Bing.

LLM-informed priors (e.g., "An orange is likely inside the fridge in the kitchen") at each level, the system efficiently narrows down search regions.

- **LLM Reliability** Although large language models exhibit impressive reasoning capabilities, they can produce erroneous outputs without proper domain-specific safeguards—for instance, mistaking a "floor" for a viable carrier. To address this, we introduce a **Chained LM Architecture** that employs semantically logically constrained prompts in a sequential fashion (our *Chain-of-Refinement*). This structured approach ensures more consistent and accurate inferences.
- **Motion Planning Efficiency** A mobile manipulator typically faces multiple candidate base positions and end-effector orientations for exploring a single carrier, which can be time-consuming. To mitigate this, we propose **Hybrid Motion Planning**, combining polar angle sorting with lexicographical distance prioritization. This strategy allows the robot to quickly evaluate and select among competing trajectories, significantly reducing travel time and maintaining directional consistency.

II. Related Work

Robotic object search strategies have advanced through three evolving paradigms: geometric reasoning, vision-language integration, and language model augmentation. Early geometric approaches established foundational principles but faced critical limitations. Heuristic spatial affordance methods [2], [3] constrained search regions using spatial affordances, while probabilistic frameworks [4], [5] modeled uncertainty through Bayesian networks and graph-based optimizations. Though effective in structured environments, these methods required exhaustive object relationship priors [5], failing to infer latent functional associations, such as linking "medicine" to "cabinet" through functional rather than geometric proximity. POMDP [6], [7] improved dynamic decision-making but remained confined to predefined taxonomies, unable to adapt to novel object configurations.

The advent of vision-language models (VLMs) enabled open-vocabulary object recognition, yet introduced new constraints. Semantic mapping systems [8], [9] enriched spatial representations with object labels but treated semantics as static attributes, overlooking dynamic relational reasoning. Hierarchical graphs [10], [11] attempted to model object interactions through predefined hierarchies, yet their rigid structures required complete environmental pre-knowledge [12]. While VLM-integrated navigation [13], [14] achieved open-vocabulary target localization, it often overfitted to superficial visual-language correlations [15], struggling to chain contextual relationships. Reinforcement learning approaches [6], [16] enhanced adaptability but lacked mechanisms to hierarchically decompose search tasks.

Recent language model integrations reveal untapped potential with persistent gaps. LLM-augmented planners [13], [17] generate plausible search hypotheses but employ flat decision structures that exhaustively evaluate all object relationships [18]. Scene-graph methods [3], [19] capture object relationships dynamically but lack incremental refinement mechanisms, often requiring full environmental observability [20]. Open-vocabulary manipulation systems [21] leverage 3D semantic maps for generalization yet retain reactive paradigms, failing to proactively narrow search scopes through semantic chaining. Notably, while existing methods [22] achieve partial progress in language grounding, none successfully emulate the human cognitive process of hierarchical task decomposition, transitioning from room-level context to carrier object features through multi-stage refinement. Our work bridges this divide by synergizing LLM-driven commonsense reasoning with principled hierarchical action planning, avoiding both the inflexibility of geometric heuristics and the inefficiency of flat neural architectures.

III. Methodology

Our search approach follows a logical hierarchical progression to locate the ultimate target. Therefore, we need to construct a hierarchically expandable algorithm that fully utilizes perceived environmental information, existing knowledge, and integrates a language model to derive semantic sequences for subsequent layers (Sec. III-A). However, since semantic processing during search operates at the lexical level, the accuracy of responses directly impacts the success of downstream tasks. To prevent erroneous outputs, we establish a Chain-of-Refinement mechanism (Sec. III-B). To ensure the mobile robot advances sequentially around the carrier while enabling a single chassis pose to correspond to multiple end-effector configurations, we employ dictionary mapping and polar angle sorting (Sec. III-C).

A. GODHS Framework

Goal-Oriented Dynamically Heuristic-Guided Hierarchical Search is an efficient search approach that combines cognitive reasoning, sensory data processing, and decision-making strategies. It is mainly used to locate target objects in complex environments. It has four key features:

- **Goal-Oriented:** The search process always focuses on a clear goal, allowing the system to effectively narrow down the search space within a hierarchical structure. The robotic arm prioritizes searching areas highly relevant to the target, significantly improving efficiency.
- **Dynamically Updated:** The system can dynamically add search layers based on newly acquired information, similar to a dynamic tree search [23], which only updates its structure in response to new data. GODHS leverages a language model to semantically reassess and reorder priorities and paths. Each time new information is obtained, the system uses the language model to reassess and reorder priorities and paths.
- **Heuristic-Guided:** The search is guided by contextually weighted probabilities and adaptive semantic information generated by the language model, reducing ineffective exploration. The language model analyzes the environment to infer the most likely paths based on target priorities. This method relies on experientially informed probability and evolving experience rather than strict mathematical optimization.
- **Bounded Hierarchical:** The search space is organized into a hierarchical structure, gradually narrowing the scope from macro to micro levels, avoiding a one-time global search [24]. The hierarchy, from high to low, includes Scene, Room, Carrier, Feature, and Item.

The full process is shown in Algo. 1. The inputs are a defined scene name `s` and a target `t` to be searched, and the output is whether the target τ is found. Line 1 initializes the scene map $\mathcal{M}_{\mathcal{S}}$, room map $\mathcal{M}_{\mathcal{R}}$, carrier

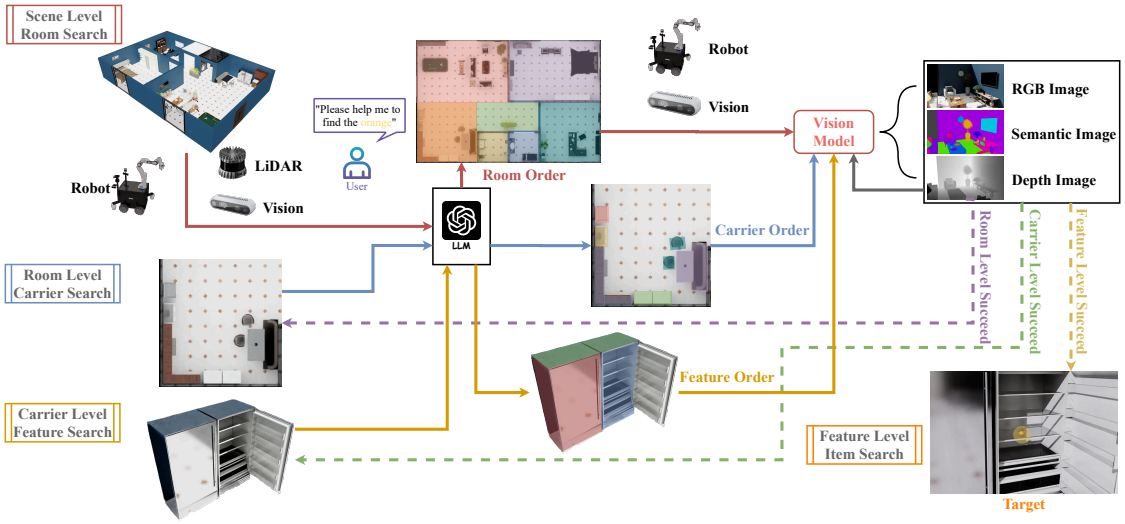

Fig. 2: **Complete Architecture of the GODHS System:** First, the user provides the target object through natural language input, which is then processed by a LLM to extract semantic intent. The robot captures geometric data using LiDAR and RGB images via cameras, generating a topological map with room names through the language model. Within the known map, the LLM-driven prioritization ranks rooms based on the likelihood of containing the target object and navigates to them sequentially. Within each room, the robot detects candidate objects through visual object detection, then utilizes the LLM to analyze whether they are carriers and ranks all collected carriers by the probability of containing the target. For each carrier, the LLM plans a hierarchical search strategy: first analyzing geometric features to identify subregions worth searching (top, interior, etc.), then prioritizing and searching these subregions according to probabilities assigned by the LLM. The loop terminates when the LLM verifies target recognition through visual-language grounding of sensor data.

map $\mathcal{M}_\mathcal{C}$, and room list $\mathbf{R}$, carrier list $\mathbf{C}$, feature list $\mathbf{F}$.

Algorithm 1: ObjectSearchGODHS(s, t)

Input : s — scene name, t — target name
Output: τ — found target

1 $\tau \leftarrow$ **False**, $\mathcal{M}_\mathcal{S}, \mathcal{M}_\mathcal{R}, \mathcal{M}_\mathcal{C} \leftarrow \varnothing$, $\mathbf{R}, \mathbf{C}, \mathbf{F} \leftarrow [\,]$
2 EnterScene(s)
3 **while** *not IsSceneMapComplete*($\mathcal{M}_\mathcal{S}$) **do**
4 EnterRandomRoom()
5 $\mathcal{M}_\mathcal{R} \leftarrow$ LidarToMap(LidarData())
6 $\mathcal{M}_\mathcal{S} \leftarrow$ UpdateSceneMap($\mathcal{M}_\mathcal{S}$, $\mathcal{M}_\mathcal{R}$)
7 $\mathcal{R}, \mathcal{I}_\mathcal{R} \leftarrow$ RoomMap($\mathcal{M}_\mathcal{R}$, InferRoom(SemSeg(CameraData())))
8 $\mathbf{R} \leftarrow$ SortRooms($\mathcal{R}$, t)
9 **foreach** r $\in \mathbf{R}$ **do**
10 MoveToRoom(r, $\mathcal{M}_\mathcal{R}$, $\mathcal{I}_\mathcal{R}$)
11 $\mathcal{C} \leftarrow$ ClassifyCarrier(SemSeg(CameraObservation()))
12 $\mathcal{M}_\mathcal{C}, \mathcal{I}_\mathcal{C} \leftarrow$ GetCarrierPCL($\mathcal{C}$, CarrierObservation())
13 $\mathcal{M}^*_\mathcal{C} \leftarrow$ GetCarrierGrid($\mathcal{M}_\mathcal{R}$)
14 $\mathbf{C} \leftarrow$ SortCarriers($\mathcal{C}$, t)
15 **foreach** c $\in \mathbf{C}$ **do**
16 $\mathcal{F} \leftarrow$ {'top', 'bottom', 'sides', 'inside'}
17 $\mathbf{F} \leftarrow$ ReasonFeatures(t, $\mathcal{F}$)
18 **foreach** f $\in \mathbf{F}$ **do**
19 $\mathcal{M}_\mathcal{F} \leftarrow$ PredictFeatureMap($\mathcal{M}_\mathcal{C}$, $\mathcal{M}^*_\mathcal{C}$, $\mathcal{I}_\mathcal{C}$, f)
20 $\mathcal{I}_\mathcal{F} \leftarrow$ PredictFeatureInfo($\mathcal{M}_\mathcal{F}$, f)
21 $\mathcal{P}_{\mathcal{E}\mathcal{E}} \leftarrow$ DetermineEEPoses($\mathcal{M}_\mathcal{F}$, $\mathcal{I}_\mathcal{F}$)
22 $\mathcal{P}_{\mathcal{C}\mathcal{H}} \leftarrow$ DetermineCHPoses($\mathcal{P}_{\mathcal{E}\mathcal{E}}$, $\mathcal{M}_\mathcal{F}$, $\mathcal{I}_\mathcal{F}$)
23 $\mathcal{P}^{\mathcal{E}\mathcal{E}}_{\mathcal{C}\mathcal{H}} \leftarrow$ CHToEEPoses($\mathcal{P}_{\mathcal{E}\mathcal{E}}$, $\mathcal{P}_{\mathcal{C}\mathcal{H}}$, $\mathcal{M}_\mathcal{F}$, $\mathcal{I}_\mathcal{F}$)
24 $\mathcal{P}^{\mathrm{EE}}_{\mathrm{CH}} \leftarrow$ PosesSorting($\mathcal{P}^{\mathcal{E}\mathcal{E}}_{\mathcal{C}\mathcal{H}}$)
25 **foreach** $\mathbf{P}_{CH} \in \mathcal{P}_{CH}$ **do**
26 NavigateToCHPose($\mathbf{P}_{\mathrm{CH}}$)
27 **foreach** $\mathbf{P}_{EE} \in \mathcal{P}_{EE}$ **do**
28 NavigateToEEPose($\mathbf{P}_{\mathrm{EE}}$)
29 **if** t $\in$ *SemSeg(CameraData())* **then**
30 **return True**
31 **return False**

Lines 2-8 are the process of Scene Level Room Search. The agent first enters scene s, activates its LiDAR and camera. The LiDAR constructs a 2D top-down map $\mathcal{M}_\mathcal{R}$ for each room, while the camera extracts object semantic information via semantic segmentation. Given multiple unknown rooms in the scene, we sequentially explore each room to acquire its corresponding map $\mathcal{M}_R$. The robot is manually navigated into unexplored rooms until no accessible rooms remain. Within each room, real-time LiDAR data is stitched into a grid map $\mathcal{M}_R$, and $\mathcal{M}_R$ is integrated into the global scene map $\mathcal{M}_S$ by appending only unexplored regions $\mathcal{M}_R \setminus \mathcal{M}_S$, formalized as: $\mathcal{M}_S \leftarrow \mathcal{M}_S \cup (\mathcal{M}_R \setminus \mathcal{M}_S)$.

The function InferRoom$(\cdot)$ in Line 7 employs a language model to predict the room category, which is stored in the room set via $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{r}^*\}$, where $\mathbf{r}^*$ is the new room name. The prediction is formulated as:

$$\mathbf{r}^* = \arg\max_{\mathsf{r} \in \mathcal{KB}} P\left(\mathsf{r} \mid \mathcal{O}, \mathbf{s}\right), \tag{1}$$

where $\mathcal{O}$ denotes the set of observed objects in the current room, $\mathbf{s}$ represents the semantic context (e.g., object spatial relationships), and $\mathcal{KB}$ is the knowledge base derived from the pre-trained language model. This formulation explicitly indicates that the reasoning process relies entirely on the language model's ability to generate hypotheses from its pre-trained knowledge, without external logical constraints or dynamic adaptation. Formally, for an input query q, the inference outcome y is generated as $y = \mathsf{LM}(q\,; \theta_{\text{pre-trained}})$, $\theta_{\text{pre-trained}}$ represents the model's fixed parameters acquired during pre-training, without additional fine-tuning or external knowledge integration. This generates a room-to-map correspondence $\mathcal{I}_\mathcal{R} : \mathcal{R} \rightarrow \mathcal{M}_\mathcal{R}$. Finally, the language model ranks elements of $\mathcal{R}$ by likelihood of target t in $\mathcal{R}$:

$$\mathbf{R} = \underset{\mathsf{r} \in \mathcal{R}}{\text{argsort}}\ P(\mathsf{r} \mid \mathcal{R}, \mathsf{t}), \tag{2}$$

here the prioritized room list determines the search sequence and denoted as $\mathbf{R}$. Leveraging the room-to-map correspondence $\mathcal{I}_R$ and grid maps $\mathcal{M}_R$, the agent navigates to each room following the order in $\mathbf{R}$. Within a room, initial stochastic observation captures images with the camera and extracts object semantics via semantic segmentation. Then ClassifyCarrier$(\cdot)$ in Line 11 of Algo. 1 is used to infer carrier objects by the language model, which are the appliances or furniture that can hold items and occupy grid areas in the top-down map. To classify the carriers, the probability of each carrier name $\mathtt{c}$ w.r.t. the observed objects $\mathcal{O}$ and target $\mathbf{t}$ must larger than the LLM-generated threshold τ_C:

$$\mathcal{C} = \{\mathtt{c} \in \mathcal{O} \mid P(\mathtt{c} \mid \mathcal{O}, \mathbf{t}) \geq \tau_C\}. \tag{3}$$

Focused scanning to capture carrier point clouds $\mathcal{M}_C$ is performed and carrier-to-map correspondence $\mathcal{I}_C : \mathcal{C} \rightarrow \mathcal{M}_C$ will be established. Carrier occupied grids should also be stored in $\mathcal{M}_C^*$ and all values will be normalized by resolution Δ, which means that all $(x, y, z) \in \mathcal{M}_C$ and $(x, y) \in \mathcal{M}_C^*$ have been normalized by hybrid ceiling-floor rounding $\left\lceil \frac{\cdot}{\Delta} \right\rfloor \cdot \Delta$. The language model ranks carriers by task relevance via likelihood of target $\mathtt{t}$ in $\mathcal{C}$, same as Eq. 2:

$$\mathbf{C} = \underset{\mathtt{c} \in \mathcal{C}}{\text{argsort}}\ P(\mathtt{c} \mid \mathcal{C}, \mathtt{t}). \tag{4}$$

Guided by the prioritized carrier list $\mathbf{C}$, the agent sequentially inspects each carrier $\mathtt{c} \in \mathbf{C}$. To mitigate inefficiencies from unstructured search and ensure operational safety, each carrier is decomposed into four characteristic spatial regions: $\mathcal{F} \coloneqq \{$'top', 'bottom', 'sides', 'inside'$\}$. The language model probabilistically identifies target-relevant regions through a synergistic mechanism combining Region Relevance Scoring and Feature Prioritization, which can be formally represented as a nested probabilistic decision framework: First, a binary screening of the full feature set $\mathcal{F}$ is performed using the conditional probability $P(\mathtt{f} \mid \mathcal{F}, \mathtt{t})$, generating the pruned feature subset $\mathcal{F}_t = \{\mathtt{f} \in \mathcal{F} \mid P(\mathtt{f} \mid \mathcal{F}, \mathtt{t}) \geq \tau_F\}$ under the target $\mathtt{t}$, where the threshold τ_F is adaptively determined by the language model based on task context. Subsequently, the retained features in $\mathcal{F}_t \subseteq \mathcal{F}$ undergo a total ordering via the re-evaluated conditional probability $P(\mathtt{f} \mid \mathcal{F}_t, \mathtt{t})$, yielding an explicitly prioritized sequence $\mathbf{F} = \underset{\mathtt{f} \in \mathcal{F}_t}{\text{argsort}}\, P(\mathtt{f} \mid \mathcal{F}_t, \mathtt{t})$. This dual-phase mechanism ensures both compactness of the search space through τ_F-driven pruning of low-relevance regions and operational guidance via the probabilistically ordered feature sequence $\mathbf{F}$, which explicitly encodes priority relationships for downstream task execution.

Lines 17-30 of Algorithm 1 iteratively process features in list $\mathbf{F}$ through feature-specific local map construction using environmental map $\mathcal{M}_\mathcal{R}$, vision data and geometric analysis, and semantic-guided pose computation. The system hierarchically organizes validated Chassis to End-Effector (CH-EE) pose pairs $\mathcal{P}_{\text{CH}}^{\text{EE}}$ using inverse kinematics (IK) validation with kinematic feasibility and collision constraints. During execution, the robot navigates prioritized chassis poses $\mathbf{P}_{\text{CH}}$, iterates through associated $\mathbf{P}_{\text{EE}}$, and terminates upon successful semantic segmentation verification. Details of Pose Dictionary Estimation are expanded in Sec. III-C.

B. Model Chain Design

Language models are powerful, but they exhibit statistical hallucination tendencies because they generate text through pattern matching rather than real "understanding" [25]. Particularly in syntactically constrained lexical processing, prompts convert inputs into textual representations while enforcing contextual grounding gaps lacking multi-alignment mechanisms. This process can lead to systematic error propagation through fabrication or semantic omission.

In designing our hierarchical reasoning framework, we drew inspiration from the "Chain-of-X" approach to guide the model in emulating human thought processes. We leverage the Chain-of-Thought [26] to systematically break down complex problems into simpler sub-problems, integrate temporally grounded contextual information [27] or cognitively structured background knowledge [28] through Chain-of-Augmentation, and refine the output through closed-loop self-feedback [29] with iterative verification protocols.

Chains can also be orchestrated through multi-stage verification for sequential tasks. Based on the above methods, smaller steps can better ensure the interpretability of the model's output. Thus, sequential tasks can achieve higher quality by simplifying atomic operations in each step and adding controlled steps. However, blindly increasing the number of steps does not guarantee maximum reliability. Here, we designed three main steps and two optional steps, which constitute the Chain-of-Refinement. The three main steps are: the Cleaning function $C(\cdot)$ to ensure input $\mathbf{I}$ structural validity, the Processing function $P(\cdot)$ to perform core task execution, and the Correcting function $R(\cdot)$ to ensure output $\mathbf{O}$ semantic consistency and reliability. The two optional steps are the Domain-Specific Knowledge Base $\mathbf{K}$ and Iterative Self-Feedback $F(\cdot)$, which are activated context-dependently as tasks progress. All steps are shown in the left of Fig. 3, and this process can be represented as:

$$\mathbf{O} = R\left(P\left(C(\mathbf{I}), \mathbf{K}\right), F\left(P\left(C(\mathbf{I}), \mathbf{K}\right)\right)\right). \tag{5}$$

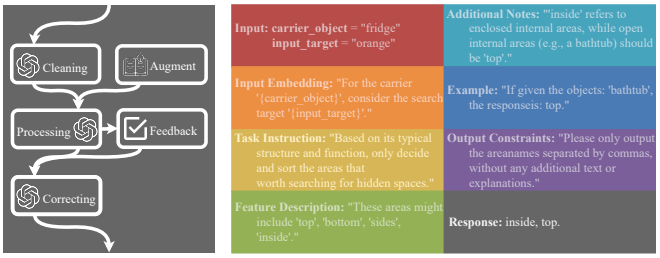

Fig. 3: **Chain-of-Refinement (Left):** Data Cleaning and Correcting ensure the input and output are structurally valid and semantically coherent, while Processing executes the core task with Knowledge as Augmentation and Feedback to guarantee the result.
Prompt Design (Right): Input embedding receives the input information, output constraints standardize lexico-syntactic results, task instructions describe goal-oriented content, feature description provides constrained descriptions of features, and there are also additional information and examples.

Embedded in the chain are prompts, which ensure high-quality inputs for the language model. An example is shown on the right of Fig. 3. This prompt aims to analyze searchable features for a carrier. First, it embeds the current carrier 'fridge' and target object 'orange' into the prompt. The task is then defined as knowledge-based identification of features to search, with possible features limited to 'top', 'bottom', 'sides', and 'inside'. If ambiguity arises, prompts are provided (e.g., "'inside' must be an enclosed space"), and examples are given (e.g., "bathtub should return 'top'").

To ensure operable outputs, results are forced into a comma-separated format with no additional explanations allowed. When processed by the language model, this example identifies 'inside' and 'top' as search-worthy features.

In Data Cleaning and Data Correcting, the goal of prompt design is to ensure inputs and outputs match the required hierarchical types. Data Correcting adds a double-check to guarantee output constraints are met.

C. Pose Dictionary Estimation

This subsection details the methodology for computing chassis (CH) and end-effector (EE) poses to enable feature-guided manipulation in unstructured environments. The goal is to generate search trajectories for the chassis and the end-effector separately, and to have the chassis move forward around the carrier while ensuring that the distance traveled by the end of the actuator at each chassis position is minimized. The process involves five sequential stages: (1) extraction of geometric features $\mathcal{M}_F$ from point cloud data, (2) generation and selection of EE poses $\mathcal{P}_{\mathcal{EE}}$, (3) fast evaluation and selection of chassis poses $\mathcal{P}_{\mathcal{CH}}$ via geometric solution, (4) validation of the relation between CH and EE $\mathcal{P}_{\mathcal{CH}}^{\mathcal{EE}}$ with subsequent inverse kinematics (IK) solving, and (5) sorting of the resulting pose mappings $\mathcal{P}_{\text{CH}}^{\text{EE}}$ for sequential execution.

The process begins with the acquisition of the point cloud of single carrier $\mathcal{M}_C \subset \mathbb{R}^3$ and the occupied grid of the carrier $\mathcal{M}_C^* \subset \mathbb{R}^2$. These data sources are used to extract the carrier's relevant features, collectively denoted as $\mathcal{M}_F$. The features are subdivided into four components: Top Surface, Side Surface, Bottom Area and Inside Area.

The top surface feature $\mathcal{M}_F^{\text{top}}$ is extracted directly from point cloud $\mathcal{M}_C$. For each (x_F, y_F) coordinate in the grid, the top surface point is identified as the point with the maximum z_F value. Formally, if we denote by $\mathcal{M}_C^*$ the grid cell corresponding to a particular (x_F, y_F) location, then

$$\mathcal{M}_F^{\text{top}} = \left\{ (x_F, y_F, z_F) \,\middle|\, \begin{array}{l} (x_F, y_F) \in \mathcal{M}_C^*, \\ (x_F, y_F, z_F') \in \mathcal{M}_C, \\ z_F = \\ \max\{z_F' \mid (x_F, y_F, z_F')\} \end{array} \right\}. \quad (6)$$

The side surface feature $\mathcal{M}_F^{\text{sides}}$ is derived from the boundary of the volumetric occupied grid $\mathcal{M}_C^*$. For each (x_F, y_F) coordinate lying on the topologically defined edge of $\mathcal{M}_C^*$ (denoted by $\partial\mathcal{M}_C^*$), the corresponding side surface value is determined by extracting the maximum z_F value in $\mathcal{M}_C$ over the range starting from $z_F = 0$. This could be expressed as:

$$\mathcal{M}_F^{\text{sides}} = \left\{ (x_F, y_F, z_F) \,\middle|\, \begin{array}{l} (x_F, y_F) \in \partial\mathcal{M}_C^*, \\ (x_F, y_F, z_F') \in \mathcal{M}_C, \\ z_F \in \\ [0, \max\{z_F' \mid (x_F, y_F, z_F')\}] \end{array} \right\}. \quad (7)$$

This representation captures the vertical boundaries of the carrier. The bottom area feature $\mathcal{M}_F^{\text{bottom}}$ is defined by the intersection of the occupied grid $\mathcal{M}_C^*$ with a spatially constrained horizontal plane at a fixed height $z_F = z_{F0}$:

$$\mathcal{M}_F^{\text{bottom}} = \left\{ (x_F, y_F, z_{F0}) \,\middle|\, (x_F, y_F) \in \mathcal{M}_C^* \right\}. \quad (8)$$

The inside area is obtained via a specialized procedure aimed at accurately localizing the door handle and can be provided manually, which we won't explain in depth here.

For the mobile manipulator, two types of poses are essential: the chassis pose and the EE pose. The candidate chassis pose is defined as $\mathbf{p}_{CH}^{Can} = (x_{CH}, y_{CH}, \theta_{CH})$, where (x_{CH}, y_{CH}) represents the base position and θ_{CH} the yaw angle. The candidate EE pose is defined as $\mathbf{p}_{EE}^{Can} = (x_{EE}, y_{EE}, z_{EE}, \phi_{EE}, \theta_{EE}, \psi_{EE})$. The direction vector of the camera's view can be expressed as $\mathbf{v}_{\text{dir}} = (\cos(\theta_{EE}) \cdot \cos(\psi_{EE}), \cos(\theta_{EE}) \cdot \sin(\psi_{EE}), \sin(\theta_{EE}))$. The vector pointing from the camera to the surface point is $\mathbf{v}_{\text{point}} = (x_F - x_{EE}, y_F - y_{EE}, z_F - z_{EE})$. The surface point is considered covered by the vision cone of the camera, if the conditions of horizontal and vertical angle $\theta_{\text{horizontal}} = \arccos\left(\frac{v_{\text{dir},x} \cdot v_{\text{point},x} + v_{\text{dir},z} \cdot v_{\text{point},z}}{\|v_{\text{dir}}\| \|v_{\text{point}}\|}\right) < \text{FoV}_{\text{horizontal}}/2$ and $\theta_{\text{vertical}} = \arccos\left(\frac{v_{\text{dir},y} \cdot v_{\text{point},y} + v_{\text{dir},z} \cdot v_{\text{point},z}}{\|v_{\text{dir}}\| \|v_{\text{point}}\|}\right) < \text{FoV}_{\text{vertical}}/2$ are satisfied, and the candidate pose earns one point. We use greedy algorithm [30] to select the set of the EE poses $\mathcal{P}_{\mathcal{EE}}$.

The set of chassis poses $\mathcal{P}_{\mathcal{CH}}$ is selected from $\mathbf{p}_{CH}^{Can}$ via deterministic greedy algorithm based on $\mathcal{P}_{\mathcal{EE}}$. If a geometric solution can establish a collision-free connection, one point is awarded. The selection continues until $\mathcal{P}_{\mathcal{CH}}$ is found that spatially covers all elements in $\mathcal{P}_{\mathcal{EE}}$.

Since fast-solving cannot guarantee a feasible solution for the actual robot operation, and directly solving the IK for all poses would lead to excessive computational overhead, we perform IK solving with Levenberg–Marquardt algorithm [31] for each chassis pose $\mathbf{P}_{\text{CH}}$ in $\mathcal{P}_{\mathcal{CH}}$ and each end-effector pose $\mathbf{P}_{\text{EE}}$ in $\mathcal{P}_{\mathcal{EE}}$. If numerically stable solutions exist, the mapping is added to the dictionary $\mathcal{P}_{\mathcal{CH}}^{\mathcal{EE}}$.

The dictionary $\mathcal{P}_{\mathcal{CH}}^{\mathcal{EE}}$ is non-sequential, so a sorting method is required to obtain an ordered version $\mathcal{P}_{\text{CH}}^{\text{EE}}$. For each end-effector pose corresponding to a chassis pose, we apply Lexicographical Sorting based on the sequence $z_{EE}, y_{EE}, x_{EE}, \psi_{EE}, \theta_{EE}, \phi_{EE}$.

For all chassis poses, we do not simply sort based on spatial distance but instead ensure movement proceeds clockwise around the carrier. To achieve this, we use centroid-aligned Polar Angle Sorting [32], and the geometric centroid $(\bar{x}, \bar{y}) = \left(\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i\right)$ is computed where $(x_i, y_i) \in \mathcal{M}_C$, with n being the size of the point set. Then we can calculate its angle relative to the average point by:

$$\rho = \arctan_2\left(y_{\text{CH}} - \bar{y},\, x_{\text{CH}} - \bar{x}\right), \quad (9)$$

where $(x_{\text{CH}}, y_{\text{CH}})$ is the position of each chassis. Following the angle ρ, the chassis poses will be sorted. Using this method, the movement platform can advance around the carrier while steadily maintaining its direction of movement, avoiding the situation where sorting based on the nearest spatial distance might result in the actual path distance being significantly greater than the spatial distance.

IV. Experimental Setup and Results

A. Simulation Setup

The simulation framework integrates the DARKO mobile manipulator, comprising an omnidirectional RB-KAIROS base and a 7-DOF Franka Emika Panda arm, equipped with Ouster OS1 LiDAR and Intel RealSense D435 RGB-D cameras. The NVIDIA Isaac Sim 4.0 simulator serves as the primary simulation environment, leveraging PhysX for physics simulation and RTX rendering for photorealistic sensor data generation. ROS Noetic facilitates system integration through ROS Bridge, with critical components including: (1) Navigation Stack (move_base) for mobile base control using Mecanum wheel odometry and 3D LiDAR

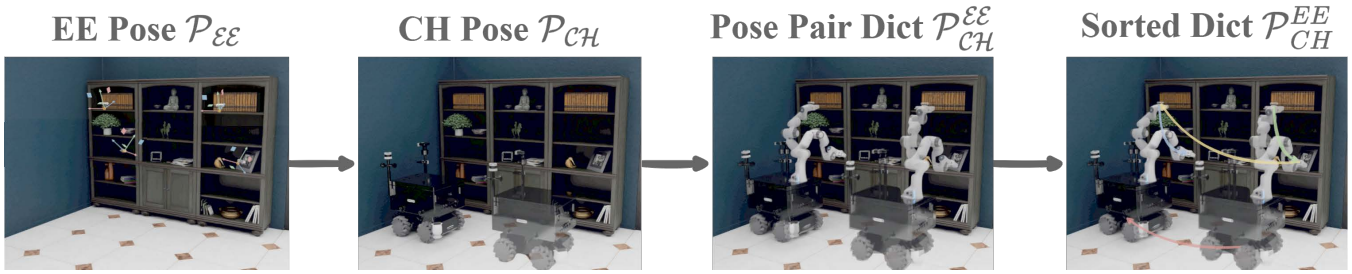

Fig. 4: **Leftmost:** Determine EE poses via carrier geometry analysis. **Second Left:** Generate CH poses through greedy EE pose exploration. **Second Right:** Verify CH-EE pairs with inverse kinematics validation. **Rightmost:** Prioritize CH poses by Polar Angle Sorting and EE poses by Lexicographical Sorting.

mapping; (2) MoveIt framework for constrained manipulator motion planning utilizing OMPL algorithms with Cartesian path constraints for straight-line end-effector trajectories; (3) Custom Python nodes for sensor fusion, processing 128-channel LiDAR point clouds (20Hz) and RGB-D data (640×480@30Hz). The cognitive layer employs Ollama, which is integrated for local LLM interactions. Physical parameters replicate real-world dynamics by aligning gravity with actual conditions, using Persistent Contact Manifold for collision detection, the Projected Gauss-Seidel solver and Multi-Box Pruning for force and collision computations, and a patch friction model to accurately constrain static and dynamic friction, while Continuous Collision Detection and discrete GPU dynamics pipelines are disabled to prevent excessive collisions of Mecanum wheels.

B. Feasibility Testing

The feasibility testing aims to validate the target search capability and semantic planning effectiveness of the proposed mobile manipulator system in a constrained indoor environment. The experiments were conducted in a simulated indoor environment ("flat" scene) consisting of seven functional zones (kitchen, living room, bedroom, study room, bathroom, restroom, and corridor), as illustrated in the bottom left of Fig. 5. The target object for single time example testing was an "orange" placed and occluded inside the fridge in the kitchen, with the system required to navigate through multiple rooms and identify the correct location.

The robot utilizes a high-fidelity LiDAR to scan the environment and generate a complete occupancy map, as shown in the lower-left corner of Fig. 5. Semantic segmentation is employed to collect the semantics of objects in the scene, which are then used by a language model to identify the room name. For example, if the collected semantics include 'bed', 'dressing table', and 'clothing cabinet', the room can be determined as a 'bedroom' by LLM.

Based on the target 'orange', the language model infers the priority of room searches according to the context, starting with the 'living room', followed by the 'kitchen'. In the 'living room', the robot first randomly observes the scene to receive the semantic items in the room. Through language model analysis, the 'coffee table', a potential carrier worth searching is identified. Further analysis suggests that both its 'top' and 'bottom' should be inspected. However, as the target is not found, the robot proceeds to the 'kitchen'.

Similarly, through language model analysis, the 'fridge', a potential carrier worth searching is identified. Further analysis suggests that the 'side surface' and 'inside area' should be inspected. Following the model's guidance, the robot examines the side surface of the 'fridge' and subsequently opens its door. Eventually, the robot successfully finds the 'orange' inside the 'fridge', completing the search task.

To quantitatively demonstrate the feasibility of our tested search approach, we conducted 81 experiments using Qwen2.5-7B and GPT-4o on a 'flat' scene consisting of 7 rooms, 38 carriers, and 137 items, where objects were placed in reasonable locations. The evaluation metrics include success rate S, room search rate R_r, carrier search rate R_c, and item search rate R_i. Overall search rate OSR can be defined as the weighted sum of the room, carrier, and item search rates. For instance, a simple definition could be:

$$OSR = w_1 \cdot R_r + w_2 \cdot R_c + w_3 \cdot R_i, \tag{10}$$

where w_1, w_2, and w_3 are weights that reflect the relative importance of each search level. A reasonable choice for the weights could be assigned as $w_1 = 0.2$, $w_2 = 0.3$, $w_3 = 0.5$.

To compare and highlight the advantages of our method, we also tested two traditional non-semantic-guided search methods: Coverage Search [33] and Random Walk Search [34]. The search results are shown in the Tab. I.

TABLE I: Search performance of different stategies.

Method	$S(\%)$	$R_r(\%)$	$R_c(\%)$	$R_i(\%)$	$OSR(\%)$
GPT-4o	74.32	**21.43**	20.53	21.17	**21.03**
Qwen2.5	68.95	33.85	**19.91**	**19.74**	22.61
Coverage	92.84	58.57	61.71	60.56	60.51
Random	**93.72**	47.14	52.10	53.38	51.75

The experimental results and data demonstrate that our method can effectively perform the search tasks while significantly reducing search costs, thereby validating the feasibility of GODHS. However, the experimental results demonstrate that the search success rate progressively decreases following semantic analysis. Traditional methods fail in narrow spaces due to grid-based maps being unable to model 3D semantic relationships, while GODHS faces additional limitations from generative language models, particularly the symbol grounding problem. For instance, an 'table' instruction might correspond to multiple carriers' physical coordinates, thereby introducing inherent uncertainty in the generated searches.

Experimental failures arise from three main issues. First, physical constraints: the DARKO robot's lower-section search is limited to a minimum reachable height of 0.45 m, while the 'coffee table' in Fig. 5 only reaches 0.3 m, meaning the 'bottom' is a fixed-height constraint rather than true lower space. Second, a lack of physical commonsense:

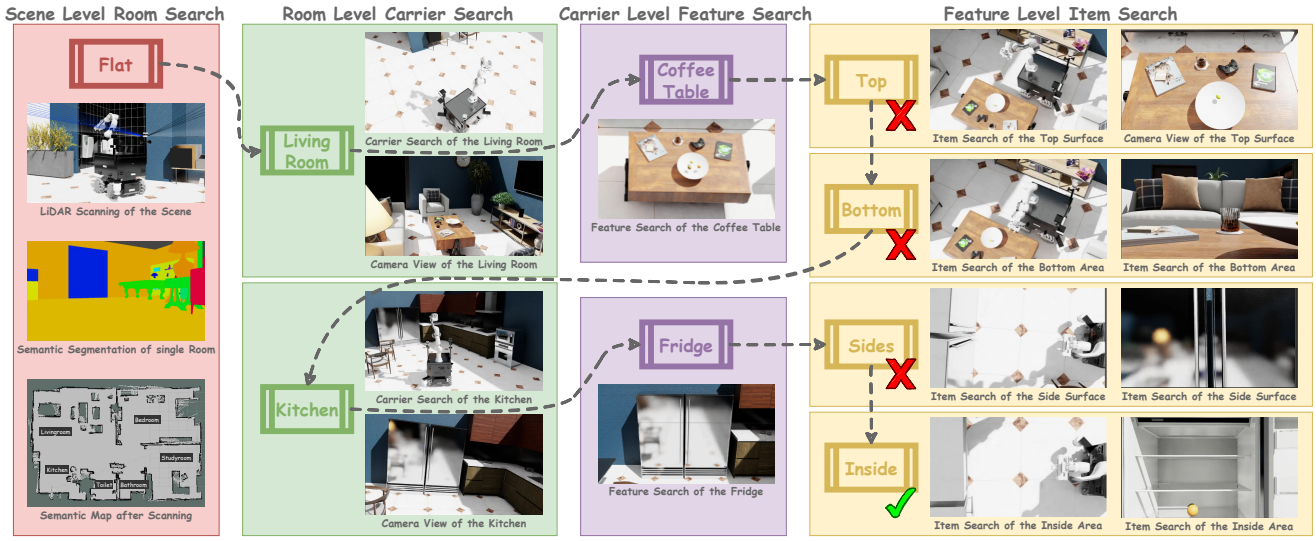

Fig. 5: **Feasibility Example:** Taking the Qwen2.5-7B-powered search as an example, the search target is an orange in a flat scene. After entering the scene, the robot first scans the map with lasers to obtain a complete map composed of each room (lower left corner). Then, driven by the language model, it enters the living room and kitchen sequentially. In the living room, the language model determines that only the upper surface and underside of the coffee table are worth searching, but finds nothing. In the kitchen, it first checks the side of the fridge without success, then opens the door and finds the orange inside.

insufficient training data causes misinterpretation of carrier features (e.g., including a fridge's side sections). Third, semantic ambiguity: language models struggle with contextual categorization, as seen with the 'billiard table', whose floor presence conflicts with its non-furniture classification, reducing detection accuracy.

C. Performance Evaluation

In this section, we evaluate the performance of our proposed Chain-of-Refinement in Sec. III-B for enhanced target search and the Path Planning Sorting method in Sec. III-C for efficient path optimization. Our experiments are designed to assess both the effectiveness and efficiency of these approaches in the simulated environment from Sec. IV-B.

As shown in the left part of Fig. 3 in the Chain-of-Refinement, the core operation is *Processing*, while the *Cleaning* and *Correcting* steps are used to ensure the correctness of the results, and *Augment* and *Feedback* are employed to further improve accuracy. Therefore, in this experiment, we compare the results obtained from only the Processing step, those with the protective steps added, and those with the precision steps added. This comparison is based on the five tasks throughout the entire process, namely, Room Reasoning, Room Sorting, Carrier Classification, Carrier Sorting, and Feature Reasoning. Since our outputs are phrases, we compare the proportion of correctly or reasonably generated phrases relative to the total generated. The experimental results are presented in Tab. II.

The table reveals several distinct characteristics in the results. First, since Room Reasoning outputs only a single room result, there is little difference among the three conditions. In contrast, Room Sorting, Carrier Classification, and Carrier Sorting generate a large number of outputs, and they exhibit significant improvement after the application of the Protective step, whereas Feature Reasoning produces fewer outputs, resulting in less improvement. Furthermore, due to the complex variety and insufficient pre-trained knowledge, the outputs of Carrier Classification, Carrier Sorting, and Feature Reasoning are markedly enhanced after augmentation, while Room Sorting shows the opposite behavior.

TABLE II: Results for Different Chain Configurations.

Task	Processing Only	With Protective	With Precision
Room Reasoning	88.53%	90.18%	**91.43%**
Room Sorting	80.98%	92.91%	**93.19%**
Carrier Classification	65.49 %	71.58%	**80.26%**
Carrier Sorting	54.01%	77.01%	**86.69%**
Feature Reasoning	78.27%	81.05%	**88.47%**

To systematically assess the effectiveness of our proposed sorting strategies, we compare four configurations: an unoptimized baseline with no sorting applied to EE or CH poses, a configuration where EE poses are optimized via lexicographical sorting, another where CH poses are optimized using polar angle sorting, and a final setup where both optimizations are applied together.

We evaluate three key performance metrics: the Normalized EE Path Length, which represents the ratio of the total EE travel distance to the theoretical shortest EE path; the Normalized CH Path Length, defined similarly for the CH poses; and the Execution Time Ratio, which is the ratio of execution time after optimization to that of the unoptimized case, with a lower value indicating better efficiency.

TABLE III: Comparison of Sorting Methods for EE and CH Poses.

Sorting Method	EE Path	CH Path	Time Ratio
Unoptimized	2.81	2.37	1.00
EE Sorting	**1.75**	2.41	0.87
CH Sorting	2.79	**1.60**	0.83
Both Optimized	**1.77**	**1.59**	**0.66**

Tab. III presents the comparative results, demonstrating

that lexicographical sorting effectively reduces EE path length while polar angle sorting significantly improves CH path efficiency. When both strategies are applied, the method achieves the highest optimization in both path length and execution time, confirming its effectiveness in improving motion planning efficiency.

V. Conclusion

In this work, we propose the **GODHS Framework**, a search strategy that combines a Language Model with scene scanning and navigation, improving the efficiency of the search process. We also build the **Chain-of-Refinement** to ensure high-quality outputs. **Polar Angle Sorting** and **Lexicographical Sorting** help the chassis and robotic arm navigate and manipulate as efficiently as possible in space. Experiments show that our method is not only feasible but also significantly outperforms traditional search methods. In the future, leveraging multi-modal foundation models [35] and fine-tuned language models [36] will enable us to solve the problems of cascading errors caused by manually spliced rules and insufficiently focused knowledge.

References

[1] B. Kuipers, "The spatial semantic hierarchy," *Artificial intelligence*, vol. 119, no. 1-2, pp. 191–233, 2000. I

[2] L. E. Wixson and D. H. Ballard, "Using intermediate objects to improve the efficiency of visual search," *Int. J. Comput. Vision*, vol. 12, no. 2–3, p. 209–230, Apr. 1994. [Online]. Available: https://doi.org/10.1007/BF01421203 II

[3] H. Huang, M. Danielczuk, C. M. Kim, L. Fu, Z. Tam, J. Ichnowski, A. Angelova, B. Ichter, and K. Goldberg, "Mechanical search on shelves using a novel "bluction" tool," pp. 6158–6164, 2022. II

[4] Y. Ye and J. K. Tsotsos, "Sensor planning for 3d object search," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 145–168, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314298907366 II

[5] E. Gelenbe and Y. Cao, "Autonomous search for mines," *European Journal of Operational Research*, vol. 108, no. 2, pp. 319–333, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221797003731 II

[6] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," *CoRR*, vol. abs/1609.05143, 2016. [Online]. Available: http://arxiv.org/abs/1609.05143 II

[7] J. K. Li, D. Hsu, and W. S. Lee, "Act to see and see to act: Pomdp planning for objects search in clutter," pp. 5701–5707, 2016. II

[8] S. Patki, E. Fahnestock, T. M. Howard, and M. R. Walter, "Language-guided semantic mapping and mobile manipulation in partially observable environments," 2019. [Online]. Available: https://arxiv.org/abs/1910.10034 II

[9] D. Joho, M. Senk, and W. Burgard, "Learning search heuristics for finding objects in structured environments," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 319–328, 2011, special Issue ECMR 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889011000327 II

[10] Y. Li, Y. Ma, X. Huo, and X. Wu, "Remote object navigation for service robots using hierarchical knowledge graph in human-centered environments," *Intelligent Service Robotics*, vol. 15, pp. 1–15, 06 2022. II

[11] M. R. Dogar, M. C. Koval, A. Tallavajhula, and S. S. Srinivasa, "Object search by manipulation," pp. 4973–4980, 2013. II

[12] B. Moldovan and L. De Raedt, "Occluded object search by relational affordances," pp. 169–174, 2014. II

[13] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," 2023. [Online]. Available: https://arxiv.org/abs/2210.05714 II

[14] M. Chang, T. Gervet, M. Khanna, S. Yenamandra, D. Shah, S. Y. Min, K. Shah, C. Paxton, S. Gupta, D. Batra, R. Mottaghi, J. Malik, and D. S. Chaplot, "Goat: Go to any thing," 2023. [Online]. Available: https://arxiv.org/abs/2311.06430 II

[15] Y. Tang, M. Wang, Y. Deng, Z. Zheng, J. Deng, and Y. Yue, "Openin: Open-vocabulary instance-oriented navigation in dynamic domestic environments," 2025. [Online]. Available: https://arxiv.org/abs/2501.04279 II

[16] A. Kurenkov, J. Taglic, R. Kulkarni, M. Dominguez-Kuhne, A. Garg, R. Martín-Martín, and S. Savarese, "Visuomotor mechanical search: Learning to retrieve target objects in clutter," *CoRR*, vol. abs/2008.06073, 2020. [Online]. Available: https://arxiv.org/abs/2008.06073 II

[17] 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," 2023. [Online]. Available: https://arxiv.org/abs/2212.04088 II

[18] L. L. Wong, L. P. Kaelbling, and T. Pérez, "Manipulation-based active search for occluded objects," IEEE, pp. 2814–2819, 2013. II

[19] 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*, vol. 9, no. 10, p. 8298–8305, Oct. 2024. [Online]. Available: http://dx.doi.org/10.1109/LRA.2024.3441495 II

[20] Y. Wang, F. Giuliari, R. Berra, A. Castellini, A. D. Bue, A. Farinelli, M. Cristani, and F. Setti, "POMP: pomcp-based online motion planning for active visual search in indoor environments," *CoRR*, vol. abs/2009.08140, 2020. [Online]. Available: https://arxiv.org/abs/2009.08140 II

[21] D. Qiu, W. Ma, Z. Pan, H. Xiong, and J. Liang, "Open-vocabulary mobile manipulation in unseen dynamic environments with 3d semantic maps," 2024. [Online]. Available: https://arxiv.org/abs/2406.18115 II

[22] M. R. Walter, S. Patki, A. F. Daniele, E. Fahnestock, F. Duvallet, S. Hemachandra, J. Oh, A. Stentz, N. Roy, and T. M. Howard, "Language understanding for field and service robots in a priori unknown environments," 2021. [Online]. Available: https://arxiv.org/abs/2105.10396 II

[23] D. D. Sleator and R. Endre Tarjan, "A data structure for dynamic trees," *Journal of Computer and System Sciences*, vol. 26, no. 3, pp. 362–391, 1983. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022000083900065 III-A

[24] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt, "Active visual object search in unknown environments using uncertain semantics," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 986–1002, 2013. III-A

[25] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022. [Online]. Available: https://arxiv.org/abs/2203.02155 III-B

[26] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023. [Online]. Available: https://arxiv.org/abs/2201.11903 III-B

[27] R. Luo, T. Gu, H. Li, J. Li, Z. Lin, J. Li, and Y. Yang, "Chain of history: Learning and forecasting with llms for temporal knowledge graph completion," 2024. [Online]. Available: https://arxiv.org/abs/2401.06072 III-B

[28] W. Yu, H. Zhang, X. Pan, K. Ma, H. Wang, and D. Yu, "Chain-of-note: Enhancing robustness in retrieval-augmented language models," 2024. [Online]. Available: https://arxiv.org/abs/2311.09210 III-B

[29] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, "Self-refine: Iterative refinement with self-feedback," 2023. [Online]. Available: https://arxiv.org/abs/2303.17651 III-B

[30] E. W. Dijkstra, "A note on two problems in connexion with graphs," pp. 287–290, 2022. III-C

[31] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," 1986. III-C

[32] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Inf. Process. Lett.*, vol. 1, pp. 132–133, 1972. [Online]. Available: https://api.semanticscholar.org/CorpusID:45778703 III-C

[33] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 77–98, 02 2001. IV-B

[34] 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'*, 1997, pp. 146–151. IV-B

[35] R. Bommasani, D. A. Hudson, E. Adeli *et al.*, "On the opportunities and risks of foundation models," 2022. [Online]. Available: https://arxiv.org/abs/2108.07258 V

[36] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," 2022. [Online]. Available: https://arxiv.org/abs/2109.01652 V