

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

- Supplementary Material -

Daniel Honerkamp^{1*}, Martin Büchner^{1*}, Fabien Despinoy², Tim Welscheshold¹, Abhinav Valada¹

In this supplementary material, we provide additional details on 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 also demonstrated in the video material.

S.1. SIMULATION ENVIRONMENT

A. Modifications

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*.

B. Task Generation

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 ≤ 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.

C. Perception

The robot in simulation is equipped with a differential drive and an RGB-D camera with a resolution of 256×256 pixels and a vertical field of view of 120° . 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 objects with a volume below 0.01 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 detect the *floor* and *carpet* categories as free space.

D. 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 sequentially to each waypoint the resulting path from the A*-planner (matching the map resolution of 0.075 m), collecting the observations along the whole path. 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], [7] and execute articulated object manipulations as “magic actions”. These actions directly set the joint values of the object to their

*These authors contributed equally.

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

²Toyota Motor Europe (TME)

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

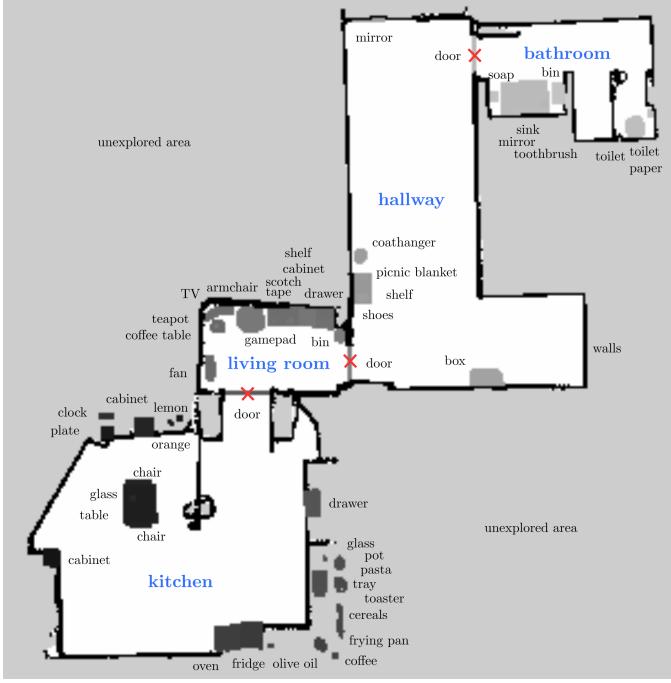


Fig. S.1. Map of the real-world environment. Different intensities represent different object instances in the environment. Room annotations are for illustration and not used by our model. Object labels denote approximate object positions.

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

S.2. REAL-WORLD ENVIRONMENT

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 methods. The target categories given to the agents were *shoes*, *milk* (inside fridge), *knife* (inside kitchen drawer), *book* (inside living room cabinet), *toilet paper*, *toaster*, *tea* (inside kitchen cabinet), *pencil* (inside living room drawer), *soap* and *lemon*.

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^2M^2 manipulation policies [37]. 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.3. HIERACHICAL SCENE GRAPH STRUCTURE

In order to provide a concise overview a complete picture of the employed scene graph hierarchy we provide a detailed overview of it in Tab. S.1.

In addition, we outline the construction of the navigational Voronoi graph used for navigation and associating objects to regions in the following. The Generalized Voronoi Diagram (GVD) [5] comprises two-equidistant faces that each represent the set of points equidistant to two obstacles C_i and C_j . Each point in this set is closer to C_i and C_j than any other obstacle:

$$\mathcal{F}_{ij} = \{x \in \mathbb{R}^m : 0 \leq d_i(x) = d_j(x) \forall k \neq i, j, \nabla d_i(x) \neq \nabla d_j(x)\}. \quad (1)$$

The union of all two-equidistant faces generated by the obstacle positions defined by \mathcal{B}_t is called the 2-Voronoi set \mathcal{F}^2 or the two-dimensional GVD of the space of obstacles $C_i \in \mathcal{B}_t$:

$$\mathcal{F}^2 = \bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^n \mathcal{F}_{ij}. \quad (2)$$

The set of points contained in \mathcal{F}^2 constitutes the initial set of edges of the generalized Voronoi graph (GVG). We extract the corresponding nodes by computing the 3-Voronoi sets, which constitute the joints of the GVD:

$$\mathcal{F}^3 = \bigcup_{i=1}^{n-2} \bigcup_{j=i+1}^{n-1} \bigcup_{k=j+1}^n \mathcal{F}_{ijk}. \quad (3)$$

Given this, the generalized Voronoi graph GVG = $\{\mathcal{F}^2, \mathcal{F}^3\}$ undergoes sparsification by eliminating edges of degree 2 to form \mathcal{G}_V .

TABLE S.1
SCENE GRAPH STRUCTURE

Component	Description
$\mathcal{G}_V = (\mathcal{V}, \mathcal{E})$	Navigational Voronoi graph that undergoes sparsification and covers traversable map areas
\downarrow	
$\mathcal{G}_V^R = (\mathcal{V}', \mathcal{E}')$ with $\mathcal{V}' = \{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_k\}$ where $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ for all $i, j \in \{1, \dots, k\}$ with $i \neq j$	Deletion of edges with sufficient edge integrals computed over $\rho_{\mathcal{N}}$ (Eq. 2) results in pair-wise edge-disjoint sets \mathcal{V}_i with each set covering a room r_i .
\downarrow	
$\mathcal{G}_S = (\mathcal{V}_O \cup \mathcal{V}_R, \mathcal{E}_{OR} \cup \mathcal{E}_{RR})$	We define the actual scene graph \mathcal{G}_S over all object nodes \mathcal{V}_O and room supernodes \mathcal{V}_R . Each room $r_i \in \mathcal{V}_R$ holds edges its corresponding Voronoi nodes $\mathcal{V}_i \in \mathcal{V}'$. Each room $r_i \in \mathcal{V}_R$ (and its Voronoi nodes \mathcal{V}_i) undergoes semantic classification (Fig. 3). The edges of \mathcal{E}_{OR} connect objects $o \in \mathcal{G}_S$ with their associated room $r_i \in \mathcal{V}_R$ via Eq. 3. \mathcal{E}_{RR} are the edges connecting neighboring rooms $r_i \in \mathcal{V}_R$ to one another.

TABLE S.2
MAPPING OF DISTANCES TO NATURAL LANGUAGE.

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

S.4. LANGUAGE ENCODING

We encode distance to natural language based on a discrete mapping, following the principle of [17]. We bin the distance to the object, then apply the mapping reported in Tab. S.2. This results in a consistent relative language encoding.

S.5. BASELINES

The *Unstructured LLM* baseline receives the same instructions and "remember" notes as our approach. The full JSON-formatted prompt of this baseline is depicted in Fig. S.2. We 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.6. ADDITIONAL RESULTS

A. Hierarchical Scene Graph

1) *Metrics*: In the following, we list the metrics used for evaluating the accuracy of the scene graph.

Room Segmentation Precision / Recall: In order to compare our method of room segmentation against the approach used by Hydra [9] we make use of the same metrics they evaluated:

$$P_t = \frac{1}{|R_e|} \sum_{r_e \in R_e} \max_{r_g \in R_g} \frac{|r_g \cap r_e|}{|r_e|}, \quad (4)$$

$$R_t = \frac{1}{|R_g|} \sum_{r_g \in R_g} \max_{r_e \in R_e} \frac{|r_e \cap r_g|}{|r_g|}, \quad (5)$$

where R_e is the set of estimated rooms and R_g is the set of ground-truth rooms. The cardinality of a set is given by $|\cdot|$. Each room r_e or r_g is defined by its set of covered pixels on a 2D grid. In order to reflect instabilities throughout exploring the environment we report the means and standard deviations of the precision and recall, respectively:

$$\bar{P} = \frac{1}{T} \sum_{t=1}^T P_t \quad \sigma_P = \sqrt{\frac{\sum_{t=1}^T (P_t - \bar{P})^2}{T}} \quad (6)$$

$$\bar{R} = \frac{1}{T} \sum_{t=1}^T R_t \quad \sigma_R = \sqrt{\frac{\sum_{t=1}^T (R_t - \bar{R})^2}{T}} \quad (7)$$

The metrics are evaluated on a dense 2D grid with a resolution of 0.05 m, which is increased compared to the normal resolution of 0.075 m used in all other evaluations. This is done to account for thin walls contained in iGibson. In order to evaluate the separated Voronoi graphs covering distinct rooms on a dense manifold we employ room-specific wavefronts initialized at each node of the separated Voronoi graph bounded by the extracted obstacles.

Room Segmentation Purity: In addition to the dense evaluation outlined above we also evaluate the purity of each of the generated components of the separated Voronoi graphs. Being a criterion used for measuring clustering quality [8] it penalizes the effect of covering multiple ground truth rooms per classified room.

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j| \quad (8)$$

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_classification_leading_out_1: ((-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, 7.547999999999999), <FRONTIER_CLASSIFICATION.WITHIN: 0>, <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: bedroom, 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: set(), closed_doors: set(), id: other_room, node_type: room}, {frontier_points: set(), 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_19, room: bedroom, node_type: object}, {id: table_lamp_14, room: bedroom, node_type: object}, {id: table_1011, room: bedroom, node_type: object}, {id: table_lamp_12, room: bedroom, node_type: object}, {id: table_1016, room: bedroom, node_type: object}, {id: table_1017, 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: 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: living_room, 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: sink_65, room: bathroom-3, node_type: object}, {id: powder_bottle_10043, room: bathroom-1, node_type: object}, {id: picture_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: picture_70, room: living_room, 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: 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_10087, room: bedroom, node_type: object}, {id: floor_lamp_10088, 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_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_67, 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, bedroom - 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_122, 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}], History: go_to_and_open(living_room, room-0) - 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.

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 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.2. 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.

where $\Omega = \{\omega_1, \omega_2, \dots, \omega_K\}$ is the set of components of \mathcal{G}_V^R and $\mathbb{C} = \{c_1, c_2, \dots, c_J\}$ is the set of ground-truth rooms. Each ω_k holds the predicted room types of all nodes of the respective component. In our case, the purity measures the extent to which a set of Voronoi nodes covers a room that contains a single class. Thus, the graph purity describes the degree of room under-segmentation apparent in the scene and thus measures how well the door-wise Voronoi graph separation performs. Similar to the room segmentation precision and recall we average the purity throughout exploring the environment.

2) Room Segmentation and Classification: 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. We visualize the scene graphs produced by Hydra as well as MoMa-LLM in Fig. S.3. Regarding the approach of Hydra, we observe that maps containing a many corridors with

TABLE S.3
CLOSED-SET ROOM CATEGORY EVALUATION

Environment	Open-Set	Purity $G_{\mathcal{V}}^R$	Room Category Accuracy
iGibson	✗	0.615	0.276
Real-World	✓	-	0.901

similar diameters produce either a very large number or a very small number of graph components when applying a range of various obstacle dilation values. As Hydra selects the final environment partitioning based on the median of the number of graph components obtained through various obstacle dilation values, it is faced with a bi-modal distribution. This ultimately renders a median-based selection of the segmentation difficult.

In addition to evaluation in Tab. II, we observe an average purity for MoMa-LLM of 0.615 throughout the exploration over

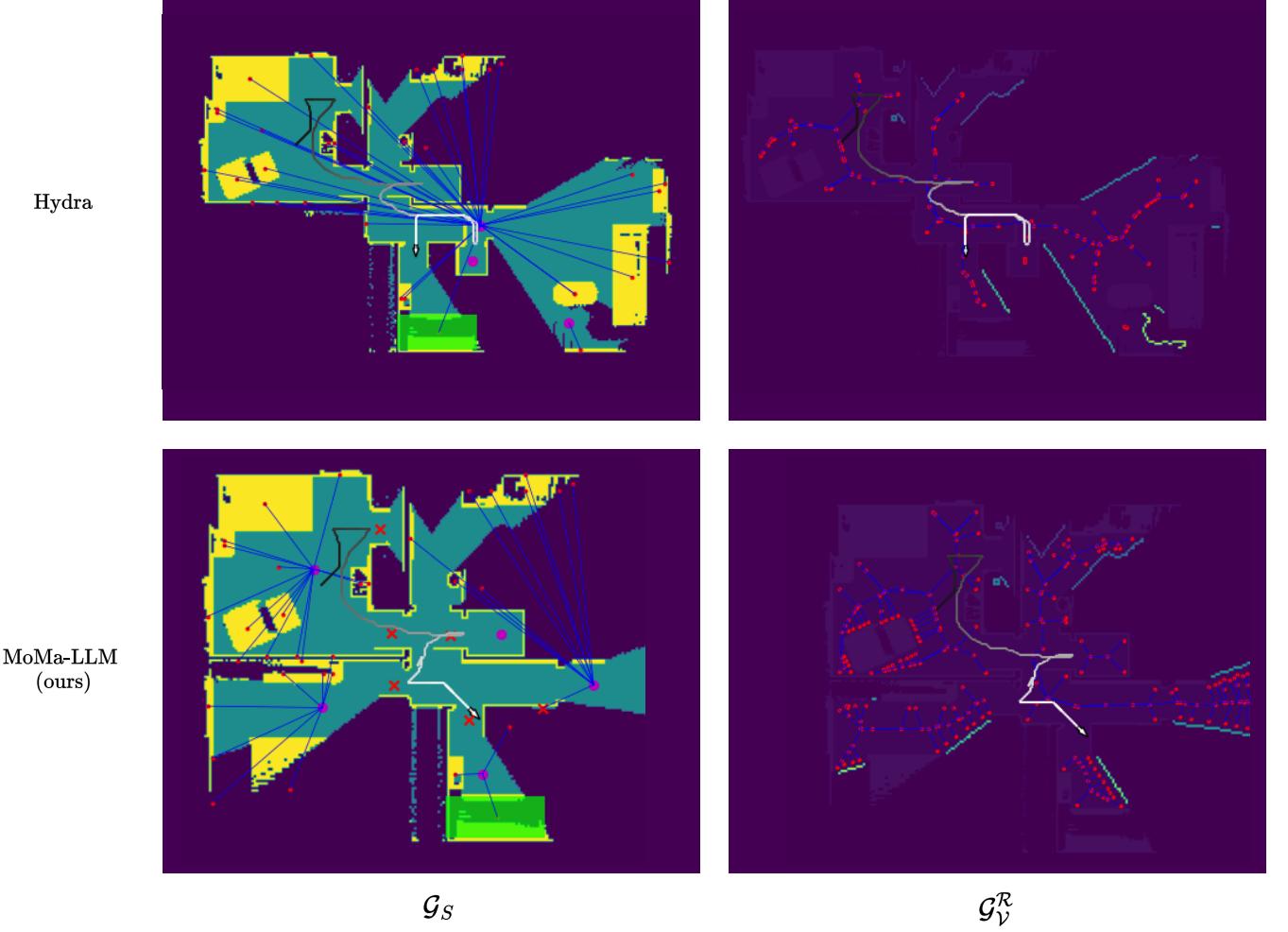


Fig. S.3. Visualization of the scene graphs produced by Hydra compared to MoMa-LLM. The left column represents the abstracted scene graphs \mathcal{G}_S while the right column depicts the separated Voronoi graphs \mathcal{G}_V^R as defined in Tab. S.1. The red crosses represent extracted door positions of MoMa-LLM. The depicted scene is *Merom_1_int* contained in the iGibson test scenes.

10 episodes across all of the iGibson test scenes. Compared with that, the room segmentation approach introduced by Hydra [9] reaches a purity of 0.562 as listed in Tab. II. While both Hydra and MoMa-LLM tend to under-segment the given room layout, Hydra is specifically affected by narrow constrictions induced by obstacle placements and non-varying door widths, which creates a significant number of isolated graphs covering small corridors that do not represent full ground-truth regions. In general, we infer that long and narrow corridors as well as cluttered scenes are challenging to segment using classical morphological segmentation algorithms (see Fig. S.3).

We found our downstream 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 time-wise averaged room segmentation results reported in Tab. II, we evaluate the semantic room

categories predicted by GPT-3.5. in Tab. S.3. Even though MoMa-LLM normally uses open-set room categories, we evaluate the performance on a closed-set of room categories to report reproducible results. To do so, we provided GPT-3.5 with all room categories contained in the iGibson dataset with the task to pick the most suitable given the objects assigned to each particular Voronoi component representing a room. Similar to the segmentation evaluation, we report numbers 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 27.6%. This number is largely affected by open room layouts as mentioned above.

In addition to the closed-set evaluation, 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. III, we follow the same evaluation protocol and obtain an average room category accuracy of 90.1% as listed in Tab. S.3. Human-level assessment

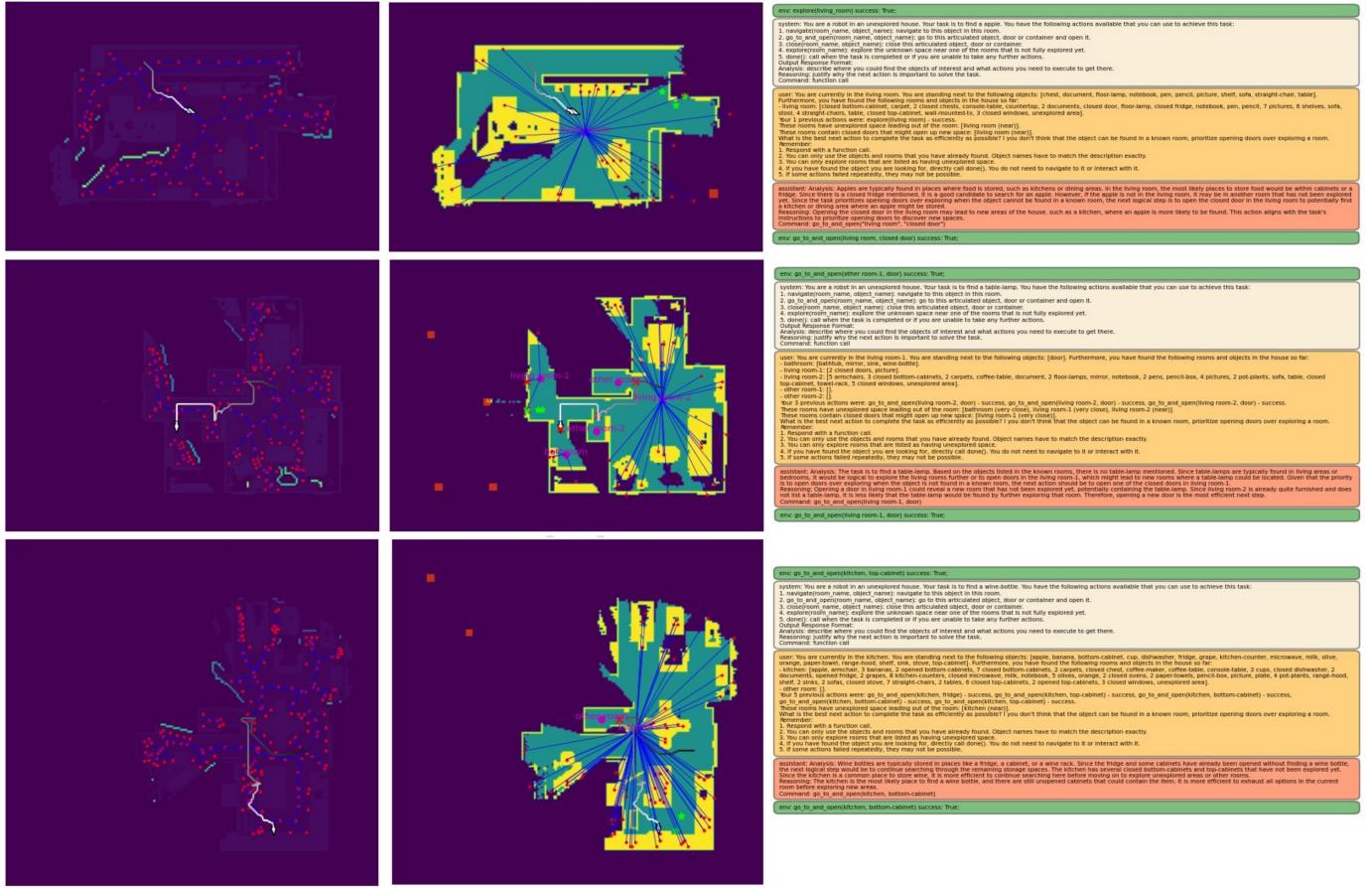


Fig. S.4. Examples of the reasoning of Voronoi graph and identified frontiers (left), scene graph (middle), MoMa-LLM (left). The green environment feedback is not provided to the LLM. Black-white: agent trace, red crosses: closed doors, red rectangles: undiscovered target objects, green stars: next selected 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.

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 its topology and object distribution compared to the iGibson environments, which feature, e.g., rooms with no objects contained.

B. Reasoning

Tab. S.4 show the full set of fuzzy search queries (top) and infeasible queries (bottom) that were evaluated in Sec. V, together with the language model’s reasoning in response to these tasks.

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 shown on the project website.

C. Deployment with Full Perception Pipeline

While we focus on evaluation with ground-truth perception in the main work to be able to focus on the representation and

decision making components, this section provides details on the requirements of full deployment and guidance to facilitate the deployment.

Our approach requires (i) an RGB-D sensor (ii) localization and mapping (iii) semantic segmentation and (iv) grasp pose detection. Modern RGB-D SLAM approaches such as RTAB-Map [9] can provide (i) and (ii). As our approach supports open-vocabulary representation and reasoning, it enables deployment with any semantic segmentation model, irrespective of its supported classes. This can be closed vocabulary methods [10], [11] such as Mask R-CNN [12] or newer, transformer-based methods [13], [14]. A further range of methods can provide object detections and bounding boxes for a given list of open-vocabulary query categories [15]–[18]. The best model should be based on the use-case, available compute, required object categories, and accuracy. Chen et al. [28] provide a possible reference implementation: they deploy an RTAB-node for localization and mapping. This can be extended to semantic labels through an additional RTAB-node that listens to the semantic masks, to then fuse the resulting point clouds. Finally (iv), for handle detection and grasp-pose detection, [19] achieve accurate results with a retrained YOLO model on a public handle-specific dataset.

TABLE S.4
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. Find things to set the kitchen table.	✓ ✓	The teapot is associated with tea that contains caffeine and can help someone wake up. 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. What's the time?	✓ ✓	Finds the oven and calls done(): "Turning on oven is implied as completion of the task". 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.

TABLE S.5
EVALUATION OF THE AVERAGE RUNTIME PER COMPONENT.

Component	Time per step (s)	Total time (s)	Share (%)
Real-World	24.1	457	100
	13.2	250	55
	7.0	139	30
	13.9	111	24
	10.9	207	45
Simulation	High-level Reasoning	11.1	91.1
	Scene Graph Construction	3.0	31.5
	Room Classification	0.6	5.0
	LLM Reasoning	7.5	64.6

Notes: Time per step is the average time for one high-level step or subpolicy call. Total time is the average total time per episode. Real-world numbers are averaged over three episodes of the fuzzy-search experiments. Simulation numbers are averaged over 175 episodes in the iGibson simulator.

D. Runtime Analysis

Tab. S.5 decomposes the runtime of the system into individual components. While the agent executes more navigation subpolicy calls (this includes driving to objects to open), each manipulation takes longer on average, resulting in similar total times spent in each. We also find that the high-level reasoning takes up a significant fraction of time. We extend this with data from the simulation experiments to be able to break it down into components. In particular the LLM queries for high-level reasoning take up the majority of this time. This demonstrates the importance of current work for compact and fast inference, which is currently receiving a lot of attention [20], as well as the importance of open-source models [21] that can be run locally instead of purely through an API.

While the current implementation of the scene graph is not optimized for speed, and the graph is fully recomputed at each time step instead of only updating areas that received new observations, we find its overall time impact to be reasonable, as it is only required at high-level reasoning steps.

E. Extended Future Work Discussion

In this work, we introduce scene graphs as an efficient and scalable representation for high-level, language model

based reasoning, by encoding the scene graphs in a structured language representation and the incorporation of knowledge about distances and unexplored areas.

In future work, we aim to relax the assumptions about accurate perception, fully constructing scene graphs from noisy sensor inputs, as e.g. done in Hydra [9] or the direct incorporation of open-vocabulary representations [8]. While we currently encode distances and spatial arrangements as adjectives and room-object relations, full maps provide much more dense spatial and geometric information. Research for more direct incorporation of such information, e.g. through vision-language models is very promising. Furthermore, more holistic approaches to incorporate spatial and semantic details in room clustering and classifications will be important to address non-standard layouts and designs. Lastly, methods to incorporate more detailed visual feedback for the identification of object states and failure reasons are an important are to increase robustness and success over long tasks.

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 Comp. Vision and Pattern Recognition*, 2020, pp. 10740–10749.
- [3] F. Schmalstieg, D. Honerkamp, T. Welschkehold, and A. Valada, "Learning hierarchical interactive multi-object search for mobile manipulation," *IEEE Robotics and Automation Letters*, 2023.
- [4] D. Honerkamp, T. Welschkehold, and A. Valada, "N²m²: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *IEEE Transactions on Robotics*, 2023.
- [5] H. Choset and J. Burdick, "Sensor based planning. i. the generalized voronoi graph," in *Proceedings of 1995 IEEE international conference on robotics and automation*, vol. 2. IEEE, 1995, pp. 1649–1655.
- [6] G. Chalvatzaki, A. Younes, D. Nandha, A. T. Le, L. F. Ribeiro, and I. Gurevych, "Learning to reason over scene graphs: a case study of finetuning gpt-2 into a robot language model for grounded task planning," *Frontiers in Robotics and AI*, vol. 10, 2023.
- [7] 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.
- [8] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press, 2008.

- [9] M. Labb   and F. 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*, vol. 36, no. 2, pp. 416–446, 2019.
- [10] N. V  disch, K. Petek, W. Burgard, and A. Valada, "Codeps: Online continual learning for depth estimation and panoptic segmentation," in *Robotics: Science and Systems*, 2023.
- [11] N. Gosala, K. Petek, P. L. Drews-Jr, W. Burgard, and A. 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*, 2023, pp. 14 901–14 910.
- [12] K. He, G. Gkioxari, P. Doll  r, and R. Girshick, "Mask r-cnn," in *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, 2017, pp. 2961–2969.
- [13] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, 2022.
- [14] M. K  ppeler, K. Petek, N. V  disch, W. Burgard, and A. Valada, "Few-shot panoptic segmentation with foundation models," *Int. Conf. on Robotics & Automation*, 2023.
- [15] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," in *Int. Conf. on Learn. Repr.*, 2022.
- [16] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen *et al.*, "Simple open-vocabulary object detection," in *Proc. of the Europ. Conf. on Computer Vision*. Springer, 2022, pp. 728–755.
- [17] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling open-vocabulary object detection," *Proc. of the Conf. on Neural Information Processing Systems*, vol. 36, 2024.
- [18] W. Kuo, Y. Cui, X. Gu, A. Piergiovanni, and A. Angelova, "Open-vocabulary object detection upon frozen vision and language models," in *Int. Conf. on Learn. Repr.*, 2023.
- [19] M. Arduengo, C. Torras, and L. Sentis, "Robust and adaptive door operation with a mobile robot," *Intelligent Service Robotics*, vol. 14, no. 3, pp. 409–425, 2021.
- [20] A. Chavan, R. Magazine, S. Kushwaha, M. Debbah, and D. Gupta, "Faster and lighter llms: A survey on current challenges and way forward," *Int. Conf. on Artificial Intelligence*, 2024.
- [21] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.
- [22] A. Werby, C. Huang, M. B  chner, A. Valada, and W. Burgard, "Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation," *Robotics: Science and Systems*, 2024.