arXiv:2410.06237v1 [cs.RO] 8 Oct 2024

BUMBLE: Unifying Reasoning and Acting with Vision-Language Models for Building-wide Mobile Manipulation

Rutav Shah Albert Yu Yifeng Zhu Yuke Zhu* Roberto Martín-Martín*
The University of Texas at Austin

***Abstract*— To operate at a building scale, service robots must perform very long-horizon mobile manipulation tasks by navigating to different rooms, accessing different floors, and interacting with a wide and unseen range of everyday objects. We refer to these tasks as Building-wide Mobile Manipulation. To tackle these inherently long-horizon tasks, we introduce BUMBLE, a unified Vision-Language Model (VLM)-based framework integrating open-world RGBD perception, a wide spectrum of gross-to-fine motor skills, and dual-layered memory. Our extensive evaluation (90+ hours) indicates that BUMBLE outperforms multiple baselines in long-horizon building-wide tasks that require sequencing up to 12 ground truth skills spanning 15 minutes per trial. BUMBLE achieves 47.1% success rate averaged over 70 trials in different buildings, tasks, and scene layouts from different starting rooms and floors. Our user study demonstrates 22% higher satisfaction with our method than state-of-the-art mobile manipulation methods. Finally, we demonstrate the potential of using increasingly-capable foundation models to push performance further. For more information, see https://robin-lab.cs.utexas.edu/BUMBLE/**

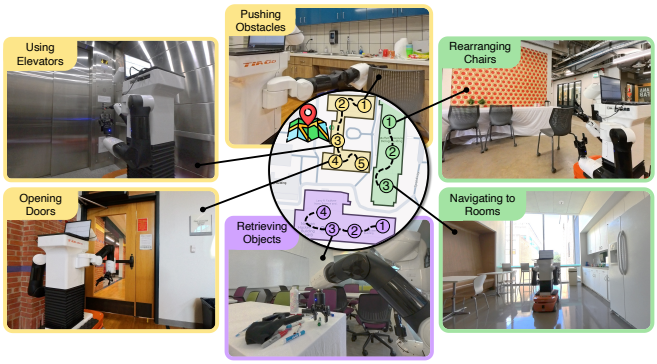

Fig. 1. **Building-wide mobile manipulation**. At a building-wide scale, mobile manipulation tasks involve sequencing multiple skills like pushing obstacles or opening doors to clear robot pathways, using elevators to reach a destination floor, rearranging chairs in the workspace, and retrieving objects. We present a framework, BUMBLE, that can solve long-horizon, spatially expansive tasks across different buildings.

I. Introduction

Autonomous service robots aiming to assist humans daily must perform mobile manipulation (MoMa) in homes, hospitals, universities, offices, and other building-wide environments. Given a user instruction, like "I am on a diet, but I want soda," the robot must interpret the free-form instruction, create a high-level task plan to accomplish the task, and instantiate low-level robot commands to fulfill it. To accomplish such tasks in buildings, the robot might have to traverse to a kitchen, potentially using an elevator if it is on a different floor. To use the elevator, the robot must draw on prior experiences with elevators in other buildings. The robot must also handle unexpected obstacles, such as by pushing items blocking a narrow corridor or opening doors to move in and out of the room. Finally, even after reaching the kitchen, the robot must identify an appropriate soda can from the diverse clutter often present in human-inhabited buildings, recognizing the "diet" option even if it has never been encountered before. We refer to such long-horizon and spatially expansive tasks as **Building-wide Mobile Manipulation** (see Fig. 1) and present a unified framework to address it.

Building-wide MoMa entails a series of technical challenges, including integrating complex locomotion and manipulation behaviors in buildings, perceiving objects and their context in the open-world, and using past experiences to learn from mistakes reason about long-horizon strategies (example execution in Fig. 2). Decades of research in Task and Motion Planning (TAMP) [1–4] and recent neuro-symbolic planning [5–8] have been applied to long-horizon robotic tasks. However, these approaches require pre-defined symbolic abstractions and domain knowledge, confining their reasoning capabilities to a closed set of objects. Large Language Models (LLMs) [9–14] are capable of reasoning about open-world scenarios to create task plans. However, they lack grounded and direct reasoning through visual perception, inherently falling short in the precise geometric reasoning crucial for producing executable robot actions. Recently, Vision-Language Models (VLMs) have emerged as a promising tool for Building-wide MoMa, unlocking grounded reasoning in open-world scenes [15–23]. Some works [24–28] have demonstrated the potential of VLMs in simple MoMa tasks but lack the necessary memory and motor skills to operate at a building scale. In contrast, building-wide MoMa requires 1) an open-world perception system for reasoning about diverse objects, 2) complex motor skills to act effectively in buildings, and 3) memory for temporally extended reasoning in long-horizon task execution.

We propose to tackle Building-wide MoMa by unifying reasoning and acting through a VLM-based framework—one that can handle diverse objects with open-world perception, act effectively in building-wide situations with a broad set of gross-to-fine motor skills, and learn and adapt from past experience through memory. By seamlessly unifying

* Equal advising. Correspondence: rutavms@utexas.edu

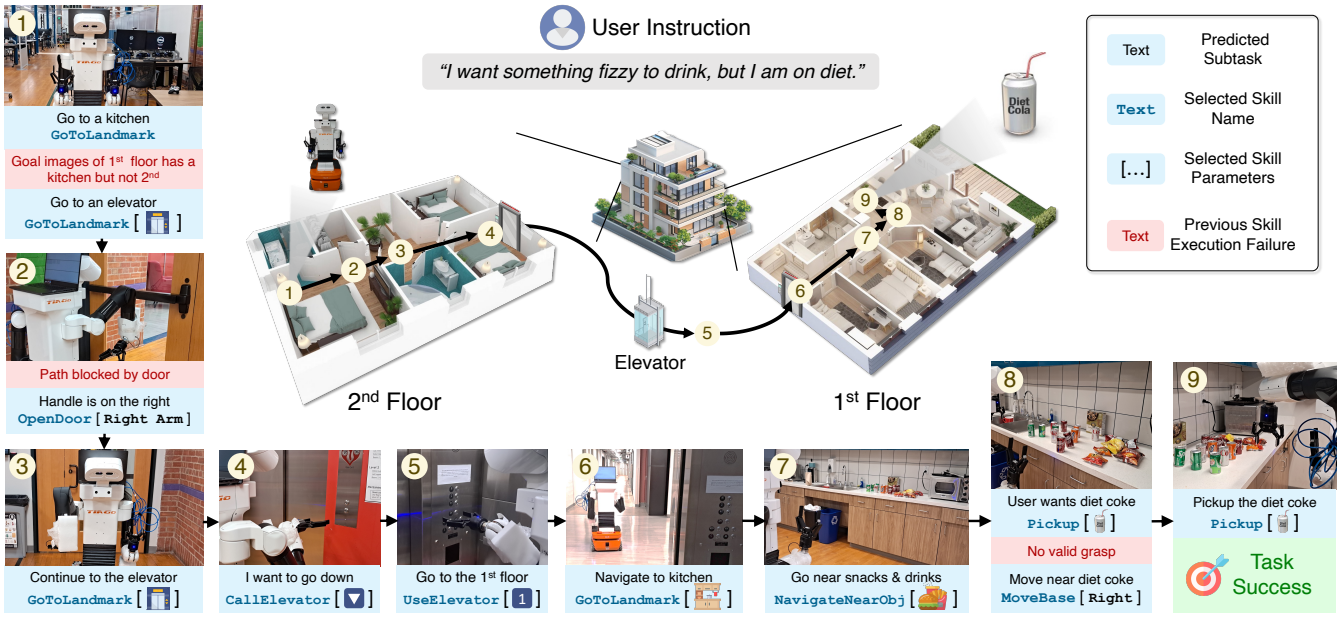

Fig. 2. **Building-wide Mobile Manipulation** tasks require navigating various rooms and floors while interacting with diverse objects. To solve such long-horizon tasks, BUMBLE leverages a diverse skill library enabling navigation and interaction, with parameterized skills adaptable to different scenes. At each skill execution step (circled numbers), BUMBLE uses VLM's reasoning capabilities to select the next skill (blue text) and skill parameters (**[...]**) and recover from failures (red text) to solve building-wide tasks effectively.

these capabilities into rapidly advancing VLMs, our general framework improves as VLM backbones become better at sophisticated perception and reasoning.

To this end, we introduce BUMBLE (**BU**ilding-wide **MoBiLE** Manipulation), a new framework that addresses all of these needs with an adaptable approach capable of reasoning and acting over new objects, tasks, and buildings. Given a task instruction in free-form language, BUMBLE reasons over the scene, past experiences, and motor skill capabilities to predict the next parameterized skill to execute. BUMBLE comprises four key ingredients: a) a **VLM** serving as the central reasoning module connecting perception, memory, and skills; b) **dual-layered memory**: short-term memory [29] to maintain robot *execution history*, and long-term memory [30] to store valuable experience and concepts from *past trials*; c) a diverse **skill library** of parameterized skills that enable the robot to navigate to different floors and rooms (e.g., `GoToLandmark[GoalImage]`, `UseElevator[Button]`), adjust the robot base for manipulation (e.g., `MoveBase[Dir.]`), and interact with objects through diverse contact-rich behaviors (e.g., `PushObjOnGround[ObjSeg., Dir.]`) present in different buildings; d) **perception system** for the VLM to not only visually reason about open-world, cluttered scenes but also process depth information necessary for embodied decisions.

Our experiments demonstrate that BUMBLE can successfully execute long-horizon, building-wide tasks requiring up to 12 parameterized skills and over 15 minutes per trial (excluding VLM query time). These tasks involve navigating different floors and rooms, clearing pathway obstacles, retrieving objects, and making granular decisions, such as adjusting the base for manipulation. Notably, BUMBLE achieves a 47.1% average success rate in tasks like retrieving soda cans and rearranging chairs across three buildings. We demonstrate that BUMBLE solves previously unseen tasks involving multiple plausible solutions and operating in completely new rooms. Through a user study, we find that BUMBLE rollouts align with human preferences 22% better than prior state-of-the-art VLM-based MoMa approaches. Furthermore, we demonstrate how BUMBLE improves performance with advancing VLM capabilities. Finally, we thoroughly analyze failures, guiding future research.

II. Related Work

Pioneering research on **mobile manipulators** [31–35] performed tasks in human environments, which naturally occur in buildings [36–38]. Early work in whole-body control [39–43] and motion planning [44–48] laid the groundwork for low-level control in building-level mobile manipulation, but did not address reasoning required in long-horizon building-wide tasks. Task and motion planning (TAMP) [1–4, 49] integrates reasoning for task and motion generation to achieve long-horizon tasks. However, it is confined to a fixed preprogrammed set of object categories and scenes, hindering its application to building-scale environments with open-world objects.

Recently, **foundation models** [50] have been leveraged for mobile manipulation. LLMs excel at semantic reasoning for planning [51, 52], but their predictions are not grounded in the scene, requiring additional machinery to match their plans to executable robot actions [9, 11, 53, 54]. VLMs provide a more promising engine to unify and ground reasoning for tabletop manipulation [15–17, 26], navigation [19–22, 55], and mobile manipulation [24, 25,

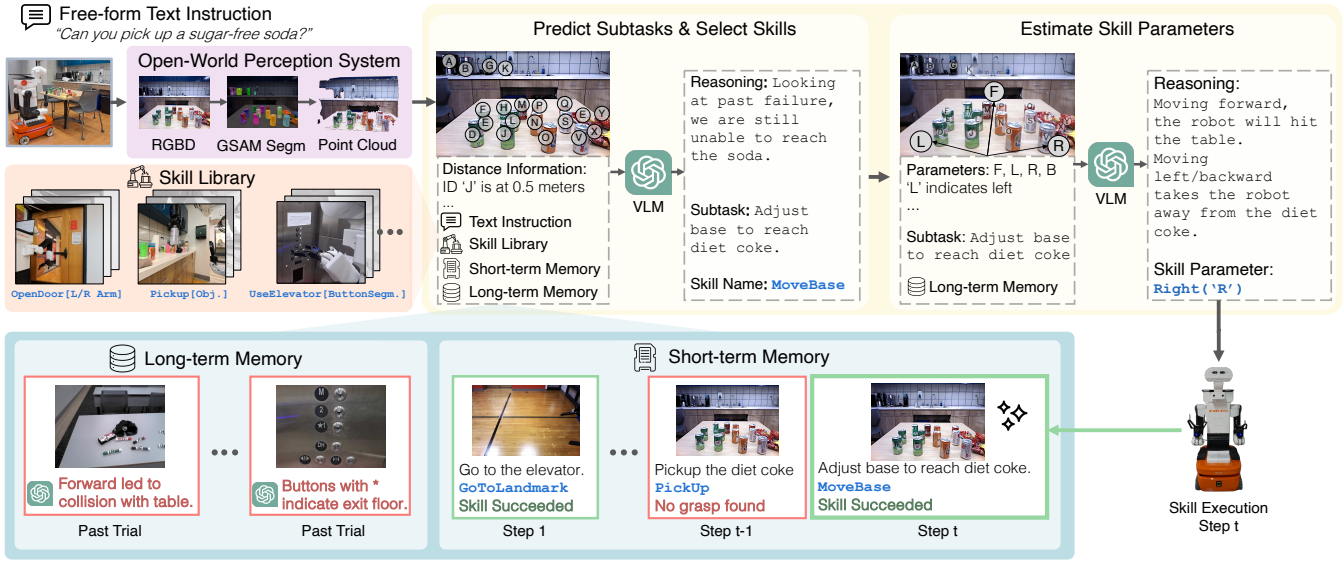

Fig. 3. **BUMBLE Architecture**. Given a free-form text instruction, skill library (*top Left*), and short- and long-term memory (*bottom middle and left*), BUMBLE iteratively perceives the environment through onboard RBGD sensors (*top left*), predicts parameterized skills (*top middle and right*), and executes them in the environment. The predicted skill and its parameters are executed and stored in short-term memory for iterative prediction (*bottom right*).

27, 28, 56], but prior methods lack a diverse skill repertoire (navigation and manipulation, coarse and fine motion) and long-horizon memory-based reasoning capability, both of which are necessary to perform building-wide mobile manipulation tasks. BUMBLE is a natural extension of VLM-based robotic systems: it uses VLMs to unify semantic and geometric reasoning for manipulation and navigation at the building-wide scale, enabling it to leverage current and future advances in VLM models. Our framework integrates perceptual and motor skills and long and short-term memory to obtain efficiency and generalization to diverse objects and scenes in different buildings.

All aforementioned prior work in mobile manipulation assumes the navigation path is either obstacle-free or that obstacles can be navigated around. In **interactive navigation** [57, 58] however, obstacles must be moved or manipulated—such as pushing away path-blocking boxes, opening doors, or pressing elevator buttons—to reach a destination. Prior methods tackled this with motion planning [59–61] and learning [62–64] but focused solely on the interactive navigation problem using only geometric information. In contrast, BUMBLE integrates both semantic reasoning (e.g., avoiding pushing delicate objects) and geometric reasoning (e.g., avoiding object collisions while pushing an obstacle) into a unified method for mobile manipulation.

III. BUMBLE: VLM-Based Building-Wide MoMa

BUMBLE is a new framework integrating perception, motor skills, and memory in a VLM-based solution for high-level reasoning and low-level control of a mobile manipulator for building-wide tasks. In this section, we describe the key technical blocks of BUMBLE (see Fig. 3): (a) capabilities for open-world perception, a library of diverse skills to act in the physical world, and short and long-term memory for on-the-fly adaptation and learning from past mistakes; and (b) a VLM-based decision-making module to reason and ground the text instruction in the current scene and predict the next parameterized skill to execute (see App. VII-A for details).

A. Perception System, Skill Library, and Memory

a) Open-world Perception System: Robots must perceive and localize diverse open-world objects to operate effectively in buildings. We address this using a perception system with a robust segmentation model, Grounded-SAM (GSAM) [65–67], for segmenting foreground (i.e., interactable) objects (Fig. 3, purple). We use GSAM instead of directly querying the VLM because segmentation models offer pixel-level accuracy, allowing for more precise localization and manipulation. After obtaining the object masks for the scene, we calculate the object point cloud by back-projecting depth images and determine the precise distance between the robot and the detected objects. The object proximity information enables BUMBLE to estimate the correct skill for manipulation in building environments, while the object point cloud enables precise manipulation using motor skills.

b) Skill Library: To overcome the challenges of building-wide MoMa, BUMBLE requires a diverse skill library allowing for high-level abstract behaviors like navigating to a room, down to more fine-grained behaviors like adjusting the base for manipulation (Fig. 3, yellow; examples in Fig. 2, 4). The skill library must also support using building-specific fixtures like elevators and perform manipulation skills like pushing objects and picking up items, addressing interactive navigation challenges [47, 58]. To this end, we construct a skill library to which new skills can easily be added and chosen by the VLM. Our library includes skills such as `GoToLandmark`, `NavigateNearObj`, `MoveBase`, `Pickup`, `PushObjOnGround`, `OpenDoor`, `CallElevator`, and `UseElevator`. The parameters of each skill are based on object or robot configurations, except

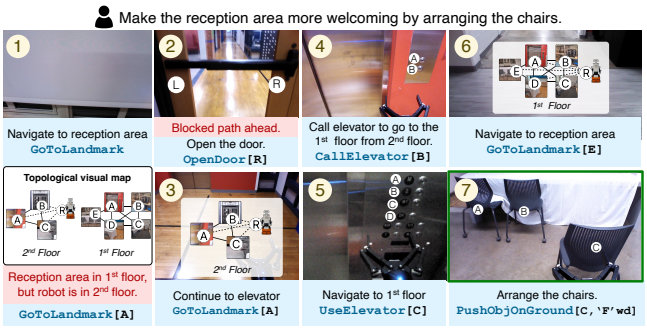

Fig. 4. **Execution trace for a user instruction.** For each decision step in the execution trace, we show the image observation with grounded skill parameters as markers, the skill name (blue text) and parameter executed, and a brief language description of the step (black) to improve readability. In trying to execute some skills, the robot fails (red text), in which case the VLM adaptively predicts the next skill.

for `GoToLandmark`. It uses a topological visual map [68] of the building, with landmark images as nodes and 2D occupancy maps to generate trajectories between them (See Fig. 4). With the library of parameterized skills and VLM's reasoning capabilities, BUMBLE can integrate semantic and geometric reasoning at the task level by predicting the subtask and skill and the robot's motion by estimating skill parameters to accomplish building-wide tasks.

c) Memory: Building-wide MoMa is inherently long-horizon, requiring the robot to track its state-action history. Moreover, the robot must recover from failed skill executions whenever possible, such as by moving away obstacles or adjusting the robot base after a failed grasping attempt. To enable these recovery behaviors, the robot must store and analyze prior failed executions when making future predictions. To address this issue, BUMBLE maintains **short-term memory** (Fig. 3, blue). The short-term memory stores the scene image, subtask, skill name, parameter, and the system-detected execution result (success/failure) at each prediction step for the *current execution trial*. This allows the VLM to reason over the entire execution history when predicting its next skill, crucial for long-horizon building-wide tasks.

The VLM may make reasoning errors, such as failing to consider object collisions with the environment, which could, for example, result in pushing an obstacle into a wall. To allow BUMBLE to learn from these mistakes and reduce them [69], BUMBLE maintains a **long-term memory** of previously collected prediction failures. Erroneous VLM predictions, flagged by a human operator, are stored together with the decision context (user instruction, scene image, predicted subtask, skill name, and predicted parameters) and a VLM text analysis of the failure reason, serving as lessons for improving future predictions and reducing the dependency on human annotations.

B. VLM-based Decision Making

BUMBLE factorizes each decision-making step into two sequential steps (Fig. 3, yellow box). First, BUMBLE exploits a VLM to perform perceptual and memory-based reasoning for predicting the next **subtask** to solve (e.g., "go to a place where you can find a soda") and for selecting the **skill** to handle it (e.g., `GoToLandmark`). Then, BUMBLE queries the VLM to estimate the best **skill parameters** (e.g., `goal image of kitchen`) and instantiates the corresponding motor skill execution.

a) Subtask prediction and skill selection: First, BUMBLE needs to infer the next step of the task and what skill to use to solve it. To that end, it integrates information about i) the language task instruction, ii) the description of robot skills, iii) the current scene (RGB) and robot state (location in the building), and iv) execution history (short-term memory) and failures from prior trials (long-term memory) into a *unified multimodal VLM query*.

The VLM must also perceive the object distance information to infer the object's reachability, which is necessary for manipulation. Naively prompting the VLM with depth images or point clouds does not yield desirable results because VLMs are not trained on those modalities. To allow the VLM to make informed decisions, BUMBLE implements a multimodal Set-of-Mark (SoM) [70] prompting: foreground objects are marked with IDs in the input RGB image, and their distances to the robot, calculated using the perception system (Sec. III-A), are passed in the prompt.

BUMBLE employs the resulting multimodal prompt to query the VLM-backbone about the next step in the task and the best-suited skill to solve it, requesting intermediate reasoning with Chain-of-Thought (CoT) [71] for better accuracy. With this VLM-backed procedure, BUMBLE can reason about long-horizon tasks, eliminating the need for a classical planner's computationally heavy search.

b) Skill parameter estimation: After predicting the subtask and skill to execute, BUMBLE predicts the parameters needed to correctly instantiate the skill and generate a sequence of low-level robot commands. These parameters must be contextualized and grounded in the current scene for informed decision-making. To that end, BUMBLE creates a multimodal VLM query that includes i) the predicted subtask, ii) memory of previously made mistakes in predicting the skill parameters, and iii) an SoM-based visual grounding of the possible skill parameters with explanations of each marker (example in Fig. 4). For example, for the `MoveBase` skill, the possible parameters (forward, left, and right) are marked in the RGB image as arrow endpoints. For object-centric skills, the candidate parameters are generated by querying the GSAM with a skill-specific prompt (e.g., find 'Buttons' for `UseElevator`). BUMBLE queries the VLM to choose the most promising skill parameter using CoT prompting for better reasoning about the subtask, scene, and past experiences. The execution outcome is stored in the short-term memory for iterative prediction.

IV. Experimental Evaluation

To systematically evaluate the performance of BUMBLE, we benchmark **three** very long-horizon tasks up to 12 skills long: *Retrieving marker*, *Retrieving diet soda can* and *Rearranging chairs*. To test the system's language understanding capabilities, the tasks are specified using free-form text

TABLE I

SUCCESS RATE (%) FROM 10 TRIALS IN BUILDING-WIDE TASKS

	Retrieve marker			Retrieve soda can			Rearrange chairs	Avg.
Building	*B1*	*B2*	*B3*	*B1*	*B2*	*B3*	*B1*	(%)
IM [72]	10	–	–	0	–	–	10	6.7
COME [28]	40	30	40	**40**	30	30	40	35.7
BUMBLE	**60**	**40**	**50**	**40**	**50**	**40**	**50**	**47.1**

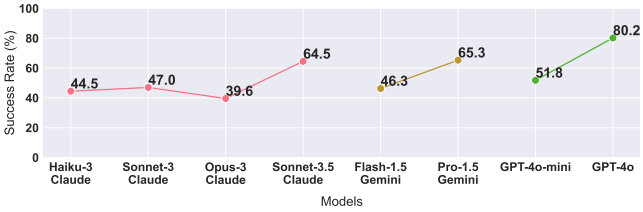

Fig. 5. Success rate (%) of VLMs in predicting skill parameters. The models are arranged with increasing capabilities, measured as per the Vision Arena [73], from left to right in *each model series*.

instructions, and the evaluations are averaged over three different task specifications for each task. For instance, in the retrieving marker task, we use "*I want to color the sky in my drawing. Can you get me a marker?*", "*I want to color grass in my drawing. Can you get me a marker?*" and "*I need to color some hearts. Can you get a marker for that?*" (Refer to App. VII-B for details).

We test BUMBLE's building-wide MoMa capabilities by initializing the agent on different floors (possibly requiring elevator use), with randomized obstacles along the robot's path, such as closed doors, chairs, wet floor signs, and cardboard boxes. BUMBLE must reason geometrically (e.g., selecting the correct arm to open a door or direction to push an obstacle) and semantically (e.g., avoiding wet floor signs).

We evaluate open-world generalization with multiple target objects per task—different brands of diet soda and markers, in random positions under varying degrees of clutter (5-25 distractor objects). We test building-level generalization across three university buildings of different layouts, visual appearance, and room structures. Results are measured using 10 trials per building per task (totaling 90+ hours of evaluations). Since BUMBLE demonstrated consistent performance on the first 2 tasks across the 3 buildings, we only evaluated our third task, rearranging chairs, in one building.

We compare BUMBLE with two baselines: **Inner-Monologue (IM)** [72] that reasons using only a language scene description without RGB images or long-term memory, and **COME** [28], which reasons using images, like BUMBLE, but without long-term memory. Note that both methods were originally unsuitable for building-wide tasks due to the limited skill library and lack of memory, so we extended both with BUMBLE's diverse MoMa library and short-term memory, for fair comparison. We instantiate BUMBLE with GPT-4o for our experiments since we found it empirically to perform best. We gather long-term memory experiences before evaluations and do not update them during evaluations.

Finally, in our evaluation, we complement the full-fledged long-horizon building-wide MoMa experiments described above by only assessing the parameter estimation for given skills in predefined scenes. To that end, we manually collected an offline dataset (`OfflineSkillDataset`) of around 120 images for three different skills and annotated them with ground truth parameters.

In our evaluation, we investigate the following questions:

1) How does BUMBLE perform compared to other methods when addressing building-wide tasks? Table I summarizes the results of our experiments. Inner-Monologue (IM) performs poorly compared to COME and BUMBLE, indicating that multimodal prompting with visual information is critical to making the right decisions. Due to its poor performance, we restricted the evaluation of IM to that first building. BUMBLE scores best in all tasks and buildings and outperforms COME by 12.1% on average, thanks to learning from past mistakes. To directly compare the reasoning capability of these approaches over a diverse but fixed set of scenarios, we test each method against the ground truth skill parameters in `OfflineSkillDataset`. BUMBLE achieves 80.2% success, outperforming COME (72.6%) and Inner-Monologue (61.7%), suggesting that BUMBLE's improvements in single-step skill parameter prediction help drive more successful executions in long-horizon tasks.

2) How well does BUMBLE scale with increasing VLM capabilities? In BUMBLE, we sought to unify reasoning with a VLM-backbone to leverage advances in this fast-paced field. For that, BUMBLE would have to work well with different VLMs and demonstrate improving capabilities with more powerful versions. Fig. 5 includes the results of our evaluation of BUMBLE using Claude (Red), Gemini (Yellow), and GPT-4o (Green) model series as VLM-backbone on the `OfflineSkillDataset`. BUMBLE works well with any VLM, and its performance with each model scales with VLM capabilities, highlighting the potential of BUMBLE to leverage next-generation foundation models.

3) What are the common failure modes in BUMBLE? We analyze and break down the failure cases observed during the evaluation of BUMBLE in the three long-horizon tasks and present the results in Fig. 6. Several of our errors (10/38) are caused by sensor failures (bad depth values, lidar failures) or GSAM segmentation mistakes, but most of them can be attributed to VLM reasoning mistakes, e.g., picking the wrong object, wrongly adjusting the base or failing to solve an interactive navigation problem [47]. We notice that these errors occur more often when BUMBLE queries the VLM to localize markers demanding precise spatial understanding, such as elevator buttons, which often leads to pushing an incorrect one. The VLM is also likelier to pick an incorrect object when there are many distractors (20-25, 38.9%) than a few (5-10, 10.0%), indicating limitations when reasoning with clutter, possibly due to presence of numerous markers.

During evaluations, we noticed a surprising emergent behavior: BUMBLE overcomes sensor failures using its

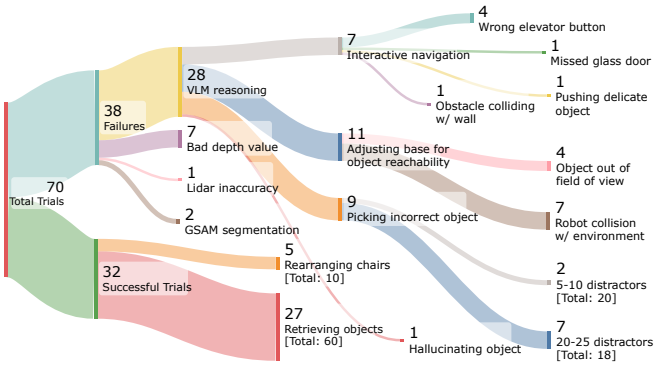

Fig. 6. **Success and Failure Categorization** from 70 trials. Most failures involve VLM reasoning, such as predicting actions that lead to a collision with the environment, picking up the wrong object, especially when there are 20-25 distractors, or pressing the wrong elevator button.

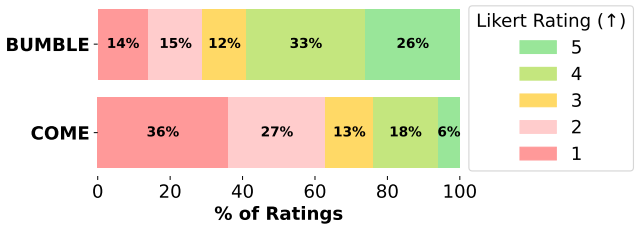

Fig. 7. **Human ratings on task completion and user satisfaction** ($n = 10$ participants, 10 robot trials evaluated per participant per method) on a 5-point Likert scale. BUMBLE is rated better than COME at aligning to human preferences.

diverse skill library. We observed that BUMBLE learns to use `MoveBase` to a) reposition towards the elevator buttons, and b) approach an obstacle (chair), both after failed manipulation attempts due to NaN depth sensor values.

4) How well does BUMBLE align to human judgement? To analyze how well BUMBLE decisions align with human judgment, we test on two completely unseen tasks without any prompt changes. These two tasks involve navigating to a completely new (shower) room: a) "*I spilled water on the floor. Can you pick up something to clean?*", b) "*I dropped my phone in water and dried it off with a napkin, but it's still not working. Can you pick up something to get rid of all the moisture?*". We then collect images of the execution using BUMBLE and COME and annotate them with the task description and the parameterized skill selected by each one. We ask 10 participants to annotate each trajectory by selecting the description that best fits the result of the robot's last decision. Participants choose from five categories: **1**: Irrecoverable failure causing permanent damage to the robot or environment, **2**: Recoverable failure that can be corrected without lasting damage, **3**: Ill-specified decision, **4**: Sub-optimal task completion, **5**: Task completion that fully satisfies the user request. Participants respond on an online form with questions shuffled with a mix of decisions made by COME and BUMBLE.

Fig. 7 summarizes the results of our survey. BUMBLE achieves a Likert rating of 3.7 out of 5.0. Compared to COME (Avg. rating: 2.6), BUMBLE has 22% points less irrecoverable and 12% points less recoverable failures. Participants rate BUMBLE as suboptimally completing the task 33% of the time, such as successfully picking up an object, but not the one most suitable on the scene. We hypothesize this is due to greedy plans generated by VLMs, which often pick up the nearest, most visible object. Incorporating multi-step reasoning using VLMs to overcome greedy behavior is an exciting future direction for BUMBLE.

5) What is the contribution of CoT and SoM to BUMBLE? We ablate two components of the BUMBLE VLM interface: Chain-of-Thought reasoning and Set-of-Mark prompting to measure their effect on performance. Not using CoT decreases performance by 19.5% points, down to 60.7%, underscoring the importance of having intermediate reasoning steps for decisions in building-wide tasks. Removing SoM and instead using the VLM to predict a desired object description in natural language, then asking GSAM to segment this object [26], decreases performance by 31% points, highlighting GSAM's difficulty in segmenting specific object instances such as "*Diet* Dr. Pepper" when a non-diet Dr. Pepper can is present (See Table II).

V. Conclusion

We demonstrate that a VLM-backed policy, acting as a central reasoning module equipped with a diverse skill library, memory, and open-world perception system, can effectively solve building-wide mobile manipulation tasks. Comprehensive ablations and experiments demonstrate the potential of BUMBLE as a stepping stone for achieving autonomous service robots in buildings.

Limitations and Future Work: BUMBLE's long-term memory storing previous failures can become computationally intractable with time; using a dynamic retrieval process from a large storage space or compressing these experiences in learnable weights can be explored in future work. Furthermore, BUMBLE assumes access to landmarks with goal images in the `GoToLandmark` skill, which requires human effort in collecting the landmarks. Autonomous exploration with landmark frame selection can be explored to offload human effort. Lastly, while BUMBLE can be easily interfaced with semantically meaningful skills, adding skills or parameters that cannot be easily expressed via language [74, 75] requires additional machinery for video conditioning or specialized weights to allow such interfacing.

VI. Acknowledgments

We thank Shivin Dass for developing the robot system infrastructure. We also thank Ben Abbatematteo, Shuijing Liu, and Mingyo Seo for their feedback on the manuscript. We thank all UT Austin Robot Perception and Learning Lab and Robot Interactive Intelligence Lab members for their invaluable feedback on BUMBLE. This work was partially supported by the National Science Foundation (FRR2145283, EFRI-2318065), the Office of Naval Research (N00014-22-1-2204, N00014-24-1-2550), and the DARPA TIAMAT program (HR0011-24-9-0428).

References

[1] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 1470–1477.

[2] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2014, pp. 639–646.

[3] C. R. Garrett *et al.*, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[4] J. Wolfe, B. Marthi, and S. Russell, "Combined task and motion planning for mobile manipulation," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 20, 2010, pp. 254–257.

[5] D. Xu *et al.*, *Neural task programming: Learning to generalize across hierarchical tasks*, 2018. arXiv: 1710.01813 [cs.AI].

[6] D. Xu, R. Martín-Martín, D.-A. Huang, Y. Zhu, S. Savarese, and L. Fei-Fei, *Regression planning networks*, 2019. arXiv: 1909.13072 [cs.AI].

[7] J. Mao, T. Lozano-Pérez, J. B. Tenenbaum, and L. P. Kaelbling, "Pdsketch: Integrated planning domain programming and learning," *arXiv preprint arXiv:2303.05501*, 2023.

[8] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6541–6548.

[9] A. Brohan *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on robot learning*, PMLR, 2023, pp. 287–318.

[10] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, *Language models as zero-shot planners: Extracting actionable knowledge for embodied agents*, 2022. arXiv: 2201.07207 [cs.LG].

[11] J. Liang *et al.*, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 9493–9500.

[12] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable task planning," *arXiv preprint arXiv:2307.06135*, 2023.

[13] Z. Hu *et al.*, "Deploying and evaluating llms to program service mobile robots," *IEEE Robotics and Automation Letters*, 2024.

[14] D. Honerkamp, M. Buchner, F. Despinoy, T. Welschehold, and A. Valada, "Language-grounded dynamic scene graphs for interactive object search with mobile manipulation," *arXiv preprint arXiv:2403.08605*, 2024.

[15] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao, "Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning," *arXiv preprint arXiv:2311.17842*, 2023.

[16] F. Liu, K. Fang, P. Abbeel, and S. Levine, "Moka: Open-vocabulary robotic manipulation through mark-based visual prompting," *arXiv preprint arXiv:2403.03174*, 2024.

[17] H. Huang, F. Lin, Y. Hu, S. Wang, and Y. Gao, "Copa: General robotic manipulation through spatial constraints of parts with foundation models," *arXiv preprint arXiv:2403.08248*, 2024.

[18] P. Chang, S. Liu, T. Ji, N. Chakraborty, K. Hong, and K. R. Driggs-Campbell, "A data-efficient visual-audio representation with intuitive fine-tuning for voice-controlled robots," in *Conference on Robot Learning*, 2023.

[19] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4247–4258, 2020.

[20] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, "Zson: Zero-shot object-goal navigation using multimodal goal embeddings," *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 340–32 352, 2022.

[21] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 10 608–10 615.

[22] A. Jagan Sathyamoorthy *et al.*, "Convoi: Context-aware navigation using vision language models in outdoor and indoor environments," *arXiv e-prints*, arXiv–2403, 2024.

[23] S. Liu *et al.*, "Dragon: A dialogue-based robot for assistive navigation with visual language grounding," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3712–3719, 2024.

[24] S. Yenamandra *et al.*, "Homerobot: Open vocabulary mobile manipulation," 2023.

[25] P. Liu, Y. Orru, 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.

[26] A. Stone *et al.*, "Open-world object manipulation using pre-trained vision-language models," *arXiv preprint arXiv:2303.00905*, 2023.

[27] D. Qiu, W. Ma, Z. Pan, H. Xiong, and J. Liang, "Open-vocabulary mobile manipulation in unseen dynamic environments with 3d semantic maps," *arXiv preprint arXiv:2406.18115*, 2024.

[28] P. Zhi *et al.*, "Closed-loop open-vocabulary mobile manipulation with gpt-4v," *arXiv preprint arXiv:2404.10220*, 2024.

[29] N. Cowan, "Working memory underpins cognitive development, learning, and education," *Educational psychology review*, vol. 26, pp. 197–223, 2014.

[30] R. M. Shiffrin, *Search and retrieval processes in long-term memory*. Stanford University, 1968.

[31] J. Schuler, "Integration von förder-und handhabungseinrichtungen," 1987.

[32] P. Dario, E. Guglielmelli, C. Mule, and M. Di Natale, "Urmad: An autonomous mobile robot system for the assistance to the disabled," in *Proc. of the 6th International Conference on Advanced Robotics (ICAR'93)*, 1993, pp. 341–346.

[33] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Coordination and decentralized cooperation of multiple mobile manipulators," *Journal of Robotic Systems*, vol. 13, no. 11, pp. 755–764, 1996.

[34] R. O. Ambrose *et al.*, "Mobile manipulation using nasa's robonaut," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, IEEE, vol. 2, 2004, pp. 2104–2109.

[35] D. Katz *et al.*, "The umass mobile manipulator uman: An experimental platform for autonomous mobile manipulation," in *Workshop on manipulation in human environments at robotics: science and systems*, University of Pennsylvania Philadelphia, PA, USA, 2006.

[36] O. Khatib, "Mobile manipulation: The robotic assistant," *Robotics and Autonomous Systems*, vol. 26, no. 2-3, pp. 175–183, 1999.

[37] J. Bohren *et al.*, "Towards autonomous robotic butlers: Lessons learned with the pr2," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 5568–5575.

[38] M. Ciocarlie, K. Hsiao, A. Leeper, and D. Gossow, "Mobile manipulation through an assistive home robot," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012.

[39] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos, "Nonholonomic navigation and control of cooperating mobile manipulators," *IEEE Transactions on robotics and automation*, vol. 19, no. 1, pp. 53–64, 2003.

[40] O. Brock, O. Khatib, and S. Viji, "Task-consistent obstacle avoidance and motion behavior for mobile manipulation," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, IEEE, vol. 1, 2002, pp. 388–393.

[41] L. Peterson, D. Austin, and D. Kragic, "High-level control of a mobile manipulator for door opening," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, IEEE, vol. 3, 2000, pp. 2333–2338.

[42] H. Seraji, "A unified approach to motion control of mobile manipulators," *The International Journal of Robotics Research*, vol. 17, no. 2, pp. 107–118, 1998.

[43] R. Holmberg and O. Khatib, "Development and control of a holonomic mobile robot for mobile manipulation tasks," *The International Journal of Robotics Research*, vol. 19, no. 11, pp. 1066–1074, 2000.

[44] Q. Huang, K. Tanie, and S. Sugano, "Coordinated motion planning for a mobile manipulator considering stability and manipulation," *The International Journal of Robotics Research*, vol. 19, no. 8, pp. 732–742, 2000.

[45] K. Nagatani, T. Hirayama, A. Gofuku, and Y. Tanaka, "Motion planning for mobile manipulator with keeping manipulability," in *IEEE/RSJ international conference on intelligent robots and systems*, IEEE, vol. 2, 2002, pp. 1663–1668.

[46] G. Oriolo and C. Mongillo, "Motion planning for mobile manipulators along given end-effector paths," in *Proceedings of the 2005 IEEE international conference on robotics and automation*, IEEE, 2005, pp. 2154–2160.

[47] Q. Li, Y. Mu, Y. You, Z. Zhang, and C. Feng, "A hierarchical motion planning for mobile manipulator," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 15, no. 9, pp. 1390–1399, 2020.

[48] T. Sandakalum and M. H. Ang Jr, "Motion planning for mobile manipulators—a systematic review," *Machines*, vol. 10, no. 2, p. 97, 2022.

[49] M. Bajracharya *et al.*, "Demonstrating mobile manipulation in the wild: A metrics-driven approach," *arXiv preprint arXiv:2401.01474*, 2024.

[50] R. Bommasani *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[51] Y. Ding *et al.*, "Integrating action knowledge and llms for task planning and situation handling in open worlds," *Autonomous Robots*, vol. 47, no. 8, pp. 981–997, 2023.

[52] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

[53] I. Singh *et al.*, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 11 523–11 530.

[54] S. Wang *et al.*, "Llmˆ 3: Large language model-based task and motion planning with motion failure reasoning," *arXiv preprint arXiv:2403.11552*, 2024.

[55] S. Nasiriany *et al.*, "Pivot: Iterative visual prompting elicits actionable knowledge for vlms," *arXiv preprint arXiv:2402.07872*, 2024.

[56] H. Wang *et al.*, "Mosaic: A modular system for assistive and interactive cooking," *arXiv preprint arXiv:2402.18796*, 2024.

[57] B. Salomon, M. Garber, M. C. Lin, and D. Manocha, "Interactive navigation in complex environments using path planning," in *Proceedings of the 2003 symposium on Interactive 3D graphics*, 2003, pp. 41–50.

[58] F. Xia *et al.*, "Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 713–720, 2020.

[59] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.

[60] J. Van Den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, "Interactive navigation of multiple agents in crowded environments," in *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, 2008, pp. 139–147.

[61] M. Wang, R. Luo, A. Ö. Önol, and T. Padir, "Affordance-based mobile robot navigation among movable obstacles," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 2734–2740.

[62] C. Li, F. Xia, R. Martin-Martin, and S. Savarese, "Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators," in *Conference on Robot Learning*, PMLR, 2020, pp. 603–616.

[63] K.-H. Zeng, L. Weihs, A. Farhadi, and R. Mottaghi, "Pushing it out of the way: Interactive visual navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9868–9877.

[64] X. Wang, Y. Liu, X. Song, B. Wang, and S. Jiang, "Camp: Causal multi-policy planning for interactive navigation in multi-room scenes," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[65] A. Kirillov *et al.*, "Segment anything," *arXiv:2304.02643*, 2023.

[66] S. Liu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[67] T. Ren *et al.*, *Grounded sam: Assembling open-world models for diverse visual tasks*, 2024. arXiv: 2401.14159 [cs.CV].

[68] K. Košnar, T. Krajník, and L. Přeučil, "Visual topological mapping," in *European Robotics Symposium 2008*, Springer, 2008, pp. 333–342.

[69] R. Wang, H. Li, X. Han, Y. Zhang, and T. Baldwin, "Learning from failure: Integrating negative examples when fine-tuning large language models as agents," *arXiv preprint arXiv:2402.11651*, 2024.

[70] J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao, "Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v," *arXiv preprint arXiv:2310.11441*, 2023.

[71] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[72] W. Huang *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[73] W.-L. Chiang *et al.*, *Chatbot arena: An open platform for evaluating llms by human preference*, 2024. arXiv: 2403.04132 [cs.AI].

[74] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," *arXiv preprint arXiv:1907.01657*, 2019.

[75] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Guided reinforcement learning with learned skills," *arXiv preprint arXiv:2107.10253*, 2021.

[76] L. Ke *et al.*, "Segment anything in high quality," in *NeurIPS*, 2023.

[77] Y. Zhu, A. Lim, P. Stone, and Y. Zhu, "Vision-based manipulation from single human video with open-world object graphs," *arXiv preprint arXiv:2405.20321*, 2024.

[78] S. Dass *et al.*, "Telemoma: A modular and versatile teleoperation system for mobile manipulation," *arXiv preprint arXiv:2403.07869*, 2024.

VII. Appendix

A. Method details

*a) **Perception System**:* We use a strong open-world segmentation model, GSAM, to extract interactable objects from the scene image with pixel-level accuracy. GroundingDINO [66] (`groundingdino_swinb_cogcoor`) provides the bounding box and SAM-HQ [76] (`sam_hq_vit_b`) serves as the segmentation model. We borrow the code from ORION [77].

*b) **Skill Library**:* BUMBLE has a diverse skill library to operate effectively in buildings. For each skill, we provide the description generating the parameters, type of parameters, name & description of the skill provided to the VLM for skill selection, and description of parameter mapping to low-level robotic actions (See Fig. 4 for prompt images).

`GoToLandmark` skill enables the robot to navigate between locations on the same floor using a topological visual map with landmark images as nodes and a 2D occupancy map to generate trajectories. The VLM receives all landmark images with markers and the corresponding floor number. Based on the image semantics, it selects the most suitable landmark for the subtask (including the elevator landmark, if necessary).
Skill parameter(s): Landmark image
Skill description prompt for skill selection:

```
skill_name: goto_landmark
arguments: Selected landmark image from the
    environment from various options.
description: Navigates to the landmark in the
    environment. For instance, bedroom, kitchen,
    tool shop, etc.
```

Parameter to low-level robot commands: The landmark images have a pre-defined mapping to the corresponding 2D pose on the occupancy map, which acts as the target pose. We use a global and local planner to generate a trajectory from the current robot pose to the target pose using the 2D occupancy map and the robot's front 2D lidar.

`NavigateNearObj` skill enables the robot to traverse near an object that is visible to the robot. We use a GSAM query `all objects` to segment objects in the scene. The segmentations are overlayed with markers on the image.
Skill parameter(s): Object segmentation
Skill description prompt for skill selection:

```
skill_name: navigate_to_point_on_ground
arguments: object
description: Moves the robot to a point near the
    selected object. This skill can be used to move
     to a point in the room to perform a task,
    example, navigating near the toaster to make a
    toast.
```

Parameter to low-level robot commands: Object segmentation is used to calculate the object's pose relative to the robot. The nearest point on the ground to the object is converted to a 2D pose on the map, which in turn is used to generate low-level robot base commands.

`MoveBase` skill equips the robot to adjust the base by 30 cm in one of the four directions: forward, left, right, and backward. The pose of the robot base after moving 30 cm in each direction is calculated and overlayed on the scene image.
Skill parameter(s): Direction
Skill description prompt for skill selection:

```
skill_name: move_base
arguments: direction
description: Moves the robot base either forward,
    backward, left, or right by 0.3 meters, w.r.t.
    the camera view. This skill should only be used
     to adjust the base of the robot in the final
    few meters of navigation and not to traverse
    between rooms.
```

Parameter to low-level robot commands: The robot's pose relative to the map frame after moving 30 cm in the selected direction is calculated and used as the target base pose.

`Pickup` skill allows the robot to pick up an object using the left arm. We use a GSAM query `all objects` to segment objects in the scene. The segmentations are overlayed with markers on the image.
Skill parameter(s): Object segmentation
Skill description prompt for skill selection:

```
skill_name: pick_up_object
arguments: object_of_interest
description: pick_up_object skill moves its arms to
     pick up object_of_interest. The pick_up_object
     skill can only pick up objects within arm
    reach and does not control the robot base. The
    robot cannot pick up heavy objects like chairs,
     tables, etc.
```

Parameter to low-level robot commands: The object pose is calculated relative to the robot using the segmentation mask, which then helps determine an approach and a goal pose for the left-arm gripper. Inverse Kinematics is used to find the corresponding joint configuration, which is then achieved via the joint controller.

`PushObjOnGround` skill enables the robot to push a visible object forward, left, or right. We query the VLM twice: a) to select the object to push where the possible options are generated using a GSAM query (`all objects`), and b) to determine the push direction, object position after pushing is approximated and overlayed on the image.
Skill parameter(s): Object segmentation, and direction
Skill description prompt for skill selection:

```
skill_name: push_object_on_ground
arguments: object, direction
description: Pushes an object on the ground one of
    the directions. The robot can push objects that
     are 2-3m away from the robot. The skill
    decides which object and direction to push.
    Example use cases: pushing obstacle to clear
    the pathway for the robot to go forward,
    pushing objects for rearrangement, etc.
```

Parameter to low-level robot commands: The selected object's pose is used to determine the target 2D pose for the robot base via heuristics. The robot then moves to the target pose and executes a predefined arm trajectory based on the chosen direction.

Skill Parameter	BUMBLE	COME	Inner Monologue	BUMBLE w/o CoT [71]	BUMBLE w/o SoM [70]
`Pickup [Obj.]` (5-10 distractors)	80.0	80.0	65.0	80.0	50.0
`Pickup [Obj.]` (20-25 distractors)	65.0	65.0	65.0	60.0	40.0
`PushObjectOnGround [Obj., Dir.]`	81.0	70.4	56.8	64.3	81.8
`CallElevator [Button Location]`	95.0	75.0	60.0	40.0	25.0
Average	**80.2**	72.6	61.7	60.7	49.2

TABLE II

SUCCESS RATE (%) IN PREDICTING SKILL PARAMETERS USING OFFLINESKILLDATASET.

`OpenDoor` enables the robot to open push-doors using the left or right arm. Two markers indicating left or right are overlayed on the image (central left, central right image positions).
Skill parameter(s): Left or Right
Skill description prompt for skill selection:

```
skill_name: open_door
arguments: door_side
description: Opens the door by pushing on a certain
     side of the door (left or right). The robot
    moves forward with its door_side arm out to
    push open the door.
```

Parameter to low-level robot commands: The robot aligns itself with the robot door. It moves the selected arm to a predefined joint configuration and then moves toward the door to push it open using the forearm of the selected arm as the contact point.
`CallElevator` allows the robot to call the elevator to the current floor and move inside the elevator. The buttons are segmented using GSAM query `buttons`.
Skill parameter(s): Button Segmentation
Skill description prompt for skill selection:

```
skill_name: call_elevator
arguments: button position depending whether you
    want to go to a floor above or below.
description: Equips the robot with calling the
    elevator capability. The robot will push the
    button selected in the argument to call the
    elevator in the current floor. The subtask must
     indicate the current floor number and the
    destination floor number to go to. Example: 'Go
     to the second floor from first floor.'
```

Parameter to low-level robot commands: The button pose is calculated using the selected segmentation mask. The button is pressed until reaching a force threshold, measured by the force-torque sensor on the robot's right wrist. After pressing the button, the robot moves inside the elevator to a predefined map pose.
`UseElevator` allows the robot to use the elevator to go to a target floor and then move outside the elevator. The buttons are segmented using GSAM query buttons.
Skill parameter(s): Button Segmentation
Skill description prompt for skill selection:

```
skill_name: use_elevator
arguments: button position of the floor to go to.
description: Equips the robot with using elevator
    capabilities. The robot will push the button
    selected in the argument. This skill is used to
     change the floor of the robot after calling
    the elevator. The subtask must indicate the
    desired floor number to go to.
```

Parameter to low-level robot commands: The button pose is calculated using the selected segmentation mask. The button is pressed until a force threshold is crossed, measured by the force-torque sensor on the robot's right wrist. After pressing the button, the robot moves outside the elevator and localizes itself on the new floor 2D map.

*c) **ROS Packages**:* For these skills, we use the following ROS packages, configured for the Tiago robot: `gmapping` (2D occupancy map generation), `amcl` (Robot localization), `move_base` (to reach the base pose relative to the 2D map). The `move_base` package uses `global_planning` (A^* global path planner), `teb_local_planner` (Time Elastic Band local planner). Finally, we use the joint controller configured for Tiago: `wiki.ros.org/Robots/TIAGo/Tutorials/trajectory_controller`. We borrow the codebase from TeleMoMa [78].

*d) **Memory**:* The short-term memory enables the robot to track state-action history and overcome past failures. These failures, detected by the skills, include the depth sensor returning NaN values for a selected object, failure to find a valid inverse kinematics solution for arm pose, and inability to find a collision-free path due to obstacles like a chair, wet floor signs, or closed doors.

The long-term memory allows the VLM to learn from its past mistakes. These experiences are collected offline and are not updated during evaluations. We collect small examples of 5 instances for each skill and annotate them with ground truth answers. We compare the predicted and ground truth answers, and only the erroneous predictions are retained. This yields 1-3 examples for the `MoveBase`, `UseElevator`, `PushObjOnGround` skill parameters, and for the skill selection.

B. Experimental Evaluations

a) Task Specifications: For each task, we evaluate the three task specifications highlighted below to demonstrate the robustness of our system to diverse user language instructions.
Retrieve diet soda can: 1) "Could you grab me a drink that is low in calories?" 2) "Any chance you can find me a sugar-free soda?" 3) "I want something fizzy to drink, but I am on

diet. Can you help me with that?"
Retrieve colored marker: 1) "I want to color the sky in my drawing. Can you get me a marker?" 2) "I want to color grass in my drawing. Can you get me a marker?" 3) "I need to color some hearts. Can you get a marker for that?"
Rearrange chairs: 1) "Could you make the seating chairs in the reception area more orderly?" 2) "Make the reception area more welcoming by arranging the chairs" 3) "Can you help me arrange the chairs in the reception area?"

b) VLM checkpoints: We demonstrate in Fig. 5 that BUMBLE is a general framework that leverages the rapidly advancing capabilities of VLM. We provide the VLM checkpoint details for reproducibility: GPT4o (*gpt-4o-2024-05-13*), GPT4o-mini (*gpt-4o-mini-2024-07-18*), Pro-1.5-Gemini (*Gemini-1.5-Pro-Exp-0827*), Flash-1.5-Gemini (*Gemini-1.5-Flash-Exp-0827*), Haiku-3-Claude (*claude-3-haiku-20240307*), Sonnet-3-Claude (*claude-3-sonnet-20240229*), Opus-3-Claude (*claude-3-opus-20240229*), Sonnet-3.5-Claude (*claude-3-5-sonnet-20240620*).