Language-Grounded Dynamic Scene Graphs for Interactive Object Search with Mobile Manipulation

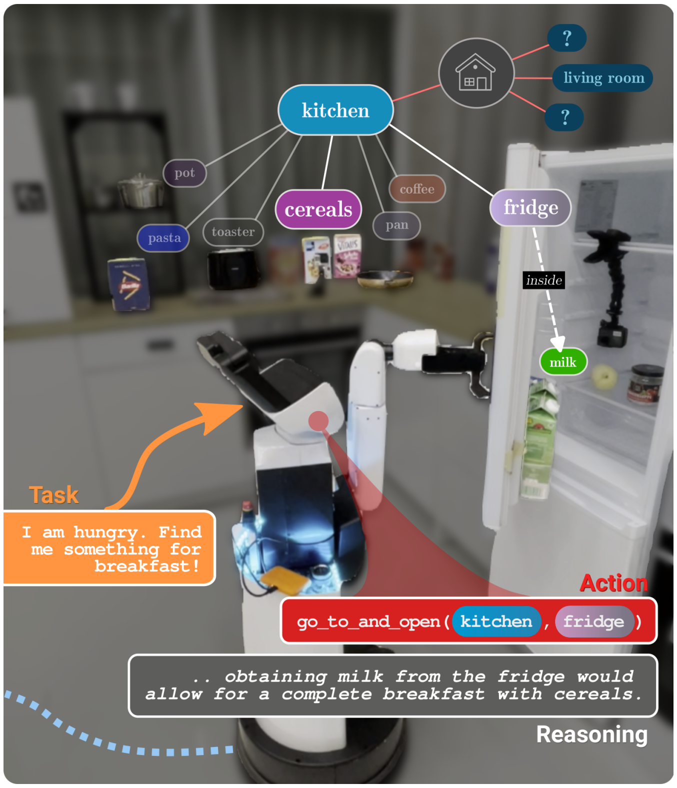
Daniel Honerkamp[1*], Martin Büchner[1*], Fabien Despinoy[2], Tim Welschehold[1], Abhinav Valada[1]

***Abstract*—To fully leverage the capabilities of mobile manipulation robots, it is imperative that they are able to autonomously execute long-horizon tasks in large unexplored environments. While large language models (LLMs) have shown emergent reasoning skills on arbitrary tasks, existing work primarily concentrates on explored environments, typically focusing on either navigation or manipulation tasks in isolation. In this work, we propose MoMa-LLM, a novel approach that grounds language models within structured representations derived from open-vocabulary scene graphs, dynamically updated as the environment is explored. We tightly interleave these representations with an object-centric action space. Given object detections, the resulting approach is zero-shot, open-vocabulary, and readily extendable to a spectrum of mobile manipulation and household robotic tasks. We demonstrate the effectiveness of MoMa-LLM in a novel semantic interactive search task in large realistic indoor environments. In extensive experiments in both simulation and the real world, we show substantially improved search efficiency compared to conventional baselines and state-of-the-art approaches, as well as its applicability to more abstract tasks. We make the code publicly available at http://moma-llm.cs.uni-freiburg.de.**

I. INTRODUCTION

Interactive embodied AI tasks in large, unexplored, human-centered environments require reasoning over long horizons and a multitude of objects. In many cases, the considered environments are a priori unknown or continuously rearranged. Recent advancements have demonstrated the potential of large language models (LLMs) in generating high-level plans [19, 30, 26, 28]. However, these efforts have predominantly focused on fully observed environments such as tabletop manipulation, or a priori explored scenes, struggling to generate executable and grounded plans suitable for real-world robotic execution. This problem is strongly exacerbated in large scenes with numerous objects and long time horizons. In turn, this increases the risk of generating impractical sequences or hallucinations [38, 1]. Furthermore, the presence of interactive scenes and articulated objects introduces a multitude of potential states and failure cases.

To address these challenges, we propose grounding LLMs in dynamically built scene graphs. Our approach incorporates a scene understanding module that, given object detections, constructs open-vocabulary scene graphs from dense maps and Voronoi graphs. These diverse representations are then tightly interweaved with an object-centric action space. Leveraging the current scene representation, we extract structured and compact textual representations of the scene to facilitate efficient planning with pre-trained LLMs.



I am hungry. Find me something for breakfast!

`go_to_and_open( kitchen , fridge )`

.. obtaining milk from the fridge would allow for a complete breakfast with cereals.

Fig. 1: MoMa-LLM performs long-horizon interactive object search in household environments from language queries using dynamically built scene graphs.

To evaluate the efficacy of our approach, we formulate an interactive semantic search task, extending previous non-semantic interactive tasks [41] to more complex scenarios. In this task, an agent has to find a target object within an indoor environment, encapsulating real-world challenges such as opening doors to navigate through the environment, and searching inside cabinets and drawers to find the desired object. This task is challenging as it requires reasoning about manipulation and navigation skills, operating in unexplored environments, spanning large apartments with numerous rooms and objects. Consequently, it is representative of more complex mobile manipulation tasks while retaining the specificity required for thorough evaluations and comparisons against conventional methods. Furthermore, we introduce a novel evaluation paradigm for object search tasks, employing

---

[*] Equal contribution.

[1] Department of Computer Science, University of Freiburg, Germany.

[2] Toyota Motor Europe (TME).

Supplementary material is provided at https://arxiv.org/pdf/2403.08605.pdf.

This work was funded by Toyota Motor Europe (TME) and an academic grant from NVIDIA.

full efficiency curves to remove the dependency on arbitrary time budgets inherent in existing methods. Additionally, we propose the *AUC-E* metric to distill these curves into a single metric for coherent evaluation. We perform extensive experimental evaluations in both simulation and the real-world, and demonstrate that given appropriately structured representations, LLMs can leverage their accumulated knowledge about the human world to achieve exceptional results, outperforming state-of-the-art approaches across diverse fields. Our approach is zero-shot, with open-vocabulary reasoning, and inherently scalable to various mobile manipulation and household robotic tasks, as we demonstrate on a set of abstract search tasks.

To summarize, our main contributions are

- A scalable scene representation centered around a dynamic scene graph with open-vocabulary room clustering and classification.
- Structured compact knowledge extraction to ground LLMs in scene graphs for large unexplored environments.
- Semantic interactive search task for large scenes with numerous objects and receptacles.
- Novel evaluation paradigm for object search tasks through full efficiency curves, instead of a single time budget.
- We release the code at http://moma-llm.cs.uni-freiburg.de.

II. RELATED WORK

3D Scene Graphs serve as sparse environment representations that abstract from dense semantic [11] or panoptic maps [45]. The disassembly of large scenes into objects, regions, etc., and their representation as nodes thus provides hierarchical and object-centric representations. In addition, nodes and edges may contain semantic attributes [20, 4, 12, 48]. Despite the lower geometric fidelity compared to dense maps, scene graphs prove particularly successful in the realm of high-level reasoning and planning, while providing a powerful interface with mapping [12, 38]. Orthogonally, Hydra [20] focuses on representing dynamically changing scenes. Multiple works have explored the use of scene graphs for reasoning in mobile robotics. Most build a hierarchical scene graph of the form (building, floors, rooms, objects). Different variants add edges among objects [14], add a Voronoi graph [47] for storing observations, or separate the set of objects into static and potentially moving objects [36]. While ConceptGraphs [14] and VoroNav [47] investigate the use of zero-shot perception inputs for task planning, others such as SayPlan [38], SayNav [36], and Taskography [1] focus on the reasoning task itself by utilizing ground truth semantic scene graphs [34, 5]. Realizing object navigation using both dynamic and interactive scene graphs has not been tackled thus far in the aforementioned works.

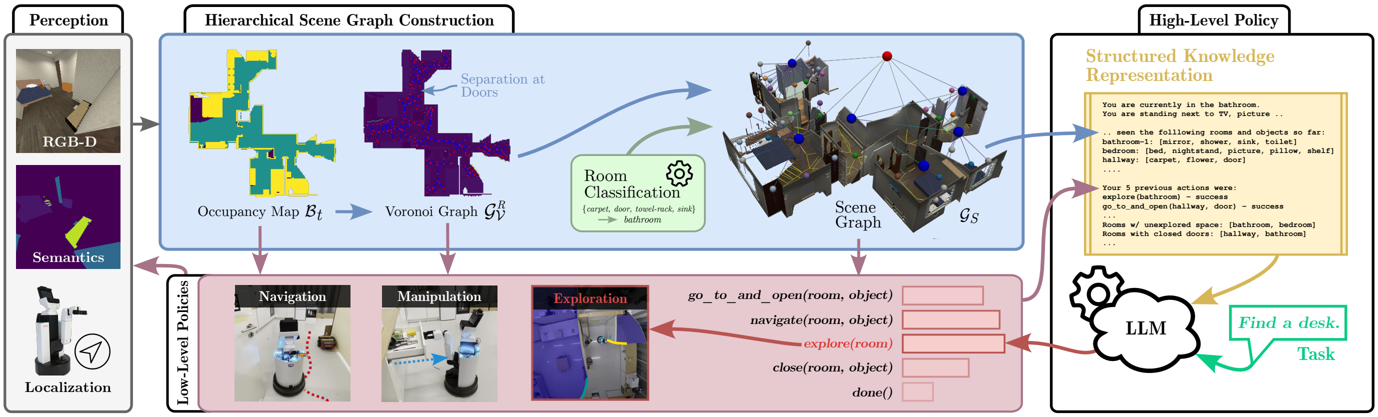
Language Models for Planning: Several recent works have investigated language models' abilities to generate task plans for robotic manipulation. These largely focus either on static table-top scenes of limited size and a limited number of objects [19, 26, 28], or a fully observable scene. On the other side of the spectrum, vision-language-navigation investigates pure navigation tasks in large apartments to either navigate along a described path or towards a specific instance of an object [13, 43]. A smaller number of works have investigated apartment-wide mobile manipulation tasks. LLM-Planner [43] uses information retrieval of the closest matching known task. Chalvatzaki *et al.* [5] finetune an LLM to encode object-object relations extracted from a scene graph. SayCan [21] combines affordance values with language scores. However, all of these methods focus on tasks restricted to single rooms. SayPlan [38] focuses on identifying relevant subgraphs in large known scene graphs by iteratively extracting or collapsing nodes. Ni *et al.* [34] learn a transformer-based model on top of a frozen LLM to predict subtasks from fully known scene graphs. In contrast, we focus on interactive search in large fully unexplored environments. As a result, simple prompting strategies, such as lists of observed objects [43, 14, 34] or raw JSON input [38] of a full scene graph to a language model becomes insufficient, as we demonstrate in our experiments.

Object Search has been tackled via a wide range of methods, including classical methods such as frontier exploration [49], vision-based reinforcement learning [6], or auditory signals [52]. Graph Neural networks (GNNs) have been used with scene graphs to find specific object instances with hierarchical and relational constraints [29] or in frequently changing, dynamic scenes [24, 50]. Schmalstieg *et al.* [41] introduced the interactive search task, in which an agent has to open doors and search through cabinets and drawers. While they focus on random target placements and a restricted number of objects and receptacles, we introduce a semantic single-object search variation of this task, which uses all objects in the scene and keeps the semantic co-occurrences in the scene intact.

Non-interactive semantic search has been previously tackled. Most recent methods used language models to extract similarities or co-occurrences with the target object to score frontiers [53, 8, 51] or predict potential functions towards a target object with supervised learning [37]. While these works focus on pairwise score calculations, we treat it as a planning problem in which the full scene is encoded jointly. In contrast to these works, we consider objects that are not freely accessible and require interaction with the environment and thereby reasoning over multiple steps such as opening doors and receptacles instead of pure directional reasoning. Lastly, given object detections, our representation and reasoning is fully open-vocabulary - both in terms of room and object categories. Conceptually most similar to our work, SayNav [36] utilizes a scene graph together with an LLM. However, it focuses on non-interactive search, restricting the LLM's access to a room subgraph, assumes restrictions such as knowledge about scene graph edges, and relies on a hardcoded heuristic of when to go to the next (already open) door.

III. PROBLEM STATEMENT: EMBODIED REASONING

In our setting, an embodied, robotic agent is situated in a large, unexplored environment and has to complete a given task, described by a language goal g. The agent is acting in a Partially Observable Markov Decision Process (POMDP)



```
You are currently in the bathroom.
You are standing next to TV, picture ..

.. seen the folllowing rooms and objects so far:
bathroom-1: [mirror, shower, sink, toilet]
bedroom: [bed, nightstand, picture, pillow, shelf]
hallway: [carpet, flower, door]
....

Your 5 previous actions were:
explore(bathroom) - success
go_to_and_open(hallway, door) - success
...
Rooms w/ unexplored space: [bathroom, bedroom]
Rooms with closed doors: [hallway, bathroom]
...
```

Fig. 2: MoMa-LLM: From posed RGB-D images and semantics, we construct a semantic 3D map from which we extract a various occupancy maps in the BEV space and construct a navigational Voronoi graph. Through room clustering and room-object assigments we then build up a hierarchical scene graph. From this scalable scene representation, we extract the task-relevant knowledge and encode it into a structured language representation. A large language model then produces high-level commands which are executed by low-level subpolicies. These in turn draw on and update the scene representations.

$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T(s'|s,a), P(o|s), r(s,a))$ where $\mathcal{S}, \mathcal{A}$ and $\mathcal{O}$ are the state, action and observation spaces, T and P describe the transition and observation probabilities, s, s' are the underlying current and next state, o is the agent's current observation consisting of posed RGB-D frame I_t, a is the current action and r is the reward. To succeed in these tasks, the agent has to perceive the environment and create a representation while reasoning about how to complete the tasks through exploration and interaction with the environment.

We introduce the task of *semantic interactive object search*. In contrast to most existing works [6, 53, 8, 40], interactive object search requires manipulation of the environment to navigate and explore it. As in realistic, human-centric environments, doors may block pathways and objects are not openly visible but may be stored away in receptacles like drawers or cabinets. We extend the interactive task introduced in [41] to a much larger number of objects and receptacles and a prior distribution of realistic room-object and object-object relations. As a result, other objects in the scene can provide valuable information about the position of the target. While existing tasks such as the Habitat challenge and Robothor use semantic placements, they do not support any physical interactions or objects placed within receptacles.

We implement the task in the iGibson scenes [27], consisting of 15 interactive apartments based on scans of real houses. At the beginning of an episode, all doors are closed and the agent is given a task description in natural language. The task is deemed successful if the agent has observed an instance of the target category and calls $done()$. The iGibson scenes contain realistic furniture and room distributions, but few other objects are placed in relation to this. We enrich the scenes with realistic object placements, both within receptacles and on top of furniture, by extending and matching previously introduced prior distributions P^{prior} over room and object relations [24], by aligning room names manually and matching object names via SBERT cosine similarities. We then assume that all objects that can be found on top of an object and that fit in size, can also be found inside it and vice-versa.. Given a valid scene instantiation, we then draw a target category $g \sim U(scene)$ from all categories in the scene. This results in the procedural generation of a wide range of tasks over 84 possible target classes. Full details can be found in the Supplementary Sec. S.1.

IV. MoMa-LLM

To address the challenges of interactive open-vocabulary household tasks, we propose *MoMa-LLM*, which intertwines high-level reasoning with scalable dynamic scene representations. We ground large-language models in hierarchical 3D scene graphs $\mathcal{G}_S$ that hold object- and room-level entities as well as a more fine-grained Voronoi graph for navigation. The LLM provides high-level actions that are executed through low-level skills as shown in Fig. 2. In general, we assume access to ground truth perception for semantic masks, depth, localization and handle detection as the focus of this work is on the reasoning aspect.

A. Hierarchical 3D Scene Graph

To provide an LLM with structured input, we craft a hierarchical scene graph that includes a navigational Voronoi graph.

1) Dynamic RGB-D Mapping: The agent perceives posed RGB-D frames $\{I_0, \dots, I_t\}$ including semantics from the environment. The contained points are transformed into the global coordinate frame and arranged on a 3D voxel grid $\mathcal{M}_t$. As we tackle an interactive problem, our map is dynamically updated based on novel explored areas or the occurrence of object dynamics in the scene. To infer obstacle positions, walls and explored free space, we first obtain the highest occupied entry per stixel in $\mathcal{M}_t$. These entries are then turned into a two-dimensional bird's-eye-view (BEV) occupancy map $\mathcal{B}_t$ by inferring all occupied positions except for those classified as free space $\mathcal{F}_t$. The latter in turn represents the navigable area that is used for robot exploration.

2) Voronoi Graph: Similar to Hydra [20], we abstract from the created dense maps by computing a navigational graph $\mathcal{G}_\mathcal{V}$. We first inflate $\mathcal{B}_t$ using an Euclidean signed distance field

(ESDF) formulation for robustness, but overwrite free space coordinates as given in $\mathcal{F}_t$ as zero. Based on this, we compute a Generalized Voronoi Diagram (GVD) that holds a set of points $\mathcal{V}$ with the same clearance to the closest obstacles:

$$\mathcal{V} = \{p \in \mathcal{F}_t | \{q \in \mathcal{B}_t | d(q,p) = \min_{q \in \mathcal{B}_t} d(q,p\}\}, \tag{1}$$

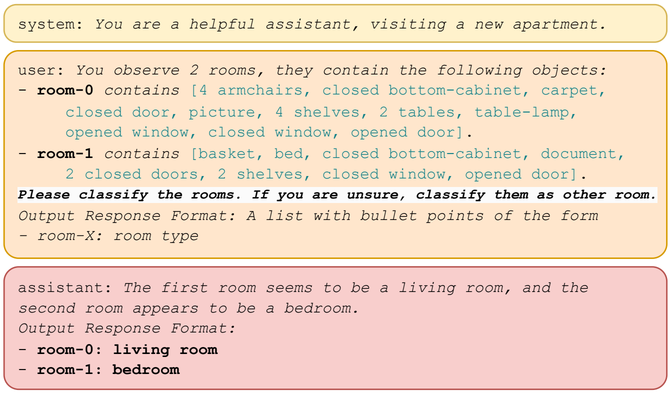
where obstacle points q are drawn from $\mathcal{B}_t$. After obtaining $\mathcal{V}$, we exclude all nodes that lie in the immediate vicinity of obstacles or do not reside within the occupied map boundaries. Given the path of obtained medial axes, we then construct edges $\mathcal{E}$ among $\mathcal{V}$ and obtain our navigational Voronoi graph $\mathcal{G}_\mathcal{V} = (\mathcal{V}, \mathcal{E})$. Throughout our experiments, we found that extracting the largest connected component of the graph provides the robot-centric Voronoi graph while other components commonly lie outside the explored area. Lastly, we sparsify $\mathcal{G}_\mathcal{V}$ to obtain fewer navigational nodes.

3) 3D Scene Graph: The MoMa-LLM-policy operates on an attributed 3D scene graph $\mathcal{G}_S$ that holds different abstraction levels, namely rooms and objects. We first separate the global Voronoi graph $\mathcal{G}_\mathcal{V}$ into multiple regions. To do so, we eliminate edges and nodes of $G_\mathcal{V}$ near doors instead of separating graphs at geometrical constrictions [20]. Using a mixture of Gaussians, we generate a two-dimensional probability distribution over all observed doors in the environment:

$$\rho_{\mathcal{N}}(\boldsymbol{x}, \boldsymbol{H}) = \frac{1}{N_D} \sum_{i=1}^{N_D} K_{\boldsymbol{H}}(\boldsymbol{x} - \boldsymbol{x_i}), \tag{2}$$

where $\boldsymbol{x}_i = (x_i, y_i)$ are the door center coordinates, $K_{\boldsymbol{H}}$ is the scaled Gaussian kernel of observed doors and $\boldsymbol{H}$ the bandwidth matrix, which we set to 2.0 based on manual tuning on the training scenes. Edges that exceed an empirically tuned probability threshold are disregarded along with isolated nodes. Following this principle, we obtain the separated Voronoi graph $\mathcal{G}_\mathcal{V}^R$ covering distinct rooms. In the next step, we infer the high-level connectivity among rooms by calculating the shortest paths between nodes of $\mathcal{G}_\mathcal{V}$ that belong to disjoint components of $\mathcal{G}_\mathcal{V}^R$. Whenever a path *traverses* just two distinct rooms as given by $\mathcal{G}_\mathcal{V}^R$, the two rooms count as immediate neighbors. Finally, we map objects to rooms. For each object $o \in \mathcal{G}_S$, we identify the node that minimizes the distance d_{vo} to the closest viewpoint v_p from which the object was seen. To this end, we calculate the shortest path from the object o to this viewpoint. It consists of the path on the Voronoi graph $\mathcal{G}_\mathcal{V}$, and the Euclidean distances d from the Voronoi nodes n_o and n_{v_p} to the object o and viewpoint v_p, respectively. By weighting the distance to the object with an exponent of $\lambda = 1.3$, we ensure to prefer nodes close to the object. Objects are then assigned to the room label R of the node n_o that minimizes Eq. (3). This prohibits the erroneous assignments of objects to a neighboring room through walls. Doors may be connected to multiple rooms.

$$d_{vo} = \min_{n_o, n_{v_p} \in \mathcal{G}_\mathcal{V}^R} path(n_o, n_{v_p}) + d(o, n_o)^\lambda + d(v_p, n_{v_p}) \tag{3}$$



```
system: You are a helpful assistant, visiting a new apartment.

user: You observe 2 rooms, they contain the following objects:
- room-0 contains [4 armchairs, closed bottom-cabinet, carpet,
     closed door, picture, 4 shelves, 2 tables, table-lamp,
     opened window, closed window, opened door].
- room-1 contains [basket, bed, closed bottom-cabinet, document,
     2 closed doors, 2 shelves, closed window, opened door].
Please classify the rooms. If you are unsure, classify them as other room.
Output Response Format: A list with bullet points of the form
- room-X: room type

assistant: The first room seems to be a living room, and the
second room appears to be a bedroom.
Output Response Format:
- room-0: living room
- room-1: bedroom
```

Fig. 3: Room Classification Prompt: based on the objects and room clusters of the scene graph, an LLM performs open-vocabulary classification.

4) Room Classification: Similar to Chen *et al.* [9], we perform room classification by providing an LLM with the set of object categories contained in each room. We perform this as open-set classification, in which we let the LLM freely pick the room categories deemed most appropriate. The resulting LLM prompts are detailed in Fig. 3. Room classification is performed in each high-level policy step, as the explored scene and scene graph evolve.

We provide a concise overview of all scene graph layers in Tab. S.1.

B. High-Level Action Space

We design an object-centric action space, which is tightly intertwined with the different granularities of the scene representation. It consists of the following high-level actions:

navigate(`room_name`, `object_name`): Navigation to an object in a room via an A* planner in the explored BEV-map $\mathcal{B}_t$, inflated by $0.1\,\mathrm{m}$. It first navigates to the Voronoi node associated with the object, then to the most central, free point on an arc around the object. This enables robust navigation to objects in partially explored space and ensures navigation to the correct room through the Voronoi assignment detailed in Sec. IV-A3. Navigation is considered successful if the agent reaches within $1.5\,\mathrm{m}$ of the object.
go_to_and_open(`room_name`, `object_name`): Navigate to a specific object, then open it. For doors, continue to navigate into the opened door frame.
close(`room_name`, `object_name`): Equivalent to opening.
explore(`room_name`): Move to an unexplored frontier within this room. Deemed successful if within $0.5\,\mathrm{m}$ of the frontier.
done(): Terminate the episode and evaluate if the target object has been found.

Ambiguities of multiple instances of the specified class in a room are resolved by selecting the closest instance. The sub-policies then generate actions in the low-level action space and return once they succeed or encounter a failure. Throughout their execution, they continuously update the scene representations. Refer to the Supplementary Sec. S.1 for details.

C. Grounded High-Level Planning

We encode the accumulated knowledge of the scene graph into natural language by extracting the relevant components and embedding them in a problem-specific structured manner. Our method fulfills three properties: (i) grounding - guiding the LLM to adhere to the physical realities of the scene, (ii) specificity - avoiding long or irrelevant context queries that increase hallucinations and the difficulty of the planning problem [38, 1], and (iii) open-set - our reasoning is open-vocabulary and performs in a zero-shot manner, enabling direct deployment with unknown semantics and perception models. The resulting prompt for the language model is shown in Fig. 4. In the following, we describe the main components of structured encoding. We demonstrate the importance of this structure in Sec. V.

1) Scene Structure: We encode the main room-object structure from the scene graph into a structured list of rooms and their containing objects and encode path distances (based on an A*-planner) by binning them and mapping them to adjectives [5], as detailed in Supplementary Sec. S.4. We then employ the following filtering to allow for compact text encodings: we summarize matching nodes within a room with a counter, we filter out open doors that provide no new connectivity, and we encode object states directly within the object name, e.g. as *"opened"* or *"closed [object-name]"*.

2) Partial Observability: As the environment is initially unknown, it requires explicit reasoning about exploration-exploitation trade-offs. We identify frontiers to explorable areas [49], then leverage the scene graph to provide them with semantic meaning. Firstly, we associate each frontier with a room through matching with $G_{\mathcal{V}}^{R}$. Secondly, we apply hole-filling to the BEV map to differentiate whether a frontier is an encapsulated area within a room, such as occluded space behind furniture, or whether the frontier is leading out to new areas. Correspondingly, we then represent them as *"unexplored area"* within a room, while frontiers that lead to other areas are listed separately, see Fig. 4. The second type of unexplored space is receptacles that may contain target objects. Together with the encoded object states, we find that the language model is capable of inferring affordances from the object descriptions, removing the need to explicitly encode them. If trying to open objects that cannot be opened, the according subpolicy will fail and the LLM has to reason about an appropriate response.

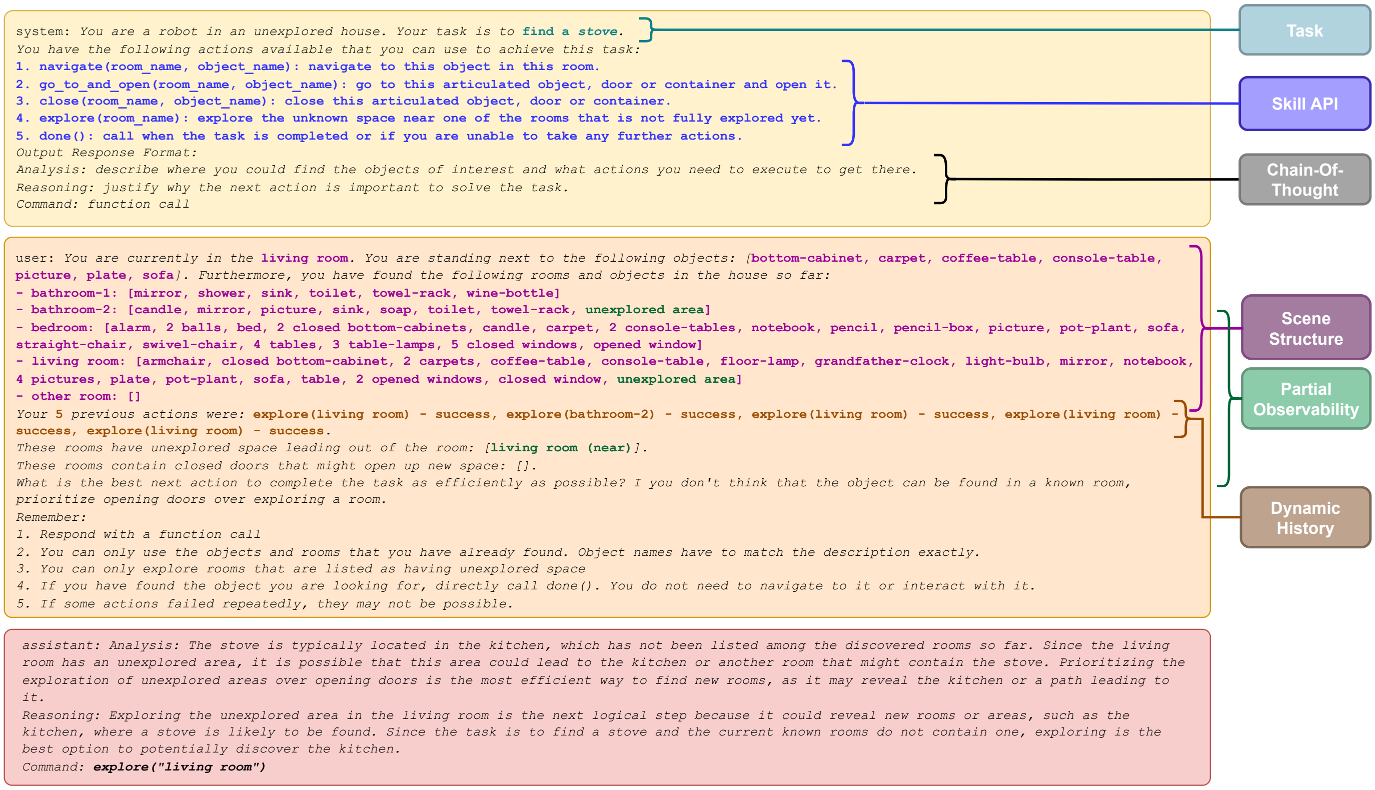
3) History in Dynamic Scenes: Given the size of the scenes, the conversation history quickly grows too large to provide to a language model directly. Instead, we aim to find the most compact representation of previous actions to fulfill the Markov property. For each high-level decision, we encode the latest scene representation and start a new query to the LLM. As the scene representation is dynamically updated, this automatically encodes all newly acquired knowledge. To account for previous interactions, we provide the LLM with a history of the last h actions. But as the scene graph changes dynamically, the previous room- and object-centric function calls may no longer match the current scene. Instead, we keep track of interaction positions, and then re-align the previous actions by matching the positions to their closest Voronoi nodes and associated room labels. We then provide the LLM with a list of the re-aligned function calls, as shown in Fig. 4. E.g., the agent executes *go_to_and_open(*`living room`*,* `cabinet`*)*. But revealing a fridge, later classifies the same room as kitchen. The realigned history will then correctly reflect this action as *go_to_and_open(*`kitchen`*,* `cabinet`*)*.

4) Re-trial and Re-planning: Extracting meaningful feedback for failure reasons for robots in the real world remains an open problem [30], as the number of possible failure reasons is almost unlimited. Instead, we provide very limited feedback about subpolicy success, which can be readily generated in the real world. We rely on a simple success state to the action history, stating *"success"*, *"failure"*, or *"invalid argument"* in case the output of the LLM could not be matched to the scene graph. We differentiate two cases of replanning: if the agent attempted interactions or commands that cannot be parsed or are deemed infeasible without attempting execution, we have not gained any new information about the scene, and we continue the conversation with the message *"The last action <function-call> failed. Please try another command."*. In case of more than five failures without state change, we terminate the episode as unsuccessful. If a subpolicy attempted execution but failed to complete its task, we re-encode the latest scene, update the action history, and let the LLM make a normal next decision with the updated state.

V. EXPERIMENTS

As language models, we use *gpt-4-1106-preview* for the high-level reasoning and *gpt-3.5-turbo-1106* for the room classification task [35]. For simplicity, we recompute the scene graph each time-step. More advanced implementations would reduce costs through incremental updates.

Baselines: We compare our approach against heuristic-based and recent learning-based and language-based methods. We provide all baselines except Unstructured LLM with a ground truth $done()$ decision when the object has been observed.
Random: uniform random choice among all available actions (detected frontiers and closed objects).
Greedy: greedily triggers the closest available action based on the shortest path calculated by an A*-planner.
ESC-Interactive: ESC is a recent approach for semantic object search [53] which scores frontiers based on object-object and object-room co-occurrences as well as their distance. We extend the approach to interactive search by using the same rules to score openable objects and then select the action with the highest value. Co-occurrences are based on similarities of a finetuned Deberta-v3 language model [17], following the authors' instructions. To isolate the impact of the decision making, we use the same scene graph and low-level policies as for our method.
HIMOS: A hierarchical reinforcement learning approach [41]



```
system: You are a robot in an unexplored house. Your task is to find a stove.
You have the following actions available that you can use to achieve this task:
1. navigate(room_name, object_name): navigate to this object in this room.
2. go_to_and_open(room_name, object_name): go to this articulated object, door or container and open it.
3. close(room_name, object_name): close this articulated object, door or container.
4. explore(room_name): explore the unknown space near one of the rooms that is not fully explored yet.
5. done(): call when the task is completed or if you are unable to take any further actions.
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 are currently in the living room. You are standing next to the following objects: [bottom-cabinet, carpet, coffee-table, console-table,
picture, plate, sofa]. Furthermore, you have found the following rooms and objects in the house so far:
- bathroom-1: [mirror, shower, sink, toilet, towel-rack, wine-bottle]
- bathroom-2: [candle, mirror, picture, sink, soap, toilet, towel-rack, unexplored area]
- bedroom: [alarm, 2 balls, bed, 2 closed bottom-cabinets, candle, carpet, 2 console-tables, notebook, pencil, pencil-box, picture, pot-plant, sofa,
straight-chair, swivel-chair, 4 tables, 3 table-lamps, 5 closed windows, opened window]
- living room: [armchair, closed bottom-cabinet, 2 carpets, coffee-table, console-table, floor-lamp, grandfather-clock, light-bulb, mirror, notebook,
4 pictures, plate, pot-plant, sofa, table, 2 opened windows, closed window, unexplored area]
- other room: []
Your 5 previous actions were: explore(living room) - success, explore(bathroom-2) - success, explore(living room) - success, explore(living room) -
success, explore(living room) - success.
These rooms have unexplored space leading out of the room: [living room (near)].
These rooms contain closed doors that might open up new space: [].
What is the best next action to complete the task as efficiently as possible? I you don't think that the object can be found in a known room,
prioritize opening doors over exploring a room.
Remember:
1. Respond with a function call
2. You can only use the objects and rooms that you have already found. Object names have to match the description exactly.
3. You can only explore rooms that are listed as having unexplored space
4. If you have found the object you are looking for, directly call done(). You do not need to navigate to it or interact with it.
5. If some actions failed repeatedly, they may not be possible.
```

```
assistant: Analysis: The stove is typically located in the kitchen, which has not been listed among the discovered rooms so far. Since the living
room has an unexplored area, it is possible that this area could lead to the kitchen or another room that might contain the stove. Prioritizing the
exploration of unexplored areas over opening doors is the most efficient way to find new rooms, as it may reveal the kitchen or a path leading to
it.
Reasoning: Exploring the unexplored area in the living room is the next logical step because it could reveal new rooms or areas, such as the
kitchen, where a stove is likely to be found. Since the task is to find a stove and the current known rooms do not contain one, exploring is the
best option to potentially discover the kitchen.
Command: explore("living room")
```

Fig. 4: High-level Reasoning Prompt: We encode the extracted scene representation to natural language, providing structured information to a language model.

which learns to combine frozen low-level policies for interactive object search, based on a semantic map memory. We adapt it by giving it the same subpolicies as our approach and scale it to the much larger number of objects in our scenes by restricting the instance navigation to target and articulated objects.
Unstructured LLM: This baseline provides the scene graph in a JSON format without any additional structure to the language model. The prompt structure is derived from SayPlan [38], adapted to the instructions and scene graph of our method. See Supplementary Sec. S.5 for a prompt example.
MoMa-LLM w/ Hydra: We incorporate the room segmentation approach introduced by Hydra [20] into our scene graph construction pipeline to measure the impact of our proposed door-wise room separation mechanism.

Metrics: We use three types of metrics to evaluate methods.
Success rate (SR): the share of episodes in which the agent finds the target object. We terminate an episode if the agent reaches 50 high-level steps, indicating being stuck.
Success weighted by Path Length (SPL) [2] calculates the fraction of distance traveled to the shortest possible path and weights it by whether the episode was successful. This metric does not take into account the costs of object interactions.
Search efficiency curve and AUC-E: While the commonly used success and SPL metrics allow for reducing the evaluation to a single number, they rely on an arbitrarily set maximum allowed time budget or number of environment steps. As a result, these metrics do not differentiate between methods that search thoroughly but on average take more steps versus methods that search large areas very quickly, but that might fail to search every corner for small objects. Instead, we reason that the desired time budget depends heavily on the use case and propose a new metric that evaluates the full efficiency curve. For each possible budget (number of steps), we calculate the share of episodes that succeeded with this or fewer number of steps. This results in an efficiency curve, in which the best policies are located in the top left corner, enabling the comparison of success rates for arbitrary budgets. We can still reduce this to a single number by calculating the area under the efficiency curve (*AUC-E*). A perfect (but unachievable) policy, that finds all objects in a single step will have a value of one, a policy that does not find any objects will have a value of zero. We calculate the integral up to 5,000 low-level steps, at which points almost all methods make no further progress.

A. *Simulation Experiments*

We instantiate the task in the iGibson simulator [27] with a Fetch robot. In contrast to previous LLM-based works [38, 36], we evaluate all approaches in completely unseen apartments, following the data split of the iGibson challenge into eight training scenes for the development of all modules and prompt engineering and seven test scenes. For each scene, we evaluate the agent over 25 procedurally generated episodes with randomized start poses, target objects, and object distributions.

Scene Understanding: We compare our door-based room separation algorithm against Hydra [20], which separates a

TABLE I: INTERACTIVE OBJECT SEARCH RESULTS IN SIMULATION

Model	SR	SPL	AUC-E	Object Interactions	Distance Traveled	Infeasible Actions
Random	93.1	50.2	77.0	5.7	32.9	–
Greedy	85.7	50.9	72.9	8.1	22.3	–
ESC-Interactive	95.4	62.7	84.5	4.1	19.6	–
HIMOS	93.7	48.5	77.4	4.8	35.9	–
Unstructured LLM	86.3	59.4	77.6	3.6	18.5	0.41
MoMa-LLM w/ Hydra	92.0	61.9	84.3	**2.7**	**12.9**	**0.06**
MoMa-LLM (ours)	**97.7**	**63.6**	**87.2**	3.9	18.2	0.19
Ours w/o frontiers	79.4	55.0	72.2	4.3	15.6	0.91
Ours w/o history	94.9	63.0	84.1	3.6	17.1	0.26
Ours w/ room-history	97.1	63.0	86.6	3.8	17.8	0.28
Ours w/o distances	97.1	61.5	86.4	3.8	18.9	0.24

Notes: Best and second best highlighted in bold and underline. Object interactions, distance travelled and infeasible actions averaged over all episodes - including early terminated failures. Infeasible Actions: average number of steps the LLM produced an action that could not be executed, resulting in re-planning with continued conversation (cf. Sec. IV-C.4).

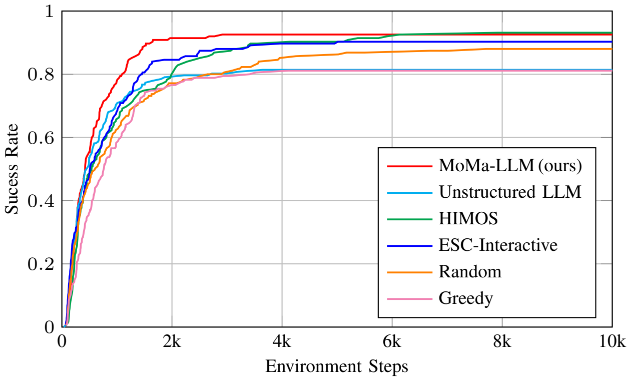


Fig. 5: Interactive search efficiency curve in simulation. Each point depicts the success rate for a given maximum time budget (x-axis).

Voronoi graph of places based on dilating obstacles. We evaluate the room segmentation precision and recall as defined by Hughes *et al.* [20] and the separated Voronoi graph purity at all high-level policy steps to account for robustness throughout exploration. As depicted in Tab. II, we observe greater average precision and recall of MoMa-LLM in terms of dense region segmentation as well as a lower variance across time. The sparsely evaluated graph purity shows that the separated Voronoi graphs of MoMa-LLM cover fewer ground-truth rooms per predicted room than graphs produced by Hydra, which may produce inferior results when facing non-apparent constrictions or object clutter. We found that real-world scenes, as the ones contained in iGibson, regularly feature constant-diameter corridors and narrow passages due to furniture placements, which impede detecting rooms based on geometrical constrictions. This demonstrates that room separation algorithms benefit from semantic cues such as detected doors, door frames, archways, or changing floor materials at room boundaries. We found our policy to be robust to under-segmented rooms even though objects from multiple rooms were, e.g., considered part of a single room. By relying on the camera pose from which an object is observed we reduce the number of wrong object-room assignments *through* walls. Following the door-wise separation of rooms, our approach however is prone to *open* room concepts such as combined kitchen and living rooms. For more information and graph depictions, refer to the Supplementary Sec. S.1.

TABLE II: ENVIRONMENT PARTITIONING THROUGHOUT EXPLORATION

Approach	Precision		Recall		Purity ↑
	μ ↑	σ ↓	μ ↑	σ ↓	
Hydra	0.621	0.081	0.943	0.044	0.562
MoMa-LLM	**0.666**	**0.064**	**0.948**	**0.032**	**0.615**

Dense room segmentation precision and recall as defined in Hughes *et al.* [20] in terms of mean and standard deviation throughout exploration. The purity (Supplementary Sec. S.6.1.A) measures the number of ground-truth rooms erroneously captured per predicted room given sparse Voronoi graphs. Evaluated across 10 episodes and all test scenes with 2D grid resolution of $0.05\,\mathrm{m}$ to account for thin walls. Best values are written bold.

Policies: The results and efficiency curves for the search task are shown in Tab. I and Fig. 5. We find that, given appropriate subpolicies, heuristics can complete a significant share of episodes. However, they are not sufficient for an efficient search strategy, resulting in low SPL and AUC-E. Similarly, while HIMOS achieves a high success rate, it is unable to explore efficiently. We found that the RL agent struggled with the much larger action space that resulted from the many more interactable instances in our scenes than in the original work. ESC in contrast, is able to exploit the co-occurrences to improve over the other baselines. However, given its pair-wise comparisons, it is unable to optimize over longer action sequences. In contrast, MoMa-LLM achieves similar success rates as HIMOS with a much higher search efficiency, both in terms of SPL and AUC-E. We find that the structured prompt representation is essential for this, with the Unstructured LLM performing much worse. We then perform an additional number of ablations of the language encodings. We find that encoding the frontiers is very important. Removing the history also leads to a, although smaller, drop in performance. Even a coarser representation of the history, consisting of only a list of visited rooms similar to [38], is already beneficial, but slightly worse than the full action history. Lastly, we evaluate the impact of not encoding distances nor nearby objects and also find a small drop in performance.

This picture is fortified by the full efficiency curves in Fig. 5, which show that the MoMa-LLM-based approaches achieve the highest performance for all given time budgets, with only MoMa-LLM w/ Hydra being more efficient for some of the small budgets, but not overall. In contrast, random heuristics achieve very high coverage, resulting in good success rates, but often take very long to find specific objects. Further examining the different models, we find that MoMa-LLM both travel much shorter distances and open fewer objects on average, indicating efficient and target-driven behavior. In contrast, Unstructured LLM produces almost 50% more invalid actions. Qualitatively, we find that MoMa-LLM is robust to various room layouts, such as "combined kitchen and living rooms"

TABLE III: INTERACTIVE OBJECT SEARCH RESULTS IN THE REAL WORLD

Model	Success Rate	Navig Fails	Manip Fails	Distance Traveled	Object Interact.
ESC-Inter.	80%	2	0	33.9	3.5
MoMa-LLM	80%	1	1	**17.9**	**2.2**

Notes: Dist. travelled is the average distance travelled per episode in meters. Object interactions are the average number of object interactions per episode.

that result in very large room clusterings and can handle the open-vocabulary room classification well. In contrast, Hydra tends to predict a large number of small rooms. For reasoning examples, refer to the Supplementary Sec. S.6.

B. Real-World Experiments

We then transfer our policy to the real world. We create a real-world apartment, consisting of four rooms: a combined kitchen and dining room, a living room, a long hallway, and a bathroom. We use a Toyota HSR robot, equipped with an RGB-D camera and a 270° LiDAR. We replace the navigation policies with the ROS Nav Stack and the manipulation actions with the N^2M^2 manipulation policies [18]. We rely on the same assumptions as in simulation and assume access to localization, accurate semantic perception, and handle detection. We implement this by pre-recording a map with the robot's LiDAR and annotating it with semantic labels. At test time, we create an occupancy map from the robot's RGBD camera and reveal the corresponding part of the semantic map to the agent. The pre-recorded map is also used for localization. To detect handles, we use AR-Markers placed on each object. For details refer to the Supplementary Sec. S.2.

We evaluate both MoMa-LLM and the most efficient baseline, ESC, on identical start positions and target categories. The results are shown in Tab. III, Fig. 6, and the accompanying video. Both methods succeeded in 8/10 episodes, demonstrating the successful transfer of the system to the real world. We find that the Voronoi- and scene graph construction transfer directly to the unseen and quite different layout. Similarly, the system directly transfers to the change in subpolicies, where the mobile manipulation policies ensure a the transition between all subpolicies. The two failures stemmed from irrecoverable failures of the subpolicies, in particular, collisions of the base during navigation or of the arm while opening the door. Comparing the methods, we find confirmation of the simulation results, with MoMa-LLM moving and opening objects more target-driven and efficiently. Furthermore, the agent was able to react to the (unseen) failure cases of the subpolicies, such as re-trying to open a drawer when the gripper slipped off the handle.

C. Towards General Household Tasks

As we move to more abstract and complex tasks, it becomes increasingly difficult to define problem-specific rules or heuristics. In contrast, our approach is readily expandable to a wide range of household and mobile manipulation tasks. Representative of this, we introduce a *fuzzy search task*. In this task, the robot does not receive a specific object class to find, but rather a fuzzy description, such as *"I am hungry. Find me something for breakfast"*. The full set of queries are shown in Tab. S.4. We find that the agent is capable of finding objects that satisfy respective queries, and correctly reasoning about task completion by calling *done()*. We further test this capability with three tasks that cannot be solved with the given subpolicies (bottom part of Tab. S.4). For these cases, the agent terminated the episode after finding the relevant objects, reasoning that these objects would now be sufficient for further completion of the tasks. This demonstrates the flexibility of our approach. We leave the extension to arbitrary tasks to future work.

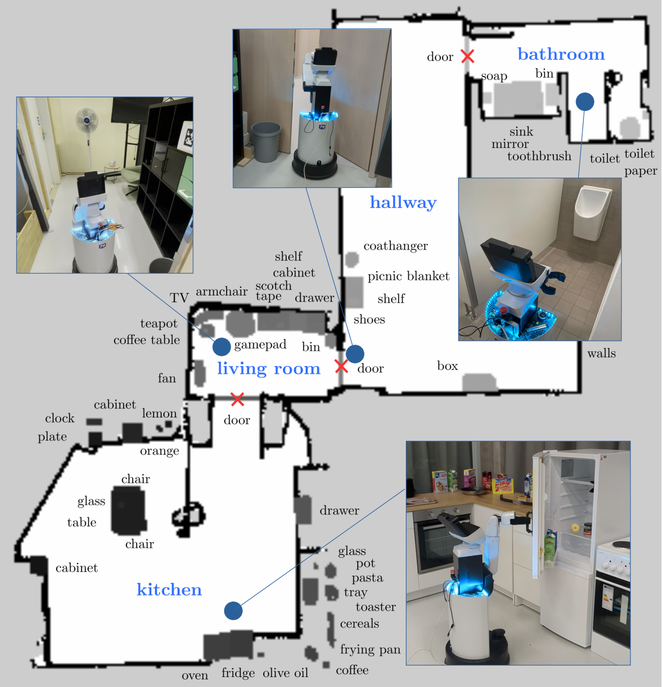


Fig. 6: We construct a real-world apartment covering four rooms and 54 objects and transfer the model to a Toyota HSR robot.

VI. CONCLUSION

We developed a method to ground language models for high-level reasoning with scalable, dynamic scene graphs and efficient low-level policies for interactive tasks that require combined reasoning about manipulation, navigation and exploration. We demonstrated the importance of extracting structured knowledge for large and unexplored scenes to enable LLMs to reason about efficient search strategies, outperforming fully learned or co-occurrence-based methods. We then transferred our method to a real-world apartment, achieving consistent performance over many episodes. Lastly, we demonstrate the extendability of our approach to abstract tasks, opening the door towards general household tasks.

REFERENCES

[1] Christopher Agia, Krishna Murthy Jatavallabhula, Mohamed Khodeir, Ondrej Miksik, Vibhav Vineet, et al. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Proc. of the Conf. on Rob. Learning*, pages 46–58, 2022.

[2] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[3] Miguel Arduengo, Carme Torras, and Luis Sentis. Robust and adaptive door operation with a mobile robot. *Intelligent Service Robotics*, 14(3):409–425, 2021.

[4] Iro Armeni, Zhi-Yang He, Amir Zamir, Junyoung Gwak, Jitendra Malik, Martin Fischer, and Silvio Savarese. 3D scene graph: A structure for unified semantics, 3D space, and camera. In *Int. Conf. on Computer Vision*, 2019.

[5] Georgia Chalvatzaki, Ali Younes, Daljeet Nandha, An Thai Le, Leonardo FR Ribeiro, and Iryna 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*, 10, 2023.

[6] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *Int. Conf. on Learn. Repr.*, 2020.

[7] Arnav Chavan, Raghav Magazine, Shubham Kushwaha, Mérouane Debbah, and Deepak Gupta. Faster and lighter llms: A survey on current challenges and way forward. *arXiv preprint arXiv:2402.01799*, 2024.

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

[9] William Chen, Siyi Hu, Rajat Talak, and Luca Carlone. Leveraging large language models for robot 3d scene understanding. *arXiv preprint arXiv:2209.05629*, 2022.

[10] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, 2022.

[11] Nikhil Gosala, Kürsat Petek, Paulo LJ Drews-Jr, Wolfram Burgard, and Abhinav Valada. Skyeye: Self-supervised bird's-eye-view semantic mapping using monocular frontal view images. In *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 14901–14910, 2023.

[12] Elias Greve, Martin Büchner, Niclas Vödisch, Wolfram Burgard, and Abhinav Valada. Collaborative dynamic 3d scene graphs for automated driving. *Int. Conf. on Robotics & Automation*, 2024.

[13] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. In *Proc. of the Assoc. for Computational Linguistics*, pages 7606–7623, 2022.

[14] Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. *Int. Conf. on Robotics & Automation*, 2024.

[15] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *Int. Conf. on Learn. Repr.*, 2022.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 2961–2969, 2017.

[17] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *Int. Conf. on Learn. Repr.*, 2022.

[18] Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. $\mathrm{N}^2\mathrm{m}^2$: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments. *IEEE Transactions on Robotics*, 2023. doi: 10.1109/TRO.2023.3284346.

[19] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Proc. of the Conf. on Rob. Learning*, 2023.

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

[21] Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Proc. of the Conf. on Rob. Learning*, 2022.

[22] Markus Käppeler, Kürsat Petek, Niclas Vödisch, Wolfram Burgard, and Abhinav Valada. Few-shot panoptic segmentation with foundation models. *arXiv preprint arXiv:2309.10726*, 2023.

[23] Weicheng Kuo, Yin Cui, Xiuye Gu, AJ Piergiovanni, and Anelia Angelova. Open-vocabulary object detection upon frozen vision and language models. In *Int. Conf. on Learn. Repr.*, 2023.

[24] Andrey Kurenkov, Michael Lingelbach, Tanmay Agarwal, Emily Jin, Chengshu Li, Ruohan Zhang, et al. Modeling dynamic environments with scene graph memory. In *Int. Conf. on Mach. Learning*, pages 17976–17993, 2023.

[25] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of field robotics*, 36(2):416–446, 2019.

[26] Boyi Li, Philipp Wu, Pieter Abbeel, and Jitendra Malik. Interactive task planning with language models. In *2nd Workshop on Lang. and Robot Learning: Language as Grounding (CoRL)*, 2023.

[27] Chengshu Li, Fei Xia, Roberto Martín-Martín, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *Proc. of the Conf. on Rob. Learning*, volume 164, pages 455–465, 08–11 Nov

2022.

[28] Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion: from natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365, 2023.

[29] Michael Lingelbach, Chengshu Li, Minjune Hwang, Andrey Kurenkov, Alan Lou, Roberto Martín-Martín, Ruohan Zhang, Li Fei-Fei, and Jiajun Wu. Task-driven graph attention for hierarchical relational object navigation. In *Int. Conf. on Robotics & Automation*, pages 886–893. IEEE, 2023.

[30] Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *Proc. of the Conf. on Rob. Learning*, 2023.

[31] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[32] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *Proc. of the Europ. Conf. on Computer Vision*, pages 728–755. Springer, 2022.

[33] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection. *Proc. of the Conf. on Neural Information Processing Systems*, 36, 2024.

[34] Zhe Ni, Xiao-Xin Deng, Cong Tai, Xin-Yue Zhu, Xiang Wu, Yong-Jin Liu, and Long Zeng. Grid: Scene-graph-based instruction-driven robotic task planning. *arXiv preprint arXiv:2309.07726*, 2023.

[35] OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.

[36] Abhinav Rajvanshi, Karan Sikka, Xiao Lin, et al. Saynav: Grounding large language models for dynamic planning to navigation in new environments. *arXiv preprint arXiv:2309.04077*, 2023.

[37] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 18890–18900, 2022.

[38] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, et al. Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. *Proc. of the Conf. on Rob. Learning*, 2023.

[39] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proc. of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, November 2019. doi: 10.18653/v1/D19-1410.

[40] Fabian Schmalstieg, Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. Learning long-horizon robot exploration strategies for multi-object search in continuous action spaces. In *Robotics Research*, pages 52–66, 2022.

[41] Fabian Schmalstieg, Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. Learning hierarchical interactive multi-object search for mobile manipulation. *IEEE Robotics and Automation Letters*, 2023.

[42] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 10740–10749, 2020.

[43] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Int. Conf. on Computer Vision*, pages 2998–3009, 2023.

[44] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[45] Niclas Vödisch, Kürsat Petek, Wolfram Burgard, and Abhinav Valada. Codeps: Online continual learning for depth estimation and panoptic segmentation. In *Robotics: Science and Systems*, 2023.

[46] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. *Robotics: Science and Systems*, 2024.

[47] Pengying Wu, Yao Mu, Bingxian Wu, Yi Hou, Ji Ma, Shanghang Zhang, and Chang Liu. Voronav: Voronoi-based zero-shot object navigation with large language model. *arXiv preprint arXiv:2401.02695*, 2024.

[48] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. SceneGraphFusion: Incremental 3D scene graph prediction from RGB-D sequences. In *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 7515–7525, 2021.

[49] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. of the IEEE Int. Symp. on Comput. Intell. in Rob. and Aut.*, 1997.

[50] Zhongmou Ying, Xianfeng Yuan, Baojiang Yang, Yong Song, Qingyang Xu, Fengyu Zhou, and Weihua Sheng. Rp-sg: Relation prediction in 3d scene graphs for unobserved objects localization. *IEEE Robotics and Automation Letters*, 2023.

[51] Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. Vlfm: Vision-language frontier maps for zero-shot semantic navigation. *Int. Conf. on Robotics & Automation*, 2024.

[52] Abdelrahman Younes, Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. Catch me if you hear me: Audio-visual navigation in complex unmapped environments with moving sounds. *IEEE Robotics and*

Automation Letters, 8(2):928–935, 2023.

[53] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. *Int. Conf. on Mach. Learning*, 2023.