

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

## - Supplementary Material -

Daniel Honerkamp<sup>1\*</sup>, Martin Büchner<sup>1\*</sup>, Fabien Despinoy<sup>2</sup>, Tim Welschehold<sup>1</sup>, Abhinav Valada<sup>1</sup>

In this supplementary material, we provide additional details on the approach and the experimental setup, including the simulation and real-world environments, the subpolicies, and the baselines. Moreover, we provide additional results from the experiments. Further examples of real-world experiments are demonstrated in the video material.

### S.1. 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 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. III-A. Navigation is considered successful if the agent reaches within 1.5 m of the object.
- *go\_to\_and\_open(room\_name, object\_name)*: Navigation to a specific object, then open it. For doors, continue to navigate into the opened door frame.
- *close(room\_name, object\_name)*: Equivalent to the opening action.
- *explore(room\_name)*: Move to an unexplored frontier within this room. Deemed successful if the agent reaches within 0.5 m of the frontier point.
- *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 subpolicies 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.

\*These authors contributed equally.

<sup>1</sup>Department of Computer Science, University of Freiburg, Germany.

<sup>2</sup>Toyota Motor Europe (TME)

Project page: <http://moma-llm.cs.uni-freiburg.de>

### S.2. SIMULATION ENVIRONMENT

#### A. Task Generation

We implement the task in the iGibson scenes [19], 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()*.

We make the following modifications to the iGibson scenes:

- We close all exterior doors and filter them out of the scene graph as they lead out to empty space.
- We scale the size of the Fetch robot by a factor of 0.85 as otherwise it is too large to navigate a significant number of areas.
- We remove six doors that block the robot’s pathway when opening (*door\_52* in *Pomaria\_0\_int*, *door\_75* in scene *Beechwood\_1\_int* and *door\_106* in *Ihlen\_1\_int*, *door\_90* in *Merom\_1\_int*, *door\_138* and *door\_139* in *Wainscott\_0\_int*).
- We rename four object categories, with misleading asset names, such as renaming *breakfast\_table* (which can be found e.g. in bedrooms) to *table*.

The iGibson scenes contain realistic furniture and room distributions, but few other objects are placed in relation to this. We provide a method to 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 [20]. 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.

We first align the rooms and objects with the prior distribution. We manually match room labels and use cosine similarities of object name embeddings to match objects. We embed the object names from the scene and from the prior distribution with SBERT [1]. We then define a match as a cosine similarity  $\leq 0.7$  and being in the top 50 matched categories. As  $P^{prior}$  only contains two "inside" relationships, we enrich these relationships by assuming 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. This results in an extended distribution  $P^{prior,ext}$ . We then procedurally enrich each episode by drawing additional objects from the set of admissible room-object-relations for all existing furniture

in the iGibson scenes. We keep drawing until the physical placement of a relation succeeds or the distribution is exhausted. Given a valid scene instantiation, we draw a target category  $g \sim U(scene)$  from all categories in the scene. We reject targets as infeasible if no target instance is reachable from the agent’s random start position. We also reject all episodes in which the target object is immediately visible. We terminate an episode early if the agent reaches 50 high-level steps, indicating being stuck.

### B. Perception

The robot in simulation is equipped with a differential drive and an RGB-D camera with a resolution of  $256 \times 256$  pixels and a vertical field of view of  $120^\circ$ . As the focus of this work is on decision making, we abstract from imperfect perception and assume access to ground truth instances and semantic segmentation from the simulator. For a realistic detection range, we restrict all sensors (depth, semantics) to a maximum range of 5 m and set a threshold of 50 segmented pixels before an object is considered detected - except for object with a volume below  $0.01 \text{ m}^3$ . We furthermore assume accurate detection of whether an articulated object is open or closed and assume that objects within receptacles are detected after opening the object. We construct all maps at a resolution of 0.075 m and remap the *floor* and *carpet* categories as free space.

### C. Execution

All methods start by fully turning around in-place to initialize the scene graph.

*Low-level action space:* The subpolicies act in a low-level action space consisting of the following actions:

- move forward by 7.5 cm
- turn-left by up to 35 rad
- turn-right by up to 35 rad
- open articulated object
- close articulated object
- done: end the episode and evaluate the success

*Navigation:* To reduce run-time, navigation actions are executed without physical simulation, but rather by directly setting the state of the robot. If no complete path in free space exists, the navigation policy will consider unexplored areas as traversable and replan with newly revealed free space. It will return ‘failed’ if it cannot find a path or reaches too many replanning attempts.

*Object interactions:* We follow previous work [2], [9] and execute articulated object manipulations as “magic actions” in simulation. These actions directly set the joint values of the object to their minimum or maximum. In the calculation of the efficiency curves and AUC-E, we weigh these actions by a time factor of 30 to make their time cost comparable to an execution duration of roughly 30 seconds.

The *go\_to\_and\_open()* action is implemented as follows:

- 1) Navigate to the Voronoi node closest to the object

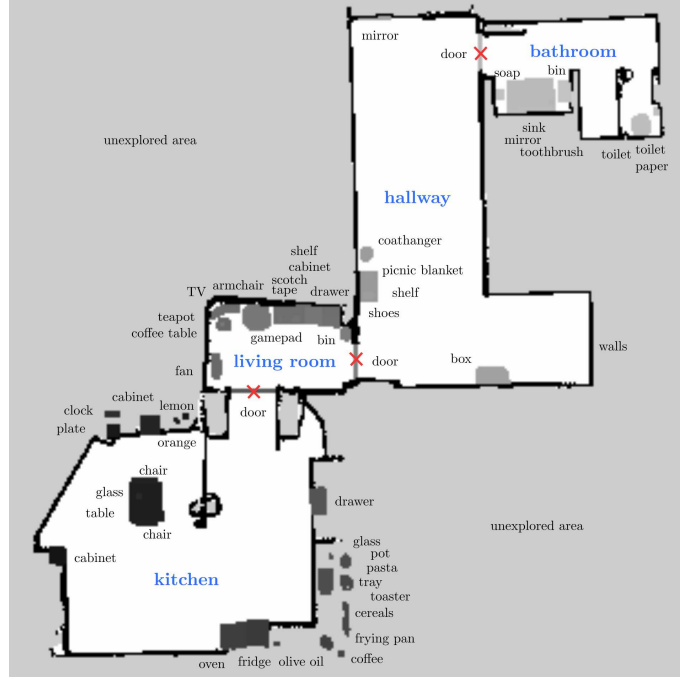


Fig. S.1. Map of the real-world environment. Different intensities represent different object instances in the environment. Room annotations are for illustration and not used by our model. Object labels denote approximate object positions.

- 2) Navigate to the most central free space in front of the object and turn toward it
- 3) Trigger the magic open action

## S.3. REAL-WORLD ENVIRONMENT

The HSR robot is equipped with an RGB-D camera and a  $270^\circ$  LiDAR. We replace the navigation policies with the ROS Nav Stack and the manipulation actions with the N<sup>2</sup>M<sup>2</sup> manipulation policies [21]. 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.

### A. Map

Fig. S.1 shows the map of the real-world environment. Different object instances are indicated in different intensities. The environment covers five different room types and 54 different object categories, including furniture such as tables, chairs, and a coat hanger, as well as small and less common objects such as soap, gamepad, or scotch tape. We set start positions for the robot in the kitchen, living room, and hallway. We ensure the same initial positions and targets for all evaluated methods.

## B. Execution

**Navigation:** We use the default manufacturer versions of the ROS NavStack as developed by Toyota for navigation. It uses the robot’s LiDAR and depth cameras for dynamic obstacle detection and navigates in a map inflated by 0.25 m.

**Mobile manipulation:** Articulated object interactions are executed with pretrained N<sup>2</sup>M<sup>2</sup> manipulation policies [21]. The policy receives the handle pose, detected through AR-Markers, and uses a local occupancy map constructed from the LiDAR of the robot for obstacle avoidance. For each articulated object, we collect a single demonstration of opening the object. This demonstration consists of a set of poses of the robot’s wrist link during the opening motion. These poses are then interpolated with splines to generate an end-effector motion. This agent’s aim is to follow this end-effector motion to complete the object interaction. We evaluate whether the motion was successful based on whether the marker on the object changed its position after the execution of the subpolicy. For doors, we do not lock the spring-loaded door latch, as the robot is not strong enough to press the handle down.

The only exception to this is the door between the kitchen and living room. We found that the robot was unable to localize itself during the opening motion, as the moving door occupies the overwhelming majority of the robot’s LiDAR measurements. As a result, it was not possible to follow the opening demonstration whenever the marker moved out of the robot’s view. Instead, we use a simpler pushing motion from the inside and abstain from opening it in the other direction.

### S.4. SCENE GRAPH CONSTRUCTION

In the following, we provide additional insights into the construction of the navigational Voronoi graph:

- In order to increase robustness, we first inflate  $\mathcal{B}_t$  using an Euclidean signed distance field (ESDF) formulation before computing the Voronoi graph. After inflation, we overwrite free space coordinates as given in  $\mathcal{F}_t$  as zero.
- 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.
- 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  $G_V$  to obtain fewer navigational nodes.
- The object-to-room assignment relies on the following optimization objective:

$$n^R = \arg \min_{n_o \in G_V^R} \text{path}(n_o, n_{v_p}) + d(o, n_o)^\lambda + d(v_p, n_{v_p}), \quad (1)$$

where  $n_o, n_{v_p} \in G_V^R$  are the closest Voronoi nodes to the object  $o$  as well as the camera viewpoint position  $v_p$ , respectively. The  $\text{path}(\cdot)$  denotes a shortest-path on  $G_V^R$ .

In order to provide more insights regarding the open-set room classification scheme used, we show the utilized prompt in Fig. S.2.

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. S.2. Room Classification Prompt: based on the objects and room clusters of the scene graph, an LLM performs open-vocabulary classification.

TABLE S.1  
MAPPING OF DISTANCES TO NATURAL LANGUAGE.

$\leq$ Distance	Encoding
3.0	very close
10.0	near
20.0	far
$\infty$	distant

### S.5. LANGUAGE ENCODING

**GPT Versions:** Unless otherwise noted, we use *gpt-4-1106-preview* for the high-level reasoning and *gpt-3.5-turbo-1106* for the simpler room classification task [17].

**Distances:** We encode distance to natural language based on a discrete mapping, following the principle of [15]. The mapping is reported in Tab. S.1.

**History Re-alignment:** 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 call.

### S.6. BASELINES

In this section, we provide additional details regarding the baselines:

**ESC-Interactable:** Co-occurrences are based on similarities of a finetuned DeBERTa-v3 language model [22], 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.

We adapt *HIMOS* [9] 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.

The *Unstructured LLM* baseline receives the same instructions “remember” notes as our approach. The full JSON-formatted prompt of this baseline is depicted in Fig. S.3. We

system: You are a robot in an unexplored house. Your task is to find an oven.

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: Scene Graph: {nodes: [{frontier_points: {((-3.9749999999999996, 7.725), <FRONTIER_CLASSIFICATION.LEADING_OUT: 1>), ((2.85, 6.825), <FRONTIER_CLASSIFICATION.LEADING_OUT: 1>), ((-4.125, 5.325), <FRONTIER_CLASSIFICATION.LEADING_OUT: 1>), ((-3.3, 7.574999999999999), <FRONTIER_CLASSIFICATION.WITHIN: 0>), ((0.375, 9.15), <FRONTIER_CLASSIFICATION.LEADING_OUT: 1>), ((-0.75, 8.1), <FRONTIER_CLASSIFICATION.WITHIN: 0>), ((-1.5, 6.75), <FRONTIER_CLASSIFICATION.WITHIN: 0>)}, closed_doors: set(), id: living room, node_type: room}, {id: room-1}, {id: room-4}, {frontier_points: set(), closed_doors: set(), id: bedroom, node_type: room}, {id: room-0}, {id: room-2}, {id: room-3}, {frontier_points: set(), closed_doors: set(), id: bathroom-1, node_type: room}, {frontier_points: set(), closed_doors: set(), id: other room, node_type: room}, {frontier_points: set(), closed_doors: set(), id: bathroom-2, node_type: room}, {id: room-5}, {frontier_points: {((-2.3249999999999997, 1.95), <FRONTIER_CLASSIFICATION.LEADING_OUT: 1>)}, closed_doors: set(), id: bathroom-3, node_type: room}, {id: alarm_10037, room: bedroom, node_type: object}, {id: straight_chair_9, room: bedroom, node_type: object}, {id: console_table_10, room: bedroom, node_type: object}, {id: table_1011, room: bedroom, node_type: object}, {id: table_lamp_12, room: bedroom, node_type: object}, {id: swivel_chair_13, room: bedroom, node_type: object}, {id: table_lamp_14, room: bedroom, node_type: object}, {id: table_1015, room: bedroom, node_type: object}, {id: table_lamp_18, room: bedroom, node_type: object}, {id: pot_plant_19, room: bedroom, node_type: object}, {id: console_table_20, room: bedroom, node_type: object}, {id: table_1021, room: living room, node_type: object}, {id: notebook_10041, room: bedroom, node_type: object}, {id: armchair_23, room: living room, node_type: object}, {id: sofa_24, room: living room, node_type: object}, {id: soap_10046, room: bathroom-2, node_type: object}, {id: console_table_25, room: living room, node_type: object}, {id: bottom_cabinet_26, room: living room, state: closed, node_type: object}, {id: floor_lamp_27, room: living room, node_type: object}, {id: coffee_table_28, room: living room, node_type: object}, {id: bed_49, room: bedroom, node_type: object}, {id: bottom_cabinet_54, room: bedroom, state: closed, node_type: object}, {id: bottom_cabinet_55, room: bedroom, state: closed, node_type: object}, {id: carpet_56, room: bedroom, node_type: object}, {id: sofa_57, room: bedroom, node_type: object}, {id: picture_58, room: living room, node_type: object}, {id: sink_59, room: bathroom-1, node_type: object}, {id: pen_10040, room: bedroom, node_type: object}, {id: toilet_60, room: bathroom-1, node_type: object}, {id: mirror_61, room: bathroom-1, node_type: object}, {id: shower_62, room: bathroom-1, node_type: object}, {id: picture_63, room: bathroom-1, node_type: object}, {id: towel_rack_64, room: living room, node_type: object}, {id: towel_rack_65, room: bathroom-2, node_type: object}, {id: toilet_66, room: bathroom-3, node_type: object}, {id: sink_67, room: bathroom-3, node_type: object}, {id: powder_bottle_10043, room: bathroom-1, node_type: object}, {id: paper_towel_10045, room: bathroom-3, node_type: object}, {id: mirror_68, room: bathroom-3, node_type: object}, {id: picture_69, room: bathroom-3, node_type: object}, {id: grandfather_clock_70, room: living room, node_type: object}, {id: mirror_71, room: living room, node_type: object}, {id: picture_72, room: living room, node_type: object}, {id: picture_73, room: living room, node_type: object}, {id: carpet_74, room: living room, node_type: object}, {id: picture_76, room: living room, node_type: object}, {id: window_115, room: bedroom, state: closed, node_type: object}, {id: window_116, room: bedroom, state: closed, node_type: object}, {id: window_117, room: bedroom, state: closed, node_type: object}, {id: window_118, room: bedroom, state: closed, node_type: object}, {id: window_121, room: living room, state: closed, node_type: object}, {id: window_122, room: bathroom-1, state: closed, node_type: object}, {id: window_123, room: bedroom, state: closed, node_type: object}, {id: window_125, room: bedroom, state: closed, node_type: object}, {id: window_127, room: living room, state: closed, node_type: object}, {id: alarm_10007, room: bedroom, node_type: object}, {id: floor_lamp_10008, room: bedroom, node_type: object}, {id: pencil_box_10014, room: bedroom, node_type: object}, {id: light_bulb_10016, room: bedroom, node_type: object}, {id: plate_10017, room: bedroom, node_type: object}, {id: pencil_box_10018, room: living room, node_type: object}, {id: pencil_box_10020, room: living room, node_type: object}, {id: pencil_box_10021, room: living room, node_type: object}, {id: document_10024, room: living room, node_type: object}, {location: living room, id: robot}}, links: [bathroom-1 - shower_62, bathroom-3 - sink_67, bedroom - pot_plant_19, living room - coffee_table_28, living room - picture_72, bedroom - swivel_chair_13, bathroom-2 - soap_10046, bedroom - table_1017, living room - mirror_71, room-1 - bathroom-1, living room - grandfather_clock_70, room-4 - bathroom-3, living room - picture_73, bathroom-2 - room-5, bedroom - alarm_10037, bedroom - floor_lamp_10008, bathroom-2 - towel_rack_65, bedroom - straight_chair_9, living room - picture_76, bedroom - room-0 - carpet_56, living room - carpet_74, bedroom - sofa_57, bedroom - table_lamp_14, bathroom-3 - mirror_68, bedroom - table_1015, bedroom - window_117, living room - pencil_box_10018, bedroom - bottom_cabinet_55, living room - robot, bedroom - pencil_box_10014, living room - window_127, bedroom - window_116, bedroom - console_table_10, living room - sofa_24, living room - towel_rack_64, bedroom - window_118, bathroom-1 - powder_bottle_10043, room-1 - other room, bedroom - light_bulb_10016, bedroom - console_table_20, bathroom-3 - toilet_66, bedroom - plate_10017, bedroom - table_1011, living room - picture_58, bedroom - notebook_10041, bedroom - bed_49, bedroom - table_lamp_18, living room - table_1021, bedroom - bottom_cabinet_54, living room - bottom_cabinet_26, bathroom-1 - picture_63, bedroom - table_1016, living room - armchair_23, living room - document_10024, bedroom - window_115, bathroom-1 - toilet_60, bathroom-1 - sink_59, living room - room-4, bedroom - window_123, living room - pencil_box_10021, living room - console_table_25, bedroom - window_125, bathroom-1 - mirror_61, living room - pencil_box_10020, living room - window_121, bathroom-3 - paper_towel_10045, bedroom - alarm_10007, living room - room-1, bedroom - room-2, living room - floor_lamp_27, bathroom-1 - window_122, bedroom - pen_10040, room-0 - bathroom-2, bedroom - table_lamp_12, bedroom - room-3, bathroom-3 - picture_69]],\nHistory: go_to_and_open(living room, room-1) - invalid argument, go_to_and_open(living room, room-2) - invalid argument, go_to_and_open(living room, room-3) - invalid argument, go_to_and_open(living room, room-0) - invalid argument, go_to_and_open(other room-2, door) - invalid argument, go_to_and_open(bathroom-2, door) - invalid argument, explore(bedroom) - success, explore(living room) - success, explore(living room) - success.\nWhat 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.\nRemember:\n1. Respond with a function call\n2. You can only use the objects and rooms that you have already found. Object names have to match the description exactly.\n3. You can only explore rooms that are listed as having unexplored space\n4. 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.\n5. If some actions failed repeatedly, they may not be possible.
```

assistant: Analysis:

The oven is typically located in the kitchen, which is not explicitly listed in the scene graph. However, there are rooms that have not been explored yet, as indicated by the presence of frontier points. Since the oven has not been found in the known rooms and the priority is to open doors over exploring rooms when the object cannot be found in a known room, the next step should be to open any unopened doors that may lead to new rooms, potentially the kitchen.

The scene graph does not list any closed doors in the known rooms, but there are frontier points in the living room and bathroom-3 that suggest there are areas leading out or not fully explored. Since the living room has already been explored multiple times without finding an oven, and the bathroom is an unlikely location for an oven, the next logical step is to explore the frontier point in bathroom-3, which may lead to a new room.

Reasoning:

Exploring the frontier point in bathroom-3 could reveal a new room, possibly the kitchen where the oven is likely to be found. Since the oven has not been located in any of the currently known rooms and there are no closed doors to open, exploring the frontier point is the next best action to potentially discover the kitchen.\n

Command:

```
...explore(bathroom-3)...
```

Fig. S.3. Prompt of the Unstructured-LLM baseline. It receives the same instructions and notes as our approach, but instead of a structured knowledge encoding, it receives a pure JSON encoding of the scene-graph.

find that the much less structured and longer prompt leads to more frequent invalid actions or hallucinations (cf. Tab. I), such as trying to open objects that do not exist or are already opened. If stuck for repeated steps, this can result in failed episodes. A second source of failures are wrong terminations, in which the LLM calls *done()* while it has not found the correct object of interest.

## S.7. ADDITIONAL RESULTS

### A. Scene Graph Accuracy

Our proposed room separation scheme relies on separating Voronoi graphs at door positions. Thus, it is prone to under-segmentation whenever faced with open room layouts or, e.g., *missing* doors to hallways. Downstream decision-making is performed at every high-level policy step, which requires

TABLE S.2  
SCENE GRAPH EVALUATION

Environment	Open-Set	Purity $G_V^R$	Room Category Accuracy
iGibson	×	0.609	0.276
Real-World	✓	-	0.901

accurate dynamically created scene graphs. Most importantly, the obtained representation varies a lot throughout the course of the exploration of the environment. In the following, all reported numbers denote the average across all high-level steps until an episode is finished to reflect this.

Tab. S.2 provides additional insights into the accuracy of the utilized scene graph representations. First, we evaluate the purity of the separated Voronoi graph components by evaluating the underlying ground-truth room label per node in



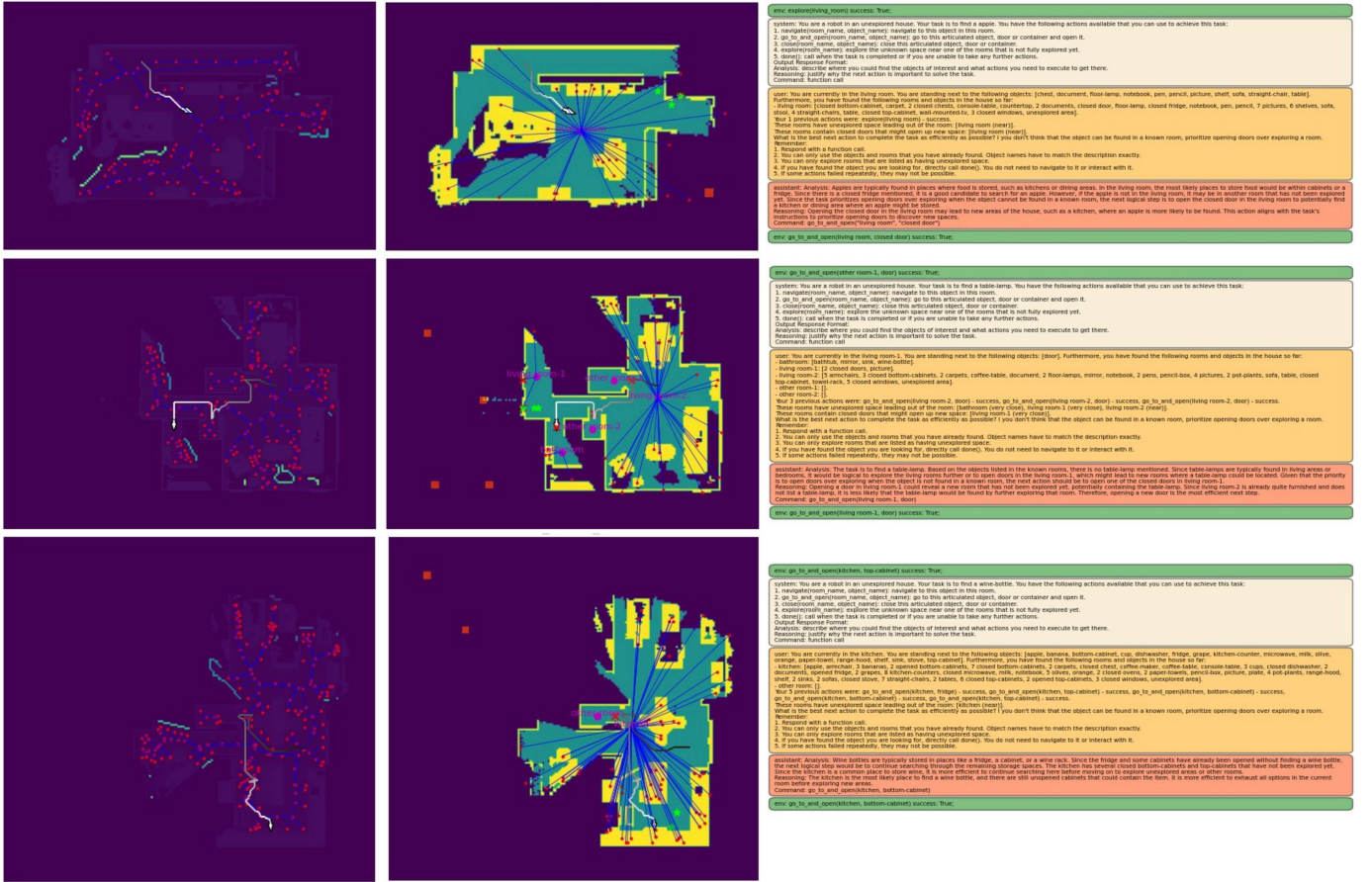


Fig. S.4. Examples of the reasoning of Voronoi graph and identified frontiers (left), scene graph (middle), MoMa-LLM (left). The green environment feedback is not provided to the LLM. Black-white: agent trace, red crosses: closed doors, red rectangles: undiscovered target objects, green stars: next selected navigation / interaction points. The bottom figure illustrates a suboptimal room clustering, which results in a very large room. We find the model to be robust against these clusterings.

the simulation. The purity measures to which degree clusters only contain a single class. In our case, each separated Voronoi graph (representing a room) takes the role of a cluster. Next, we quantify how many ground-truth rooms are covered per the predicted Voronoi graph. Thus, the graph purity describes the degree of room under-segmentation apparent in the scene and thus measures how well door-wise Voronoi graph separation performs.

We observe an average purity throughout the exploration of 0.609 over 10 episodes across each of the iGibson test scenes. Nonetheless, open room layouts still pose a considerable hurdle, which would require additional perception inputs. More general approaches to segment rooms in the iGibson environments have shown inferior results. This is due to a large number of long and narrow corridors, which are hard to segment using classical morphological segmentation algorithms. Nonetheless, 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 false object-room assignments (*through* walls) to a minimum. We show multiple resulting Voronoi graphs in Fig. S.4. In addition to the graph purity, we also evaluate the semantic room categories predicted

by GPT-3.5. Even though MoMa-LLM uses open-set room categories as described in the main manuscript, we evaluate the performance on closed-set room category prediction to allow comparison. To do so, we provided GPT-3.5 with all room categories provided in the iGibson dataset in order to pick the most suitable given the metrics that are averaged over 10 episodes per scene as well as across all high-level policy steps per episode. We compare the predicted room category of each Voronoi node with the underlying ground-truth room layout maps. Following this, we arrive at an average predicted room category accuracy of 0.276. The number is further affected by open room layouts as mentioned above.

To provide a comparison, we also evaluated the predicted room categories in an open-set manner on the real-world map shown in Fig. S.1. Across the 10 trials executed in the real world as given in Tab. II, we follow the same evaluation protocol and obtain an average room category accuracy of 90.1% as detailed in Tab. S.2. Human-level assessment allows evaluating errors such as *entryway* instead of *hallway* positively, which drastically increases the metrics. Nonetheless, the real-world map is less complex in terms of topology and object distribution

TABLE S.3  
RESULTS ON THE FUZZY SEARCH TASKS IN THE REAL WORLD.

Task	Success	Reasoning
I am hungry. Find me something for breakfast.	✓	Found cereals, toast, coffee. Then opened fridge to look for milk or eggs. Found milk in the fridge and decided these are sufficient items.
Find me something to wash my hands.	✓	Searched kitchen for a sink. When not finding one, searches for storage room or bathroom until it finds the sink in the bathroom.
I feel sleepy. Find me something to wake up.	✓	The teapot is associated with tea that contains caffeine and can help someone wake up.
Find things to set the kitchen table.	✓	Explored kitchen, opening cabinets. Found knife, glasses, plates. Continued to look for forks or spoons. When not finding them, called done. (No forks or spoons existed).
Find me the book in the living room.	✓	Explored until finding living room, then opened cabinet looking for book, found it inside.
Pour me a glass of milk.	✓	Finds milk in fridge, glass on table. Navigates between the two, assuming to transport the last object. Then terminates, reasoning that it has found and navigated to both.
Turn on the oven.	✓	Finds the oven and calls done(): "Turning on oven is implied as completion of the task".
What's the time?	✓	Finds the clock and calls done(): "The clock is the object that will provide the time".

Notes: Top: fuzzy search queries. Bottom: infeasible task queries. Success in these tasks is evaluated by human judgment as a reasonable response. The reasoning has been qualitatively paraphrased for brevity.

compared to the iGibson environments, which feature, e.g., rooms with no objects contained.

### B. Reasoning

This section provides additional examples of the scene encodings and the model's behavior. Tab. S.3 lists the queries for the fuzzy search task, together with a summary of the model's behavior and reasoning. As discussed in Sec. IV, we find that the model is well capable of handling these more complex tasks, finding relevant objects and correctly evaluating the task completion.

Fig. S.4 shows additional examples of the scene representations and the model reasoning. It depicts the Voronoi graph and frontiers to unexplored areas (left), the BEV-map together with the constructed scene-graph (middle) and the input prompt and answers of the LLM (right). Additional video material with full prompt reasoning is provided on the project website.

### REFERENCES

- [1] N. Reimers and I. 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)*, Nov. 2019, pp. 3982–3992.
- [2] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2020, pp. 10 740–10 749.
- [3] F. Schmalstieg, D. Honerkamp, T. Welschhold, and A. Valada, "Learning hierarchical interactive multi-object search for mobile manipulation," *IEEE Robotics and Automation Letters*, 2023.
- [4] D. Honerkamp, T. Welschhold, and A. Valada, "N<sup>2</sup>m<sup>2</sup>: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *IEEE Transactions on Robotics*, 2023.
- [5] G. Chalkatzaki, A. Younes, D. Nandha, A. T. Le, L. F. Ribeiro, and I. Gurevych, "Learning to reason over scene graphs: a case study of finetuning gpt-2 into a robot language model for grounded task planning," *Frontiers in Robotics and AI*, vol. 10, 2023.