MORE: Mobile Manipulation Rearrangement Through Grounded Language Reasoning

Anonymous Authors

***Abstract*— Autonomous long-horizon mobile manipulation encompasses a multitude of challenges, including scene dynamics, unexplored areas, and error recovery. Recent works have leveraged foundation models for scene-level robotic reasoning and planning. However, the performance of these methods degrades when dealing with a large number of objects and large-scale environments. To address these limitations, we propose MORE, a novel approach for enhancing the capabilities of language models to solve zero-shot mobile manipulation planning for rearrangement tasks. MORE leverages scene graphs to represent environments, incorporates instance differentiation, and introduces an active filtering scheme that extracts task-relevant subgraphs of object and region instances. These steps yield a bounded planning problem, effectively mitigating hallucinations and improving reliability. Additionally, we introduce several enhancements that enable planning across both indoor and outdoor environments. We evaluate MORE on 81 diverse rearrangement tasks from the BEHAVIOR-1K benchmark, where it becomes the first approach to successfully solve a significant share of the benchmark, outperforming recent foundation model-based approaches. Furthermore, we demonstrate the capabilities of our approach in several complex real-world tasks, mimicking everyday activities. We make the code publicly available at `redacted-for-review`.**

I. Introduction

Recent studies have achieved notable advances in the completion of long-horizon robotic tasks in large-scale environments [1]–[4]. This progress was largely fueled by recent breakthroughs in scene comprehension and the integration of foundation models. At the same time, evaluations are still limited to known environments [1], [3], interactive search tasks [2], [5], or a series of hand-crafted tasks in specific real-world settings that lack reproducibility by the community [1], [3], [6]–[8].

Previous research conducted within single-room environments has demonstrated the capability to accomplish a broad spectrum of tasks [9], [10]. Nonetheless, in the context of mobile robotic manipulation, the scope of environments can expand to encompass entire apartments and outdoor areas, leading to an exponential increase in the number of objects and possible interactions, leading to exploding planning times [11] or hallucinations [1]. Furthermore, assumptions of a priori known scene layouts and object locations do not account for environment changes or flexible deployment. These partial scene observability and unexplored areas complicate planning by requiring reasoning about the unknown.

In this work, we focus on generalizing to rearrangement tasks with a mobile manipulator in a zero-shot manner, tackling both large-scale and unexplored environments spanning across indoor and outdoor spaces. We scale previous work in natural language-based interactive object search [2] and

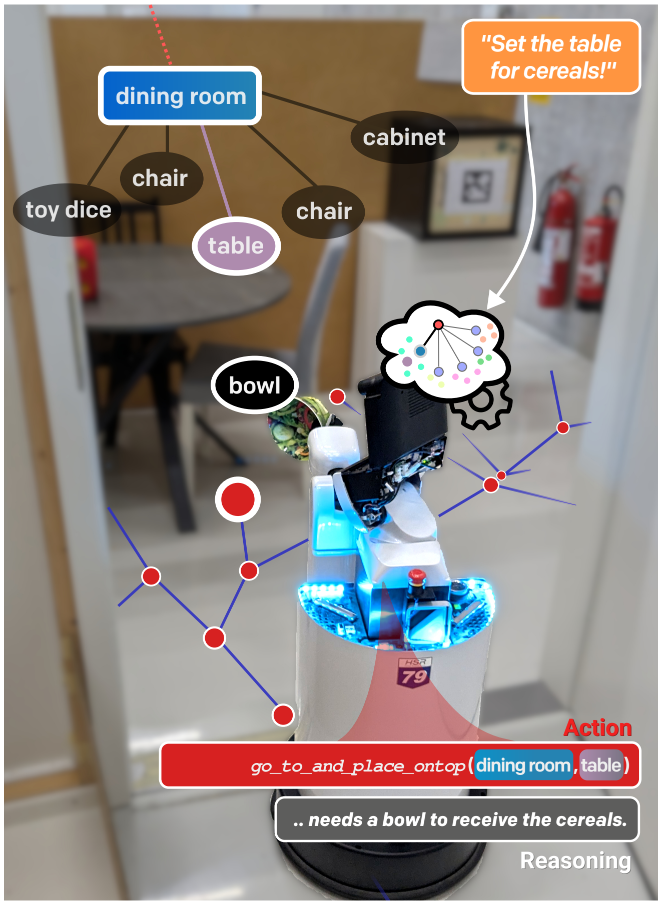

"Set the table for cereals!"
dining room
cabinet
chair
toy dice
chair
table
bowl
Action
go_to_and_place_ontop(dining room, table)
.. needs a bowl to receive the cereals.
Reasoning

Fig. 1. We present MORE, an efficient model for the task of rearrangement through mobile manipulation. We utilize 3D scene graphs as a logical scene representation manifold that is filtered to obtain task-relevant subgraphs.

thus expand the range of application to general everyday activities and introduce the first method capable of solving a substantial part of the BEHAVIOR-1K benchmark [12]. We refine the benchmark by providing fully defined language task descriptions and implementing computationally efficient evaluations, thereby facilitating reproducible assessments across an extensive set of tasks. Subsequently, we present a methodology that can operate effectively in unknown environments, both indoor and outdoor, and extends the reasoning to a large set of subpolicies. Our approach introduces novel solutions to integrate object and instance information and to identify the subscene pertinent to the current task, thereby maintaining the planning problem within manageable bounds and significantly mitigating hallucinations and distractions.

In our experiments, we conduct a comparative analysis of previous approaches based on Large Language Model

(LLM) and Vision-Language Model (VLM) planners. Our experiments reveal substantial benefits from incorporating explicit memory, such as scene graphs, and employing our filtering approach to streamline the planning phase by minimizing the search space.

Our key contributions can be summarized as follows:

- A novel scene representation that encompasses environments spanning across both indoor and outdoor areas.
- An efficient LLM-based planning framework that identifies the task-relevant subscenes and can execute instance-level actions in large-scale environments.
- The extension of the BEHAVIOR-1K benchmark with fully-specified language task descriptions, facilitating reproducible assessments of everyday activities.
- The first approach to complete a significant subset of BEHAVIOR-1K tasks, containing 81 daily activities together with real-world evaluation of representative tasks in a multi-room apartment.

II. Related Work

3D Scene Graphs and Spatial AI: Robotic reasoning and planning struggle to scale effectively in large-scale environments when utilizing dense semantic representations [13], [14]. Instead, recent works break down large-scale indoor or outdoor scenes into hierarchical 3D scene graphs covering distinct concepts at various levels of granularity. [15]–[18]. While Hydra [15] focuses on real-time execution, Concept-Graphs [19] and HOV-SG [16] investigate the incorporation of open-vocabulary vision-language features for language-grounded robot navigation. OrionNav [20] and MoMa-LLM [2] further introduce frontier-based exploration stacks for dynamically populating the scene graph based on language-guided high-level policy calls. Orthogonally, KESU [21] proposed constructing a knowledge base by employing vision-language foundation models to enhance spatial environment understanding. Clio [22] proposes to cluster task-agnostic object primitives based on task instructions to reduce scene complexity. Related to the problem of scene granularity, Search3D [23] proposes an additional hierarchical level below the object primitives by breaking down class-agnostic object proposals into granular segments representing partial objects. While still in its early stages, another branch of studies deals with dynamic updates of scene graphs through various input signals. Most works solely focus on the aspect of incorporating novel observations [2], [20], whereas MM-3DSGU [24] is the first general approach to also consider human interactions, robot actions, or language inputs [24]. Compared to previous works, we propose a scene graph approach that spans both indoor and outdoor domains under a single combined representation and allows space partitioning through Voronoi decomposition.

Vision-Language Models for Planning: Foundation models have shown strong reasoning capabilities that provide accurate representations. Language models have been utilized to formulate a planning domain and establish a goal for a symbolic planner to execute planning within [25], [26]. Coupled with scene graphs, LLMs have been able to solve long-horizon tasks [1], [2], [27]. Recently, VLMs have enabled finer-grained decision-making based on annotated images [7], [28]. Where COME relies on global semantically annotated object maps to enable longer-horizon reasoning [6], OK-Robot [29] only executes open-vocabulary pick-and-drop tasks. BUMBLE further extends to additional skills and includes visual memory [3]. Furthermore, fully end-to-end models have succeeded in performing tasks such as unloading and folding laundry. However, these models concentrate on confined environments [6], [7], [28]–[30] that do not require robust memory mechanisms, rely on a priori known scenes [1], [3] or involve a restricted range of interactions related to search and exploration [2], [27].

In large environments, methods to reduce the size of the planning problem have been proposed by predicting the importance of objects through graph neural networks [31]. Liu *et al.* [26] query an LLM to prune a fully known scene graph given a task description. Ray *et al.* [32] identify rules for removing symbols from a task and motion planning (TAMP) navigation problem. In contrast to these methods, our work concentrates on formulating plans within unknown environments, thus necessitating an analysis of the significance of nodes in the context of exploration. EmbodiedRAG [33] enables an LLM to envision task-relevant objects, subsequently formulating plans only with those objects, which correspond to the a priori knowledge.

Embodied AI Benchmarks: In the last years, numerous simulators and benchmarks have been introduced, including large and navigable scenes [34]. Recently, they have been extended to partially interactive scenes for rearrangement tasks, without doors or articulated objects [35], [36], and often restricted to confined room-sized scenes [9], [10]. Where iGibson [37] provides 15 fully interactive apartment scenes, ProcThor procedurally generates large scenes, nonetheless with limited complexity [38]. A number of recent works have shown capabilities on real-world hand-crafted tasks, complex to replicate and use for benchmarking [1], [3], [6]–[8]. In an effort to enable evaluations that are both accessible and replicable, we have chosen BEHAVIOR-1K [12], a dataset that augments the scenes from iGibson to encompass 50 varied daily task scenarios with near-photorealistic visualization. It offers an open and challenging testbed for long-horizon task planning in large environments that require strong capabilities of understanding and memory.

III. Approach

A. Problem Statement: Embodied Reasoning

This work investigates embodied reasoning in large and unexplored environments. The agent is situated in an unknown environment and given a task description g in natural language together with its proprioceptive sensor observations, consisting of a posed RGB-D frame I_t and the current robot state s_{robot}. For computational simplicity, we assume accurate semantic perception by relying on ground truth semantics. To complete the task, the agent has to explore its surroundings,

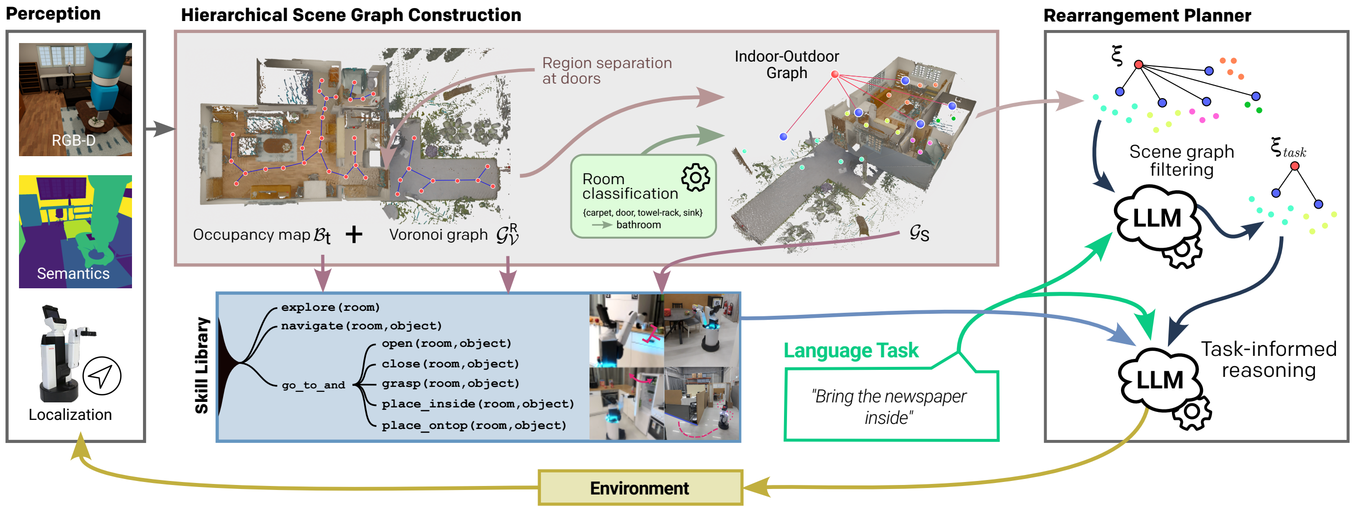


Fig. 2. Overview of MORE. Starting in an unexplored environment, we continuously construct a hierarchical scene graph of the environment based on RGB-D data and semantic segmentation. We first build a dense occupancy map and then extract a navigational Voronoi graph. This graph is separated at room borders based on door locations, then clustered and segmented into rooms and outdoor regions. The resulting scene graph is converted to natural language descriptions. To generate a bounded planning problem, we first filter the observed objects by task relevance and then use an LLM as a task planner in the resulting subgraph. The LLM orchestrates navigation, manipulation, and exploration subpolicies, which in turn result in an updated scene representation.

identify relevant objects given the task, and interact with the environment until all goal conditions are satisfied.

Although recent studies are centered on manually defined tasks executed under real-world conditions [1], [3], [6]–[8], these tasks frequently exhibit a high degree of specificity unique to each research endeavor and are often challenging to reproduce. This specificity has resulted in numerous approaches that are difficult to compare. For this reason, we focus on one of the broadest benchmarks for everyday tasks, BEHAVIOR-1K [12]. While this benchmark has been established for a considerable period, it remains unsolved. Its public availability and broad scope enable reproducible, large-scale evaluation of general task solving capabilities.

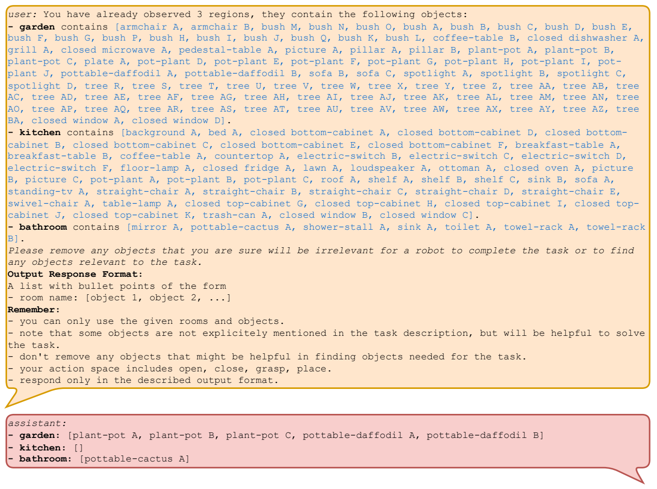
The BEHAVIOR-1K benchmark consists of 1,000 activities in over 50 scenes across both indoor and outdoor areas [12]. Each task is defined by its associated objects as well as initial and final conditions in the predicate logic language BDDL [39]. The tasks are then procedurally initialized in the OmniGibson environment, built on top of the NVIDIA Isaac simulator. The environment samples the required task objects and places them in a randomly chosen scene such that they fulfill the initial task conditions. Following successful initialization, the agent, embodied as a Fetch robot, is then randomly placed in the scene. We carefully analyzed both the task environments and definitions and provide the following benchmark modifications for practicality and reproducibility. We verified that all scenarios are computationally feasible in a reasonable amount of time.

We categorized the tasks into two groups: currently solvable and nearly unsolvable. Specifically, we isolate a subset of 163 tasks that necessitate interactions involving articulated objects and, in the broadest interpretation, entail rearrangements with *inside* and *onto* state transitions. We further exclude any task wherein the initialization process is unsuccessful. The final set includes 81 tasks such as `packing picnic food into car` or `buy dog food` across 17 unique scenes. Given that the original task descriptions frequently lack unique specificity regarding the desired task outcomes, we provide natural language descriptions derived from the BDDL goal conditions for the complete set of tasks. For instance, the natural language description for the first mentioned task is: `Put all the food in the bags and load the bags into the car`. The enumeration of the tasks, along with their corresponding natural language descriptions, is accessible in Supplementary Sec. S.1.

Nevertheless, conducting evaluations across this extensive set of tasks is exceedingly time-consuming, owing to their prolonged duration and the constrained simulation speed within large environments. We address this problem by introducing significantly accelerated subpolicies. These policies follow the concept of "magic actions", removing the need for physical simulations and their associated steps. Although this methodology disables the precise execution of arm movements, it enables a reasonable simulation pace to evaluate high-level reasoning capabilities across a broader set of tasks compared to previous works. We make these subpolicies publicly available with our code. The implementation of these magic actions is detailed in Supplementary Sec. S.2.

B. *MORE Model*

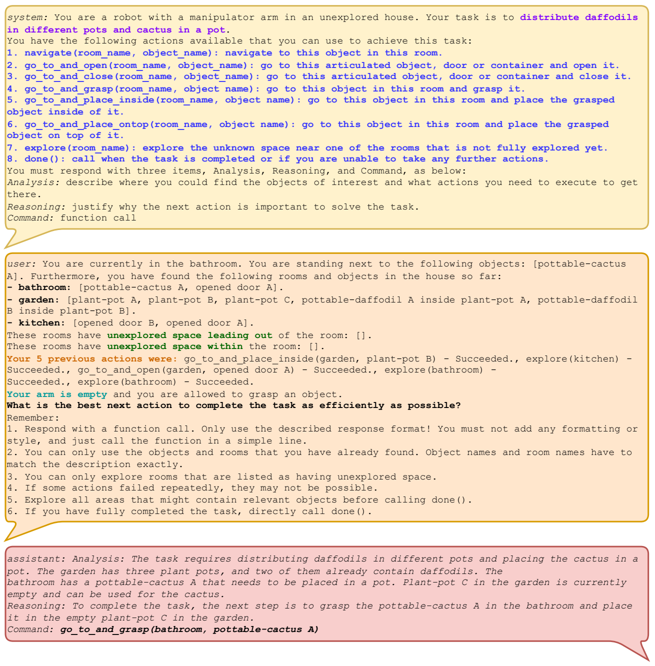
In this work, we scale previous work on language model-based planning via scene graphs to reason and act in large environments while extending it to general tasks [2]. Our model consists of three main components: a hierarchical scene graph representation, a high-level language-based planner, and subpolicies that control the robot. An overview is shown in Fig. 2.



```
user: You have already observed 3 regions, they contain the following objects:
- garden contains [armchair A, armchair B, bush M, bush N, bush O, bush A, bush B, bush C, bush D, bush E,
bush F, bush G, bush P, bush H, bush I, bush J, bush Q, bush K, bush L, coffee-table B, closed dishwasher A,
grill A, closed microwave A, pedestal-table A, picture A, pillar A, pillar B, plant-pot A, plant-pot B,
plant-pot C, plate A, pot-plant D, pot-plant E, pot-plant F, pot-plant G, pot-plant H, pot-plant I, pot-
plant J, pottable-daffodil A, pottable-daffodil B, sofa B, sofa C, spotlight A, spotlight B, spotlight C,
spotlight D, tree R, tree S, tree T, tree U, tree V, tree W, tree X, tree Y, tree Z, tree AA, tree AB, tree
AC, tree AD, tree AE, tree AF, tree AG, tree AH, tree AI, tree AJ, tree AK, tree AL, tree AM, tree AN, tree
AO, tree AP, tree AQ, tree AR, tree AS, tree AT, tree AU, tree AV, tree AW, tree AX, tree AY, tree AZ, tree
BA, closed window A, closed window D].
- kitchen contains [background A, bed A, closed bottom-cabinet A, closed bottom-cabinet D, closed bottom-
cabinet B, closed bottom-cabinet C, closed bottom-cabinet E, closed bottom-cabinet F, breakfast-table A,
breakfast-table B, coffee-table A, countertop A, electric-switch B, electric-switch C, electric-switch D,
electric-switch F, floor-lamp A, closed fridge A, lawn A, loudspeaker A, ottoman A, closed oven A, picture
B, picture C, pot-plant A, pot-plant B, pot-plant C, roof A, shelf A, shelf B, shelf C, sink B, sofa A,
standing-tv A, straight-chair A, straight-chair B, straight-chair C, straight-chair D, straight-chair E,
swivel-chair A, table-lamp A, closed top-cabinet G, closed top-cabinet H, closed top-cabinet I, closed top-
cabinet J, closed top-cabinet K, trash-can A, closed window B, closed window C].
- bathroom contains [mirror A, pottable-cactus A, shower-stall A, sink A, toilet A, towel-rack A, towel-rack
B].
Please remove any objects that you are sure will be irrelevant for a robot to complete the task or to find
any objects relevant to the task.
Output Response Format:
A list with bullet points of the form
- room name: [object 1, object 2, ...]
Remember:
- you can only use the given rooms and objects.
- note that some objects are not explicitely mentioned in the task description, but will be helpful to solve
the task.
- don't remove any objects that might be helpful in finding objects needed for the task.
- your action space includes open, close, grasp, place.
- respond only in the described output format.
```

```
assistant:
- garden: [plant-pot A, plant-pot B, plant-pot C, pottable-daffodil A, pottable-daffodil B]
- kitchen: []
- bathroom: [pottable-cactus A]
```

Fig. 3. Language-based scene graph filtering: We employ a filtering prompt that structurally represents all regions and respective objects of the unfiltered scene ξ. Next, we employ an LLM in order to identify which objects are task-relevant. Those are returned, yielding a task-informed sub-graph ξ_{task}.



```
system: You are a robot with a manipulator arm in an unexplored house. Your task is to distribute daffodils
in different pots and cactus in a pot.
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. go_to_and_close(room_name, object_name): go to this articulated object, door or container and close it.
4. go_to_and_grasp(room_name, object name): go to this object in this room and grasp it.
5. go_to_and_place_inside(room_name, object name): go to this object in this room and place the grasped
object inside of it.
6. go_to_and_place_ontop(room_name, object name): go to this object in this room and place the grasped
object on top of it.
7. explore(room_name): explore the unknown space near one of the rooms that is not fully explored yet.
8. done(): call when the task is completed or if you are unable to take any further actions.
You must respond with three items, Analysis, Reasoning, and Command, as below:
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 bathroom. You are standing next to the following objects: [pottable-cactus
A]. Furthermore, you have found the following rooms and objects in the house so far:
- bathroom: [pottable-cactus A, opened door A].
- garden: [plant-pot A, plant-pot B, plant-pot C, pottable-daffodil A inside plant-pot A, pottable-daffodil
B inside plant-pot B].
- kitchen: [opened door B, opened door A].
These rooms have unexplored space leading out of the room: [].
These rooms have unexplored space within the room: [].
Your 5 previous actions were: go_to_and_place_inside(garden, plant-pot B) - Succeeded., explore(kitchen) -
Succeeded., go_to_and_open(garden, opened door A) - Succeeded., explore(bathroom) -
Succeeded., explore(bathroom) - Succeeded.
Your arm is empty and you are allowed to grasp an object.
What is the best next action to complete the task as efficiently as possible?
Remember:
1. Respond with a function call. Only use the described response format! You must not add any formatting or
style, and just call the function in a simple line.
2. You can only use the objects and rooms that you have already found. Object names and room names have to
match the description exactly.
3. You can only explore rooms that are listed as having unexplored space.
4. If some actions failed repeatedly, they may not be possible.
5. Explore all areas that might contain relevant objects before calling done().
6. If you have fully completed the task, directly call done().
```

```
assistant: Analysis: The task requires distributing daffodils in different pots and placing the cactus in a
pot. The garden has three plant pots, and two of them already contain daffodils. The
bathroom has a pottable-cactus A that needs to be placed in a pot. Plant-pot C in the garden is currently
empty and can be used for the cactus.
Reasoning: To complete the task, the next step is to grasp the pottable-cactus A in the bathroom and place
it in the empty plant-pot C in the garden.
Command: go_to_and_grasp(bathroom, pottable-cactus A)
```

Fig. 4. Language-based reasoning: Assuming a filtered sub graph ξ_{task}, we provide the LLM with the task description, a skill API, and the filtered set of regions and objects. Based on the high-level action chosen by the LLM, we orchestrate the next policy steps to be executed in order to fulfill the task at hand.

1) Scene Graph: Given the requirement to act in combined indoor-outdoor scenes, we extend previous work to larger scenes. Similarly to MoMa-LLM [2], we employ Voronoi graphs to cover the navigational space. Although Voronoi graphs allow for efficient space partitioning, they quickly grow to an impractical number of nodes, which is exacerbated in the considered large-scale environments. In general, each morphological irregularity generates a substantial quantity of Voronoi nodes that do not contribute to functional utility. Therefore, we introduce a novel sparsification scheme operating on the navigational Voronoi graph, whereby we identify and extract nodes of degree two that fall within a specified distance of one another. We then replace the original edges with a single coarser edge. The resulting algorithm scales linearly in the number of vertices and is provided in Supplementary Sec. S.3. While MoMa-LLM [2] assigns all objects to the largest connected component of the Voronoi graph, this is no longer sufficient for areas that extend to the outdoors. A specific challenge is the observability of different regions of the scene through unnavigable windows. We instead assign objects directly to their closest component and only then reduce the observed scene graph to the component connected to the robot's current position. This eliminates unreachable regions until a path is discovered by opening a door, for instance. Given these modifications, the proposed navigational graph can efficiently represent unified indoor-outdoor scenes that feature highly disparate node counts. In addition, we extend the room separation mechanism to an adaptive kernel width based on the observed door bounding boxes, thereby facilitating more robust segmentation of regions. Subsequently, we adhere to MoMa-LLM's strategy to assign objects to the separated Voronoi graph, based on both the viewpoint they are observed from and the language-model based room classification of each region.

2) High-level planning: Given our perceived state of the scene, our aim is to take the action with the highest likelihood of progressing the task. We introduce a novel scene graph filtering mechanism and extend LLM-based reasoning to instances and attributes.

Scene representation: We extend prior research to encode the scene graph, including potential unobserved regions, into a structured language representation [2]. Whereas previous research concentrated predominantly on class-specific object search, our objective is to generalize to a broader range of tasks. As a result, we represent the task relevance of object attributes and properties, even for multiple target object instances, in our scene graph representation. Inferred object states or attributes, accessible through recent VLM methods [16], are incorporated into their related description as supplementary attributes. However, given that objects can possess a large number of attributes, this can lead to a large increase of the scene description. We tackle this through the filtering described below. As a task may require us to interact with a specific instance, we first uniquify each instance by assigning it an alphabetical ID and provide all instances to the LLM. Note that this principle is interchangeable with using specific properties as instance identifiers. However, alphabetical IDs enable fair comparisons across different methods. The resulting scene description is depicted in Fig. 3. We specify each region and its corresponding objects including IDs and ask the LLM to remove object instances that are not relevant given the task as detailed in the following.

Scene graph filtering: Planning in the full environment is a highly challenging task, as considering all possible objects and interactions leads to either exploding planning times [11] or hallucinations [33]. For common everyday tasks, only a small subset of objects and areas in the scene are actually task relevant. Therefore, we hypothesize that there exists a filtering function f that maps from the full scene ξ to the relevant subscene ξ_{task}, which is simpler to attain than the full task planning. Its aim is to ignore distracting scene entities. In contrast to previous work [26], [31], [32], we face two additional challenges: we are acting in unexplored environments, making not just task-specific objects relevant, but also objects that may inform us about the unexplored parts of the scene, and we are considering tasks that might require multiple instances of objects or instances with specific attributes.

We propose an LLM-based scene filtering for partially observed scenes. The LLM receives a structured natural language description of the scene together with the task instructions. It is then prompted to remove any objects that will be irrelevant for a robot to completing the task, or in finding any of the objects relevant for the task, and to respond with a formatted list of rooms and objects. An example is shown in Fig. 3.

Task planning: Given the identified subscene ξ_{task}, we conduct MPC-style task planning, outputting the next subpolicy to execute before replanning with the newest scene observation. We instantiate this as another large language model call $a = l(\xi_{task}, s_{robot}, g, h)$, where g is the task instruction, h is an action history, including potential environment feedback from the subpolicy execution. We extract two parts of information from the robot state s_{robot}: the room it is currently in, and whether it is holding an object in its gripper. The action history h consists of the previous five actions as well as all previously failed actions. The environment feedback c consists of a binary indicator whether a subpolicy succeeded or failed, as well as easily obtainable feedback such as the gripper being full or requiring an articulated object to be opened before placing another object inside. While more information would be available in simulation, it is often not realistic to obtain such details in the real world. An example prompt is shown in Fig. 4.

We parameterize object-centric subpolicies, consisting of `explore(room)`: explore the closest frontier within a specified room. `navigate(room, object)`: navigate to the vornoi node closest to this object via A*-planner in the constructed occupancy map. `go_to_and_{open, close, grasp, place_inside, place_ontop}(room, object)`: navigate to the closest Voronoi node, then execute a magic action to complete the interaction. `done()`: complete the episode and evaluate success. These subpolicies on one hand incorporate often implied precursor actions such as moving to the object, and on the other hand are parameterized by object and room names. We found this important to match the reasoning level of the LLM. The LLM is provided with a short description of each subpolicy.

An episode terminates if the model calls `done()`, exceeds 50 high level steps or continues to act for more than five high-level steps after all goal conditions are completed.

TABLE I
RESULTS ON 81 TASKS OF THE BEHAVIOR-1K BENCHMARK.

Model	SR [%]	TTC [%]	TP [%]	rTP [%]
BUMBLE [3]	11.3	15.0	35.2	39.7
BUMBLE [3] + filtering	15.7	20.5	43.7	48.8
MoMa-LLM [2]	19.7	32.1	<u>67.6</u>	<u>72.4</u>
MoMa-LLM [2] + spatial relations	<u>36.1</u>	<u>41.2</u>	64.6	67.4
MORE	**48.1**	**50.6**	**70.1**	**80.1**

The reported numbers are averaged across all evaluated tasks. We report the success rate (SR), the average task progression (TP), the relative task progression (rTP), and the TTC denoting the total task completion rate.

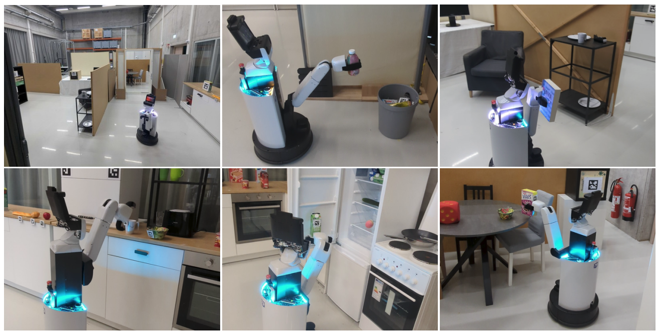


Fig. 5. Real-world experiments. From top left to bottom right: overview of the environment, throwing away a plastic bottle, fetching a book, grasping a bowl, taking milk out of the fridge, placing cereals on the dinner table.

IV. EXPERIMENTS

A. Experimental Setup

We evaluate our approach on the BEHAVIOR-1K benchmark [12] with a Fetch robot and magic actions as detailed in Sec. III-A. For every model tested, we use GPT-4o as a language model. For MoMa-LLM and MORE, we use the simpler model GPT-3.5 Turbo for room classifications.

Metrics: We compare the following statistics:
Success Rate (SR): share of tasks with all goal conditions completed and `done()` called correctly.. Note that this does not require the model call `done()` correctly after doing so.
Total Task Completion (TTC): share of tasks with all goal conditions completed, irresp. of calling `done()` correctly.
Task Progression (TP): average share of completed goal conditions of all goal conditions.
Relative task progression (rTP): share of completed goal conditions that include observed task objects. Removing a component of ambiguity from tasks such as "throw away *all* boxes" where it is unclear when to stop searching for further objects.

Baselines: We compare MORE against a range of different methods detailed in the following.
MoMa-LLM [2]: language-based planning through scene graphs tailored towards interactive object search, which we extend to our set of subpolicies.
MoMa-LLM [2] + *spatial relations*: the above model, equipped with an improved scene representation including spatial relations among observed objects, same as MORE.
BUMBLE [3]: a current VLM-based reasoning method that

TABLE II
REAL-WORLD EVALUATION

Task	Goal Conditions	Completed	Reasoning
Set the table for cereals.	Place bowl, cereals and milk from fridge on dinner table	3/3	Placed bowl and cereals on table; opened fridge to search for milk, found it, and placed it on table, then called done.
Clean up the empty bottles and cans.	Find two bottles and one can and put in the bin	3/3	Found and threw away all objects. Continued to search kitchen and living room. Then failed to call done, stopped manually when base controller disconnected.
Fetch a book and store it in the dinning room.	Find book and place in storage box or cabinet in dining room	1/1	Explored all rooms, found and grasped book, drove back to dining room and placed in storage box. Due to gripper opening too late book fell outside box. Then called done.

provides a visual short- and long-term memory together with the current, annotated view to a VLM model to select subpolicies. We adapt the model to have access to the same ground-truth semantic perception as MORE, replace its original skill set with our subpolicies and adapt the prompt descriptions accordingly. Note that BUBMLE does not employ a structured scene representation. Additional details are provided in Supplementary Sec. S.5.
BUMBLE [3] + filtering: the above model, extended with our scene filtering function, applied to the objects that get annotated in the current view.

B. Results

The results across the identified set of 81 tasks are reported in Tab. I. We find that BUMBLE is able to complete a significant share of tasks. However, the landmark-based memory does not scale efficiently to very large environments, as this requires a very large number of frames. Furthermore, it does not allow reacting to changes in the scenes easily (which would require updated images). While the VLM reasoning can take into account fine-grained spatial relationships, it struggles with cluttered scenes that result in heavily annotated images, which in turn leads to incorrect selections or hallucinations. Accordingly, we observe an increase in performance when incorporating our filtering in this model. Nevertheless, the other issues persist. One advantage of MoMa-LLM and MORE over BUMBLE is that they base action selection on a comprehensive representation of the entire scene, rather than relying solely on the current view and the histories. This approach allows MoMa-LLM and MORE to achieve superior results compared to BUMBLE. MoMa-LLM achieves a success rate of 19.7%, though completing all conditions in 32.1% of cases. We found that it struggles with the large number of objects and fails to differentiate instances. MoMa-LLM + spatial relations improves the differentiation of objects, enhancing performance and achieving a 36.1% success rate underlining the effectiveness of our scene representation.

MORE achieves a success rate of 48.1% and in over 50% of all tasks completes all goal conditions. The average share of overall completed task conditions is even higher at over 70%. Nonetheless, the model may face challenges in the following cases: (i) numerous constraints - although the model demonstrates high progress (rTP), it might overlook a few constraints and find a partial solution, (ii) task ambiguity - the task description might not provide the number of relevant objects, and the model struggles to determine when to stop searching, causing a gap between TP and rTP, (iii) simulator failures - spawning objects in unreachable areas, and getting stuck in dead spots due to opening articulated objects.
We conducted our experiments on a cluster consisting of 4 nodes, each equipped with 8 NVIDIA GeForce RTX 2080 Ti GPUs. We achieved an average completion time of 1 hour and 27 minutes per task.

C. Real-World Experiments

We construct an apartment consisting of three rooms, kitchen, living room, dining room, and deploy our approach on the Toyota HSR robot, with an omnidirectional base and 5-DoF arm. We follow previous work in assuming accurate semantic perception by revealing a pre-annotated semantic map based on the areas observed by the depth camera of the robot. We use the ROS Navstack for navigation and manipulate objects with the N^2M^2 mobile manipulation policy [40], [41] based on single end-effector demonstrations per object, relative to an AR marker pose. For place poses, we infer the closest point to the robot on the selected object to place down objects, then generate linear end-effector motions to that pose.

We evaluate our approach qualitatively on three challenging tasks. The results are reported in Tab. II and shown in Fig. 5 as well as the accompanying video. We find that our approach transfers well to the real world, completing all required goal conditions for all tasks, including complex interactions such as fetching the milk from the closed fridge. A small failure in opening the gripper led a book to fall outside the container. The largest issue was task ambiguity, leading to indeterminism whether the cleaning task is completed or if the agent should continue to search for further bottles. This reflects a general issue of abstract task definitions. Additionally, we find long LLM inference times to cause a strain on evaluations. While these are restricted to high-level reasoning at a very low control frequency, they still accumulate to a significant fraction of runtime.

V. CONCLUSION

We introduced MORE, a model capable of diverse rearrangement tasks across very large environments. We generate fully specified task descriptions for a large subset of the BEHAVIOR-1K tasks as well as a set of policies that allow computationally feasible evaluation. We construct scalable scene graphs across indoor/outdoor environments to subsequently identify the subscene and instance information directly related to the task. This enabled the efficient orchestration of object-centric subpolicies through a large language model. In extensive experiments, we find our approach to outperform both previous LLM-based models and VLM-based approaches and, for the first time, complete a significant subset of the BEHAVIOR-1K benchmark. Finally, we demonstrated that our approach successfully transfers to real-world scenarios. In future work, we aim to investigate the integration of VLM models for local fine-grained decision-making as well as further approaches for ambiguity resolution, either through human interaction, if available, or explicit reasoning about the desired task context.

References

[1] K. Rana, J. Haviland, S. Garg, J. 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.

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

[3] R. Shah, A. Yu, Y. Zhu, Y. Zhu, and R. Martín-Martín, "Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation," *Int. Conf. on Robotics & Automation*, 2025.

[4] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada, "Learning long-horizon robot exploration strategies for multi-object search in continuous action spaces," in *Robotics Research*, 2022, pp. 52–66.

[5] ——, "Learning hierarchical interactive multi-object search for mobile manipulation," *IEEE Robotics and Automation Letters*, 2023.

[6] P. Zhi, Z. Zhang, M. Han, Z. Zhang, Z. Li, Z. Jiao, B. Jia, and S. Huang, "Closed-loop open-vocabulary mobile manipulation with gpt-4v," *Int. Conf. on Robotics & Automation*, 2025.

[7] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess *et al.*, "Pivot: Iterative visual prompting elicits actionable knowledge for vlms," in *Int. Conf. on Mach. Learning*, 2024.

[8] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. 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.

[9] M. Shridhar, J. Thomason, D. Gordon *et al.*, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, 2020, pp. 10 740–10 749.

[10] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh *et al.*, "RoboCasa: Large-Scale Simulation of Household Tasks for Generalist Robots," in *Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[11] C. Agia, K. Jatavallabhula, M. Khodeir, O. Miksik, V. Vineet *et al.*, "Taskography: Evaluating robot task planning over large 3d scene graphs," in *Proc. of the Conf. on Rob. Learning*, 2022, pp. 46–58.

[12] C. Li, R. Zhang, J. Wong, C. Gokmen *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in *Proc. of the Conf. on Rob. Learning*. PMLR, 2023, pp. 80–93.

[13] K. Jatavallabhula, A. Kuwajerwala, Q. Gu, *et al.*, "Conceptfusion: Open-set multimodal 3d mapping," *Robotics: Science and Systems*, 2023.

[14] S. Prasanna, D. Honerkamp, K. Sirohi, T. Welschehold, W. Burgard, and A. Valada, "Perception matters: Enhancing embodied ai with uncertainty-aware semantic segmentation," *Int. Symposium of Robotics Research*, 2024.

[15] 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.

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

[17] E. Greve, M. Büchner, N. Vödisch, W. Burgard, and A. Valada, "Collaborative dynamic 3d scene graphs for automated driving," *Int. Conf. on Robotics & Automation*, 2024.

[18] J. Strader, N. Hughes, W. Chen, A. Speranzon, and L. Carlone, "Indoor and outdoor 3d scene graph generation via language-enabled spatial ontologies," *IEEE Robotics and Automation Letters*, 2024.

[19] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen *et al.*, "Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning," *Int. Conf. on Robotics & Automation*, 2024.

[20] V. N. Devarakonda, R. G. Goswami, A. U. Kaypak, N. Patel, R. Khorrambakht, P. Krishnamurthy, and F. Khorrami, "Orionnav: Online planning for robot autonomy with context-aware llm and open-vocabulary semantic scene graphs," *arXiv preprint arXiv:2410.06239*, 2024.

[21] F. Gao, J. Tang, J. Wang, S. Li, and J. Yu, "Enhancing scene understanding for vision-and-language navigation by knowledge awareness," *IEEE Robotics and Automation Letters*, 2024.

[22] D. Maggio, Y. Chang, N. Hughes, M. Trang, D. Griffith, C. Dougherty, E. Cristofalo, L. Schmid, and L. Carlone, "Clio: Real-time task-driven open-set 3d scene graphs," *IEEE Robotics and Automation Letters*, 2024.

[23] A. Takmaz, A. Delitzas, R. W. Sumner, F. Engelmann, J. Wald, and F. Tombari, "Search3d: Hierarchical open-vocabulary 3d segmentation," *IEEE Robotics and Automation Letters*, 2025.

[24] E. Olivastri, J. Francis, A. Pretto, N. Sünderhauf, and K. Rana, "Multi-modal 3d scene graph updater for shared and dynamic environments," *CoRL Workshop on Lifelong Learning for Home Robots*, 2024.

[25] T. Birr, C. Pohl, A. Younes, and T. Asfour, "Autogpt+ p: Affordance-based task planning with large language models," *arXiv preprint arXiv:2402.10778*, 2024.

[26] Y. Liu, L. Palmieri, S. Koch, I. Georgievski, and M. Aiello, "Delta: Decomposed efficient long-term robot task planning using large language models," *arXiv preprint arXiv:2404.03275*, 2024.

[27] A. Rajvanshi, K. Sikka, X. Lin, B. Lee, H. pang Chiu, and A. Velasquez, "Saynav: Grounding large language models for dynamic planning to navigation in new environments," in *Int. Conf. on Automated Planning and Scheduling*, 2024.

[28] Y. Wang, T.-H. Wang, J. Mao, M. Hagenow, and J. Shah, "Grounding language plans in demonstrations through counterfactual perturbations," in *The Twelfth International Conference on Learning Representations*, 2024.

[29] P. Liu, Y. Orru, J. Vakil, C. Paxton, N. M. M. Shafiullah, and L. Pinto, "Ok-robot: What really matters in integrating open-knowledge models for robotics," *arXiv preprint arXiv:2401.12202*, 2024.

[30] K. Black, N. Brown, D. Driess *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[31] T. Silver, R. Chitnis, A. Curtis, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Planning with learned object importance in large problem instances using graph neural networks," in *Proc. of the National Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 962–11 971.

[32] A. Ray, C. Bradley, L. Carlone, and N. Roy, "Task and motion planning in hierarchical 3d scene graphs," *Int. Symposium of Robotics Research*, 2024.

[33] M. Booker, G. Byrd, B. Kemp, A. Schmidt, and C. Rivera, "Embodiedrag: Dynamic 3d scene graph retrieval for efficient and scalable robot task planning," *arXiv preprint arXiv:2410.23968*, 2024.

[34] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub *et al.*, "Habitat: A Platform for Embodied AI Research," in *Int. Conf. on Computer Vision*, 2019.

[35] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai *et al.*, "Habitat 3.0: A co-habitat for humans, avatars, and robots," in *Int. Conf. on Learn. Repr.*, 2024.

[36] C. Gan, S. Zhou, J. Schwartz *et al.*, "The threedworld transport challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied ai," *Int. Conf. on Robotics & Automation*, 2022.

[37] C. Li, F. Xia, R. 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*, vol. 164, 08–11 Nov 2022, pp. 455–465.

[38] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs *et al.*, "Procthor: Large-scale embodied ai using procedural generation," *Proc. of the Conf. on Neural Information Processing Systems*, vol. 35, pp. 5982–5994, 2022.

[39] S. Srivastava, C. Li, M. Lingelbach *et al.*, "Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments," in *Proc. of the Conf. on Rob. Learning*. PMLR, 2022, pp. 477–490.

[40] D. Honerkamp, T. Welschehold, and A. Valada, "Learning kinematic feasibility for mobile manipulation through deep reinforcement learning," *IEEE Robotics and Automation Letters*, 2021.

[41] ——, "N^2m^2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *IEEE Transactions on Robotics*, 2023.

MORE: Mobile Manipulation Rearrangement Through Grounded Language Reasoning

- Supplementary Material -

Anonymous Authors

In this supplementary material, we provide additional details on the generated task descriptions and simulation setup. We furthermore provide details on the scene graph construction and the implementation details of the BUMBLE baseline. We also provide additional details on the real world subpolicies.

S.1. Task Subset

Tab. S.1 presents the tasks utilized to evaluate our method. The first column lists the activity names, each corresponding to a task from the BEHAVIOR-1K dataset. The second column provides our new task descriptions given to the agent. These descriptions ensure that all task relevant details are included for the goal conditions provided by the BEHAVIOR-1K benchmark. For example, if goal conditions are specific to a certain object instance in the scene, the task description entails this detail. For tasks requiring interaction with specific object instances in the scene, the variable is preceded by a dollar sign \$. During task initialization, variables preceded by a dollar sign \$ will be substituted with the corresponding object names.

S.2. Simulation Environment

A. Modifications

- We close exterior doors and filter them out of the scenes as they lead to empty space.
- We take non-physical steps as the robot size cannot be adjusted, and many collisions between the robot and objects in narrow areas are unavoidable and would otherwise not allow the robot to complete certain tasks.
- We skip scene `restaurant diner` as its area is divided into 2 disconnected regions, and the scene `school gym` because of floors with different heights.

B. Perception

The robot in simulation is equipped with an RGB-D camera with a resolution of 256×256 pixels. 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. We furthermore assume accurate detection of whether an articulated object is open or closed. Moreover, all inner objects within an articulated object become visible to the robot once the outer object is opened. We construct all maps at a resolution of 0.075 m and detect the floor, carpet, lawn and driveway categories as free space.

C. Execution

All trajectories 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 $7.5\,\mathrm{cm}$
- turn-left by up to $35\,\mathrm{rad}$
- turn-right by up to $35\,\mathrm{rad}$
- explore a region: the agent navigates to the closest frontier point of that region.
- open articulated object: if the arm is empty and the object is closed, the agent opens the object.
- close articulated object: if the arm is empty and the object is opened, the agent closes the object.
- grasp an object: if the arm is empty, and the agent grasps the object
- place an object relative to another object: the agent puts the object {inside, on top} of another object. If placing an object inside, the target object must be open.
- done: end the episode and evaluate the success

To execute any action in {open, close, grasp, place}, the robot must first navigate to a position near the target object. Specifically, it samples points along a fixed-radius circle around the object and moves to the closest sampled position.

Algorithm 1: Voronoi Graph Sparsification

```
Require: Voronoi Graph $V$ with nodes $N$ and edges $E$,
    threshold value $c$.
Ensure: Simplified Voronoi Graph
1: Let $E_2$ be the set of nodes in $N$ with a degree of 2
2: for each node $x \in E_2$ do
3:     Let $v$ and $u$ be the neighbors of $x$
4:     Let $w_1$ and $w_2$ be the weights of edges $(v, x)$ and
       $(u, x)$ respectively
5:     if $w_1 + w_2 < c$ then
6:         Remove node $x$ from $V$
7:         Remove edges $(v, x)$ and $(u, x)$ from $V$
8:         Add edge $(u, v)$ with weight $w_1 + w_2$ to $V$
9:     end if
10: end for
```

S.3. Voronoi Graph

A. Creation

The agent perceives a sequence of posed RGB-D frames, $\{I_0, \dots, I_t\}$, including semantic maps of the environment. Using the camera's intrinsic parameters, the depth maps are converted into point clouds. These points are then transformed into a global coordinate frame and arranged on a 3D voxel grid, M_t. Since we address an interactive problem, our map is dynamically updated to incorporate newly explored areas and changes due to manipulated objects.

To determine obstacle positions, walls, and free space, we extract the highest occupied entry per stixel in M_t. This value may be negative in cases where objects like swimming pools are present. These entries are then used to generate a two-dimensional bird's-eye-view (BEV) occupancy map, B_t, by marking all occupied positions except those classified as free space, F_t. The latter represents the navigable area that is used in the exploration.

Similar to MoMa-LLM [2], we construct a navigational graph, G_V from the dense occupancy map. First, we inflate B_t using a Euclidean Signed Distance Field (ESDF) formulation for robustness, while setting the free space coordinates defined in F_t to zero. Based on this, we compute a Generalized Voronoi Diagram (GVD), which consists of a set of points, V, which maintain equal clearance to the nearest obstacles derived from B_t. We exclude all nodes that are on obstacles or do not reside within B_t. Given the boundaries of the GVD, we construct edges E among V and obtain our navigational Voronoi graph, $G_V = (V, E)$.

B. Sparsification

We employ the Voronoi graph as a proxy to extract the closest area to an object or detect frontier points in unexplored areas. However, the graph grows rapidly in large-scale environments. Therefore, any large process on the graph becomes computationally infeasible. In this case, each morphological irregularity consists of a vast number of Voronoi nodes that lie close to each other and do not add any functionality. We propose an iterative sparsification algorithm, described in Alg. 1, which reduces the complexity of the graph while preserving its structure.

Our algorithm removes redundant nodes with a degree of two if their neighbors are sufficiently close. The node is eliminated, and a direct edge is introduced between its neighbors. Note that the removal of one node does not change the degree of any other node, and after each removal step, exactly one degree-two node is eliminated. The algorithm processes each degree-two node only once. The final graph is minimal—i.e., no further degree-two nodes can be removed. This approach runs in a computational complexity of $O(|V|)$.

C. Region Separation at Doors

A scene is divided into regions when no navigable path exists between navigable regions, most commonly due to closed doors or gates. To extract the Voronoi graph for each region separately, we first remove all edges whose nodes belong to different regions. To achieve this, we treat all observed doors as separation gates and remove all edges that cross doors. Mathematically, we construct a 2D plane and fit a Gaussian kernel to each door, where the center of the kernel aligns with the center of the door. The Gaussian kernel diameters are adjusted according to the door's bounding box dimensions. Each door and its associated Gaussian kernel also share the same orientation.

Next, we evaluate each edge by integrating the values of its points in the 2D plane. If this integral exceeds a threshold, it indicates that the edge passes through a Gaussian kernel, that is, a door, implying that its nodes belong to different regions. As a result, when these edges are removed, it ensures a clear segmentation of the scene.

S.4. Scene Graph

Similar to MoMa-LLM [2], we employ a hierarchical 3D scene graph with three levels of abstraction: root, regions, and objects. Consequently, our approach requires (i) distinguishing different regions within the environment and (ii) assigning each object to its corresponding region.

To detect different regions, we use the region separation method described in Supplementary Sec. S.3. Due to partial observability, a single region (e.g., a room or garden) in the environment may be divided into multiple connected components within the Voronoi graph, each covering a sub-region. These sub-regions are merged once the exploration of the region is complete.

For each observed object O, we define $v_p(O)$ as the robot's viewpoint that is currently the closest to the object. We identify the nodes in the Voronoi graph that are in close proximity to the object and its viewpoint, denoted as S_O and $S_{v_p(O)}$, respectively.

For every pair (v, u), where $v \in S_O$ and $u \in S_{v_p(O)}$, we determine the optimal solution to the following minimization problem:

$$\min_{v,u} \quad d_V(v,u) + d_E(O,v)^\lambda + d_E(v_p,u), \tag{1}$$

where d_V represents the shortest path length in the Voronoi graph, and d_E denotes the Euclidean distance in 2D space. By weighting the distance to the object with an exponent of $\lambda = 1.3$, we prioritize selecting nodes that are closer to the object. If v and u belong to different connected components in the Voronoi graph, the shortest path length $d_V(v,u)$ is considered infinite. Once the optimal pair (v,u) is found, the object O is assigned to the connectivity component that contains both v and u.

After constructing the scene graph, the objects within each region are provided to an LLM to predict the region's label (e.g., kitchen, bathroom, etc.). Additionally, we determine the robot's current region and remove all unreachable regions and their associated objects from the scene graph.

S.5. BUMBLE Implementation

In this section, we will introduce the modifications we have made to BUMBLE [3] to adapt it to the BEHAVIOR-1K tasks. All implementation details provided below refer to the latest version of the author's repository at the time of writing[1]. BUMBLE uses Grounded SAM (GSAM) [3] to obtain a segmentation mask of non-background objects. This is achieved by prompting GSAM with the query '*all objects*'. For a fair comparison, we replaced GSAM with the simulator's ground truth information. Specifically, we obtained the segmentation mask and excluded background objects such as the floor and walls. As previously described in section IV, we integrated the same set of subpolicies are used for all models for BUMBLE.

In the original BUMBLE pipeline, the agent could directly interact only with objects visible in the current frame, eliminating the need to store information about previously seen objects. However, for the tasks tackled in this work, the agent may need to interact with previously discovered objects. For this reason, we integrated a new buffer containing previously acquired frames. These frames are continuously sampled during action execution.

Whenever the agent intends to interact with an object, i.e., when performing `navigate` or `go_to_and_*` subpolicies, it must first select an image from the buffer containing the desired target object. The frame selection process is implemented based on BUMBLE's `goto_landmark` skill. The agent is provided with the current subtask to achieve, a list of frames, each associated with a landmark. As we found BUMBLE prone to hallucinations, we additionally provide it with the list of objects within each image, which we found to reduce the problem to some degree. The agent then selects the landmark corresponding to the most suitable frame.

Subsequently, the agent must choose one object from the selected frame. This follows BUMBLE's `navigate_to_point_on_ground` skill. The agent is provided with the subtask to achieve, a description of all the objects in the selected frame, and the annotated frame. For annotating the image, we followed the public BUMBLE code.

The original work does not integrate a subpolicy for exploring the environment, relying instead on hardcoded positions within the tested buildings, which the robot could navigate between using the `goto_landmark` skill. We incorporate an exploration subpolicy following the same approach of the `goto_landmark` skill, but, instead of using hardcoded positions, we provide the VLM with frames from the buffer where the unexplored areas are visible and a description of all the objects in each frame. The unexplored areas are extracted from the scene graph. The agent subsequently chooses the image that depicts the areas to be explored. If multiple areas to explore are present in an image, the closest one is selected.

In BUMBLE, the feedback received after executing an action must be related to the subtask generated by the agent. For the subpolicies `navigate` and `go_to_and_*` we use the same feedback as for our method. For the more abstract exploration subpolicy, feedback is generated by the Visual Language Model (VLM). The VLM is tasked with determining whether the subtask was accomplished during the exploration. It is provided with the frames acquired during the exploration. Again, beyond BUMBLE's original implementation, we additionally include the list of objects in the frame to reduce halucinations.

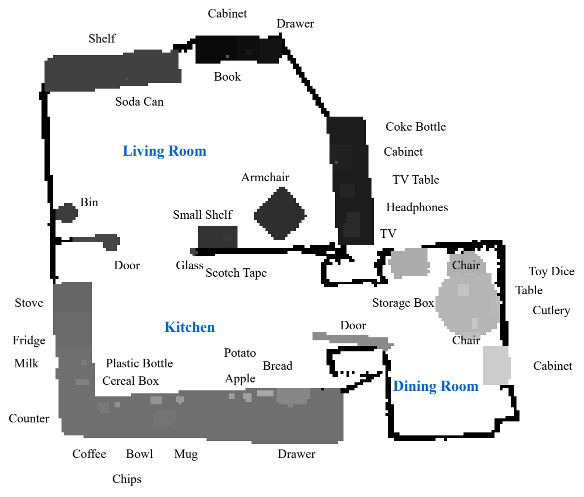


Fig. S.1. Map of the real world environment. Object positions are shown in different intensities and annotated with their category name.

S.6. Real World Implementation

A. Map

The real world environment spans three rooms, furnished as kitchen, living room and dining room with six articulated, openable objects and 38 different objects overall. A map of the environment is shown in Fig. S.1.

B. Subpolicies

As discussed in Sec. IV-C, we simulate accurate semantic perception by annotating a recorded 2D map of the environment with semantic labels. We then use the depth camera of the robot to identify the explored area, and reveal the according parts of the map to robot. The recorded map is also used for LiDAR based localization. This allows us to

[1]https://github.com/UT-Austin-RobIn/BUMBLE, commit *4ff5ac0*.

isolate the factors relevant to the decision-making part of problem.

`Navigate`: the agent uses the default omnidirectional ROS Navstack provided by the robot's manufacturer. Obstacles are avoided based on the static map, inflated by $0.25\,\mathrm{m}$ as well as dynamic obstacles perceived with the robot's LiDAR and depth camera mounted on the head.

Manipulation: All manipulation motions consist of an inferred end-effector motion. This end-effector motion is converted to a whole-body motion through the $\mathrm{N}^2\mathrm{M}^2$ mobile manipulation policy [40], [41]. This policy takes the desired end-effector motion together with the robot state and a local occupancy map for collision avoidance and produces joint velocities for the full robot at a high control frequency.

`Open object`: We provide a single demonstration for the motion of the end-effector relative to an AR marker for each articulated object.

`Pick object`: Object poses are detected via AR markers. We then define a grasp pose relative to the marker. To grasp the object, we generate a linear end-effector motion from a point in front of the grasp pose to the grasp pose and back to a point in front and slightly above the grasp pose.

`Place object`: We identify the mask of the object where the agent wants to place something down. We erode this mask and take the point closest to the robot as place pose. We predefine the height of all placeable areas to select the relevant height component. We then generate linear end-effector motions from a point in front and above the place pose to a pose above the place pose. Followed by a linear motion to the place pose and back to a point in front of the place pose.

References

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

[2] R. Shah, A. Yu, Y. Zhu, Y. Zhu, and R. Martín-Martín, "Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation," *Int. Conf. on Robotics & Automation*, 2025.

[3] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan *et al.*, "Grounded sam: Assembling open-world models for diverse visual tasks," *arXiv preprint arXiv:2401.14159*, 2024.

[4] D. Honerkamp, T. Welschehold, and A. Valada, "Learning kinematic feasibility for mobile manipulation through deep reinforcement learning," *IEEE Robotics and Automation Letters*, 2021.

[5] ——, "$\mathrm{N}^2\mathrm{m}^2$: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *IEEE Transactions on Robotics*, 2023.

TABLE S.1

Task names & task description for 81 BEHAVIOR-1K tasks

Task name	Task description
putting_away_tools	put tools in toolbox. Wrenches must be in the same toolbox. Also screwdrivers must be in the same toolbox
attach_a_camera_to_a_tripod	put the digital camera on top the tripod
sorting_vegetables	distribute chard and bok choy in one bowl, leek and artichoke in another bowl, and sweet corns in another one
setting_up_room_for_games	put games and dices on $table.n.02_1
recycling_office_papers	throw all legal documents in $recycling_bin.n.01_1
shopping_at_warehouse_stores	put the cream cheese, watermelon, pomegranate, baguette and corn flakes on checkout
put_together_a_scrapping_tool_kit	put the tools in the toolbox
putting_dishes_away_after_cleaning	put all the plates in $cabinet.n.01_1. In total there are 8 plates
fold_a_tortilla	put $tortilla.n.01_1 on $plate.n.04_1 and $tortilla.n.01_2 on $tortilla.n.01_1
putting_shopping_away	store dairy and meat in $electric_refrigerator.n.01_1, and other food and oil in $cabinet.n.01_1
remove_sod	put the sods on top of the wheelbarrow
distributing_groceries_at_food_bank	put milk, pasta, apple juice and canned for in each box
buy_eggs	put $egg.n.02_1 and money on the checkout
packing_fishing_gear_into_car	load all fishing stuff into the car
hoe_weeds	put weeds in $ashcan.n.01_1
loading_the_car	put $bag.n.06_1, $sack.n.01_1 and $laptop.n.01_1 in the car
buy_a_good_avocado	put the avocado and money on the checkout
packing_documents_into_car	load the documents and the book into the car
pack_a_pencil_case	put the pencil, pen, eraser and shear in the pencil box
packing_car_for_trip	put sunglasses, money and laptop in suitcases, and load all suitcases into the car
packing_art_supplies_into_car	put the pencil and markers in the suitcase, and put the suitcase in the car
store_beer	store all beer bottles in $electric_refrigerator.n.01_1
moving_stuff_to_storage	put ice skates and bowling balls in one carton while the painting and textbooks in the other one
outfit_a_basic_toolbox	put all the tools in the tool box
organizing_boxes_in_garage	put balls, plates and saucepan in boxes
organizing_office_documents	put legal docs in a folder. Then, put post-it on the folder, and put the folder in $cabinet.n.01_1
packing_books_into_car	load all the books into the car and make sure suitcase is not inside the car
buy_boxes_for_packing	put the cartons and money on the checkout
cleaning_parks	put bottles in $recycling_bin.n.01_1
buy_dog_food	put the dog food and money on the checkout
buying_groceries	put milk, apple juice and prawn in one bag, and peanut butter, brown rice and banana in another bag. put all the money in $cash_register.n.01_1
buy_a_air_conditioner	put the air conditioner and money on the checkout
buy_a_keg	put the beer barrel and credit card on the checkout
buy_pet_food_for_less	put the canned foods and dog foods on checkout and all money in $cash_register.n.01_1
clearing_the_table_after_dinner	put bowls, cups and bottles in $bucket.n.01_1
buy_alcohol	put the wine, vodka, beer and money on the checkout
buying_groceries_for_a_feast	put all the groceries in a shopping card, and money in $cash_register.n.01_1
stacking_wood	stack all logs on top of each other and put them on $table.n.02_1
cleaning_up_branches_and_twigs	put branches in $ashcan.n.01_1
buy_food_for_a_party	put all pasta, milk, apple juice, and cake on the checkout. Put all the money in $cash_register.n.01_1
buy_candle_making_supplies	put candle sticks, paraffin and money on the checkout
putting_out_condiments	put the knifes, pickles, spaghetti sauce and all the bottles on $breakfast_table.n.01_1
returning_videotapes_to_store	put DVDs in $shelf.n.01_1
putting_leftovers_away	store all lasagna and spaghetti sauces in $electric_refrigerator.n.01_1
boxing_books_up_for_storage	put all the books in a carton
assembling_gift_baskets	put a cheese, candle, cookie and bow in each basket
re_shelving_library_books	put $book.n.02_1, $book.n.02_2, $book.n.02_3, and $book.n.02_4, $book.n.02_5, $book.n.02_6, $book.n.02_7 and $book.n.02_8 on $shelf.n.01_1
sorting_bottles_cans_and_paper	distribute water bottles in one bucket, cans in another bucket, and magazines and newspapers in another one
preparing_lunch_box	store all the food and drink in $packing_box.n.02_1
bringing_newspaper_in	put the newspaper on $coffee_table.n.01_1
put_togethera_basic_pruning_kit	put the pruner and the shears in the toolbox
bag_groceries	put egg, apple, orange juice and canned food in a bag
bringing_water	put bottles in $cabinet.n.01_1
preparing_food_for_a_fundraiser	distribute the bottles in cartons. Put the food on tupperware and then put tupperware in cartons