MoMaStage: Skill-State Graph Guided Planning and Closed-Loop Execution for Long-Horizon Indoor Mobile Manipulation

Chenxu Li*, Zixuan Chen*, Yetao Li, Jiapeng Xu, Hongyu Ding, Jieqi Shi, Jing Huo, Yang Gao
State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing 210023, China

Abstract*—Indoor mobile manipulation (MoMA) enables robots to translate natural language instructions into physical actions, yet long-horizon execution remains challenging due to cascading errors and limited generalization across diverse environments. Learning-based approaches often fail to maintain logical consistency over extended horizons, while methods relying on explicit scene representations impose rigid structural assumptions that reduce adaptability in dynamic settings. To address these limitations, we propose MoMaStage, a structured vision-language framework for long-horizon MoMA that eliminates the need for explicit scene mapping. MoMaStage grounds a Vision-Language Model (VLM) within a Hierarchical Skill Library and a topology-aware Skill-State Graph, constraining task decomposition and skill composition within a feasible transition space. This structured grounding ensures that generated plans remain logically consistent and topologically valid with respect to the agent's evolving physical state. To enhance robustness, MoMaStage incorporates a closed-loop execution mechanism that monitors proprioceptive feedback and triggers graph-constrained semantic replanning when deviations are detected, maintaining alignment between planned skills and physical outcomes. Extensive experiments in physics-rich simulations and real-world environments demonstrate that MoMaStage outperforms state-of-the-art baselines, achieving substantially higher planning success, reducing token overhead, and significantly improving overall task success rates in long-horizon mobile manipulation. Video demonstrations are available on the project website: https://chenxuli-cxli.github.io/MoMaStage/.

I. INTRODUCTION

Indoor long-horizon mobile manipulation (MoMa) requires robots to execute extended navigation and interaction sequences in complex, dynamic environments like homes and kitchens [39, 6], tightly coupling perception, decision-making, and control amid evolving states. Although imitation learning [42, 9, 41] improves visual robustness and short-horizon dexterity over classical planning-control pipelines, scaling it to long-horizon MoMa remains challenging.

Long-horizon mobile manipulation is bottlenecked by the combinatorial growth of task complexity, which raises expert data collection costs, limits end-to-end policy generalization [4, 36], and exacerbates compounding errors. Prior map-based symbolic planning methods [15, 27, 12, 38] partially address this, but rely on rigid assumptions, limited real-world semantic understanding, and costly explicit mapping.

Recent Vision-Language Models (VLMs) [5, 10, 20] offer a promising alternative by grounding natural language instructions into sequential sub-tasks. However, purely VLM-driven agents often lack structural constraints: open-loop language planners [1, 18, 22] may generate semantically plausible yet physically inconsistent skill sequences. This is because executability depends not only on the current subtask, but also on the accumulated effects of previous skills, which are often ignored by existing methods [34, 7, 17].

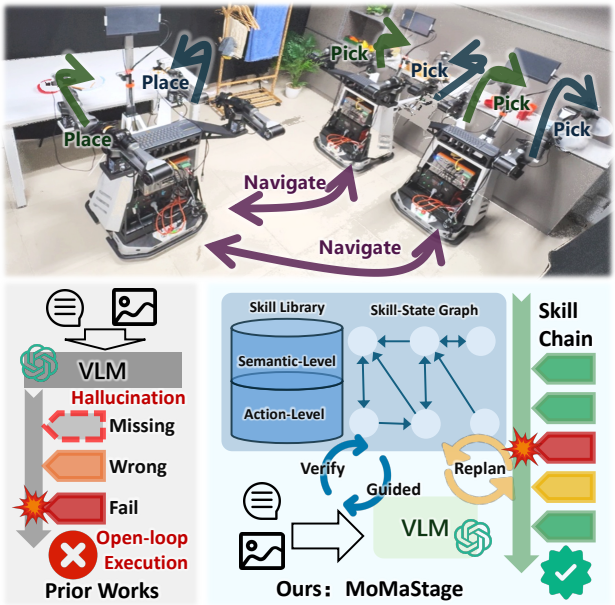


Fig. 1: We propose MoMaStage, a framework for long-horizon mobile manipulation that drives VLMs to translate instructions into valid skill chains via a Skill-State Graph and a hierarchical skill library, with closed-loop proprioceptive verification for guided replanning upon failure.

To address this gap, we propose **MoMaStage**, a structured closed-loop VLM-driven framework for indoor long-horizon mobile manipulation without explicit scene mapping (Figure 1). Our key insight is that the missing abstraction is not a richer explicit scene graph, but a state-grounded skill transition model tracking how each semantic skill changes the robot-task state. Accordingly, MoMaStage constrains VLM reasoning with a Hierarchical Skill Library and a topology-aware Skill-State Graph. Inspired by classical precondition-effect reasoning, we recast open-loop VLM planning as a state-grounded skill transition process, using a lightweight embodiment-state grounding interface to embed evolving physical constraints into the graph without heavy symbolic world modeling.

Robust long-horizon execution requires dynamic adaptability. MoMaStage introduces a closed-loop mechanism that

*Equal Contribution

decouples physical safety from semantic reasoning. High-frequency proprioceptive ego-state monitoring evaluates task progress and ensures immediate safety without inducing inference latency. When anomalies occur, targeted semantic verification [11, 43] triggers VLM-based replanning [34], enabling recovery from out-of-distribution disruptions.

In summary, our contributions are as follows:

- We present MoMaStage, a map-free VLM-driven framework for indoor long-horizon mobile manipulation that unifies instruction understanding, skill-chain generation, execution, and feedback-driven refinement in a closed-loop pipeline.
- We introduce a state-grounded planning and execution mechanism based on a Skill-State Graph and hierarchical skill library, enforcing cumulative state feasibility over VLM-generated skill chains and supporting robust recovery through lightweight ego-state monitoring and targeted semantic replanning.
- We validate MoMaStage through extensive mobile manipulation experiments in complex simulations and dynamic real-world environments, demonstrating improved planning validity, execution robustness, and long-horizon task completion over representative baselines.

II. RELATED WORK

Mobile manipulation and skill learning. Household mobile manipulation has progressed through benchmarks and systems such as BEHAVIOR [35], Mobile ALOHA [14], HomeRobot [39], and OK-Robot [23]. Large-scale VLA and generalist robot policy models, including RT-1/2 [4, 5], VIMA [44], RoboAgent [3], Octo [36], OpenVLA [20], and DP-VLA [17], learn language-conditioned policies. In parallel, imitation and visuomotor learning methods such as Implicit Behavioral Cloning [13], BeT [30], ACT [42], Diffusion Policy [9], UMI [8], and DP3 [41] provide low-level control. Skill-prior methods such as SPiRL [26] improve downstream learning, but long-horizon execution remains sensitive to error propagation and state dependencies across skills.

VLMs for task planning and long-horizon execution. Language models support semantic decomposition [1, 18], reward-driven skill synthesis [40, 24], code or structured-program generation [21, 33], and PDDL planning [22, 38]. Embodied systems such as PaLM-E [10], DeCo [7], VoxPoser [19], and OWMM-Agent [6] connect visual observations to sequential actions, while ReAct [37] combines reasoning and acting in language-based decision processes. SayPlan [27], GRID [25], and VeriGraph [12] add scene-graph grounding, but require explicit spatial representations. In contrast, MoMaStage represents only embodiment variables required to validate skill transitions, avoiding full scene reconstruction.

Closed-loop replanning and state verification. TAMP maintains explicit state through planning and execution [15]; Reflexion [31] and KnowNo [28] use verbal feedback or uncertainty; and RePLan [34], AHA [11], and Code-as-Monitor [43] detect or explain failures. These methods commonly rely on repeated semantic-model queries.

III. METHOD

Figure 2 summarizes three stages: structured skill grounding, graph-guided planning and verification, and closed-loop execution. Appendix A retains the complete definitions and procedural description.

A. Skill and State Representation

The hierarchical library $\mathcal{S} = \mathcal{S}_{semantic} \cup \mathcal{S}_{action}$ separates semantic skills used in task planning (e.g., navigate, pick, place) from fine-grained action primitives used for execution and recovery. We organize semantic skills in a directed Skill-State Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: node v_i denotes skill s_i, and edge (v_i, v_j) permits a local transition.

Adjacency alone cannot capture long-horizon constraints, such as whether a gripper is already occupied. Each skill therefore has a precondition and an effect over a compact embodiment state. For our dual-arm platform,

$$C = (L_{scene}, O_{left}, O_{right}), \tag{1}$$

where L_{scene} is the current location and O_{left}, O_{right} are held objects ($\emptyset$ means empty and "_" is a wildcard). Effects $\Delta = (f_{scene}, f_{left}, f_{right})$ use $MOVE(A, B)$, $ADD(a)$, and $SUB(a)$ to update these variables. This intentionally models only state needed for executability, not a complete scene.

Specifically, $MOVE(A, B)$ changes the base location from A to B; $ADD(a)$ requires an empty gripper and records object a; and $SUB(a)$ requires that a is held and resets the gripper to empty. A null operator leaves the corresponding component unchanged. Recursively applying these effects captures dependencies that adjacency alone misses, such as a later pick becoming invalid after both grippers have been occupied.

B. Planning and Verification

Given instruction I, observation O_{vis}, and the topology-only graph $\mathcal{G}_{topo}$, the VLM produces a candidate path $\hat{\mathrm{s}} = [\hat{s}_1, \ldots, \hat{s}_K]$. Restricting its output vocabulary and transitions reduces invalid decomposition without exposing a large symbolic world model. The full graph then propagates effects from C_0. A plan is accepted only if

$$\forall i, \quad C_{i-1} \models \mathrm{pre}(\hat{s}_i), \qquad C_i = \Delta_i(C_{i-1}). \tag{2}$$

An incompatible location or gripper state rejects the chain and returns the conflict to the planner for graph-constrained revision. Thus topology controls local composition, while deterministic state propagation enforces global consistency.

C. Closed-Loop Execution and Replanning

For each executed skill, high-frequency proprioceptive checks use joint and gripper signals to test the expected effect. Semantic visual verification is invoked when these checks indicate a deviation, avoiding continuous VLM latency. If the observed state C_{obs} disagrees with the expected C_i, MoMaStage retains completed subtasks and replans from C_{obs} to a graph state satisfying the next unfinished goal. Every corrective chain is checked by Eq. (2) before execution. This localized loop handles failures such as missed grasps and blocked navigation without restarting the task.

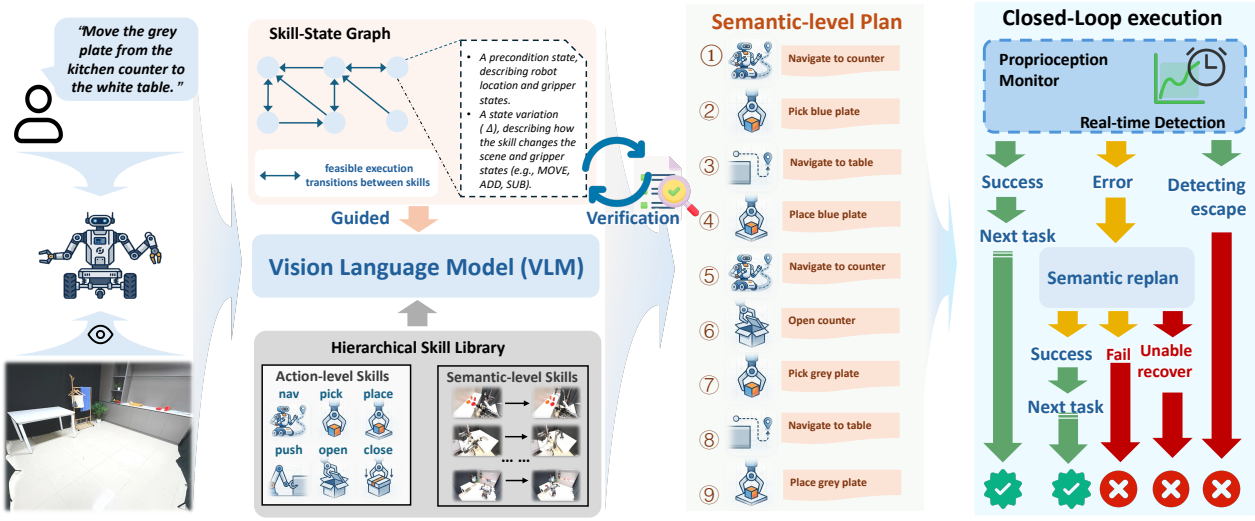


Fig. 2: **MoMaStage overview.** A VLM plans over graph-valid skill transitions; deterministic state propagation verifies the complete chain; and execution monitoring invokes graph-grounded recovery after failures.

IV. EXPERIMENTS

We evaluate whether graph grounding improves planning and whether closed-loop recovery mitigates execution attrition. Appendix B retains the full setup, evaluation questions, interpretation, and additional ablation and failure analyses.

A. Setup

Real world. We deploy on an Agilex Cobot Magic dual-arm mobile robot in a 3×3 m workspace containing cupboard, dining-table, and pantry regions; these labels are descriptive and no map is constructed. Three Orbbec Dabai cameras, including two wrist-mounted cameras and one overhead camera, and an Intel RealSense D435i provide manipulation and navigation views. The graph contains six pick, six place, four navigation, and ten recovery skills. ACT manipulation skills are trained on 50 spatially generalized trajectories; navigation and action-level recovery follow predefined rules. We construct 100 tasks of varying horizon with navigation–manipulation alternation and dual-arm coordination.

Simulation. Our *mshab** extension of ManiSkill-Hab [32] adds VLM task decomposition and API translation, chains heterogeneous skill types, and permits recovery by changing the sequence at the interruption point. We retain the benchmark skill weights: SAC [16] and BC [2] for pick/place, PPO [29] and BC for open/close, and PPO for navigation. The three tasks are *tidy_house* with 20 subtasks, *prepare_groceries* with 12 subtasks, and *set_table* with 16 subtasks, evaluated with BC/RL skill variants.

Baselines and protocol. GT executes a human-authored sequence without retries or replanning. DeCo* adapts DeCo [7] to both domains; unlike map-dependent alternatives, it supports our lightweight setting. End-to-end ACT, trained from 50 complete-task demonstrations, is additionally evaluated on the robot. Gemini 2.5 Flash is the common reasoning engine. Simulation reports three groups of 100 episodes per method, scene, and graph configuration, totaling 8,100 trajectories.

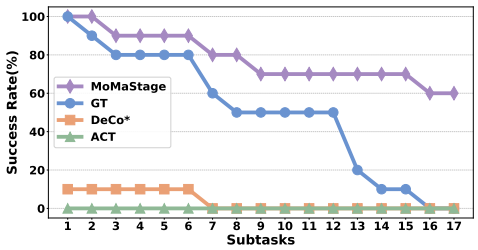
B. Real-World Evaluation

Figure 3 reports ten trials of the hardest 17-subtask setting. End-to-end ACT fails immediately; DeCo* starts 10% of trials but reaches 0% by subtask 7; and open-loop GT reaches 0% by step 16 after unrecovered physical errors. MoMaStage has 100% initial planning success and 60% final completion. The trajectory traces show that this gain comes from localized recovery, including replanning after the subtask-7 failure in Fig. 3c, rather than error-free low-level execution.

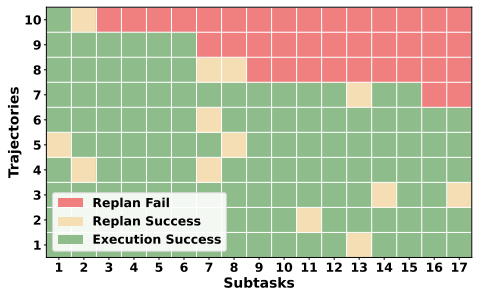
C. Simulation Benchmarking

Table I shows that MoMaStage consistently achieves substantially stronger planning robustness than DeCo* across all scenes and skill configurations. This indicates that grounding VLM planning with the Skill-State Graph effectively constrains long-horizon skill composition and reduces invalid action chains. The comparison across skill configurations also suggests that planning success is not determined by the planner alone, but by the interaction between structured high-level constraints and the available low-level skill vocabulary.

For long-horizon execution, Table II reports subtask survival before the terminal checkpoints of each task. We omit the final checkpoint column for each task because terminal survival is nearly saturated at zero for most methods, making those columns less informative for comparing intermediate execution robustness. The remaining phase-wise results still show severe physical attrition for all methods, confirming that simulation performance is dominated by accumulated low-level execution failures. MoMaStage can trail GT in early phases because its first phase includes planning failures, but it often becomes competitive or stronger in later phases by recovering from execution anomalies. In contrast, the real-robot results demonstrate that semantic replanning is most valuable when the underlying skills can reliably execute the proposed correction.



(a) Success rates for methods.



(b) Detailed execution states of 10 trajectories.

(c) Qualitative visualization of a successful trajectory execution, highlighting the replanning capability to recover from a failure during subtask 7.

Fig. 3: Real-world evaluation. (a) Survival over 17 subtasks. (b) Ten MoMaStage trajectories and recovery events. (c) A successful trajectory replanned after a failure at subtask 7.

TABLE I: Simulation planning success (%, mean ± standard deviation).

Methods	Tidy_House			Prepare_Groceries			Set_Table		
	BC	RL	RL_Per_Obj	BC	RL	RL_Per_Obj	BC	RL	RL_Per_Obj
DeCo*	44.00±3.74	46.67±4.50	37.00±4.55	35.00±3.56	26.67±3.68	25.67±0.94	23.67±2.49	20.33±1.70	22.00±2.16
MoMaStage	**80.67**±2.87	**85.67**±3.09	**79.67**±4.11	**90.33**±2.05	**85.67**±4.11	**91.67**±2.62	**93.67**±1.25	**87.00**±5.10	**91.00**±2.16

TABLE II: Long-horizon subtask survival before terminal checkpoints (%, mean ± standard deviation).

Skill-State Graph Type	Methods	Tidy_House				Prepare_Groceries		Set_Table
		Phase 1 (Step 4)	Phase 2 (Step 8)	Phase 3 (Step 12)	Phase 4 (Step 16)	Phase 1 (Step 4)	Phase 2 (Step 8)	Phase 1 (Step 8)
RL_ALL (SAC+PPO)	GT	34.7 ±2.1	10.0 ±0.8	2.3 ±0.5	0.3 ±0.5	19.7 ±2.5	3.3 ±0.5	22.0 ±1.6
	DeCo*	13.7 ±2.1	0.7 ±0.5	0.0 ±0.0	0.0 ±0.0	6.3 ±1.3	0.0 ±0.0	4.0 ±1.4
	MoMaStage	27.0 ±1.4	**12.3** ±1.9	**4.0** ±0.0	**1.7** ±0.9	17.7 ±2.5	**3.7** ±1.7	19.0 ±0.0
IL_ALL + RL_ALL (BC+PPO)	GT	38.7 ±2.9	9.0 ±0.0	3.0 ±0.8	0.3 ±0.5	14.0 ±1.6	1.0 ±0.8	17.3 ±1.7
	DeCo*	18.3 ±0.5	4.3 ±1.2	0.3 ±0.5	0.0 ±0.0	3.0 ±0.8	0.0 ±0.0	2.7 ±0.5
	MoMaStage	33.3 ±2.6	7.0 ±1.4	2.3 ±0.9	**1.0** ±0.8	11.3 ±3.4	**1.0** ±0.8	**23.0** ±1.4
RL_Per_Obj (SAC+PPO)	GT	41.3 ±2.9	15.0 ±2.2	3.6 ±0.9	0.7 ±0.9	27.0 ±0.8	2.3 ±1.3	25.3 ±2.1
	DeCo*	11.7 ±0.5	4.0 ±0.8	0.3 ±0.5	0.0 ±0.0	6.7 ±0.9	0.3 ±0.5	7.0 ±1.4
	MoMaStage	31.7 ±2.6	10.0 ±2.2	**4.7** ±0.5	**1.0** ±0.8	20.0 ±1.6	**2.3** ±0.5	**27.3** ±3.8

D. Discussion and Limitations

The results indicate that explicit graph grounding mainly improves semantic validity and recovery consistency, while final long-horizon reliability is still bounded by low-level physical execution. Additional planning-efficiency ablations and terminal failure-mode statistics are provided in Appendix B-H. The graph assumes that compact executability variables can be identified; richer relations may require additional state factors. The real-world study uses one workspace, and recovery still requires a suitable primitive and reachable physical state.

V. CONCLUSION

In this work, we present MoMaStage, a map-free, vision-language-driven framework for long-horizon indoor mobile manipulation. By introducing a state-grounded Skill-State Graph and a graph-constrained closed-loop execution mechanism, MoMaStage improves the logical validity and execution consistency of long-horizon skill chains for robust, real-time adaptation. Extensive real-world and simulated evaluations demonstrate that MoMaStage greatly reduces token costs, achieves substantially higher initial planning success, and significantly improves overall execution reliability, suggesting that explicitly modeling embodiment-state transitions is an effective way to bridge high-level semantic reasoning and long-horizon robot execution. Finally, as high-level reasoning is no longer the primary bottleneck, future work will focus on enhancing the physical dexterity of foundational action-level skills to bridge the gap between logical planning and robust physical embodiment.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine intelligence 15*, pages 103–129, 1995.

[3] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4788–4795. IEEE, 2024.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. *URL https://arxiv.org/abs/2307.15818*, 1:2, 2024.

[6] Junting Chen, Haotian Liang, Lingxiao Du, Weiyun Wang, Mengkang Hu, Yao Mu, Wenhai Wang, Jifeng Dai, Ping Luo, Wenqi Shao, et al. Owmm-agent: Open world mobile manipulation with multi-modal agentic data synthesis. *arXiv preprint arXiv:2506.04217*, 2025.

[7] Zixuan Chen, Junhui Yin, Yangtao Chen, Jing Huo, Pinzhuo Tian, Jieqi Shi, Yiwen Hou, Yinchuan Li, and Yang Gao. Deco: Task decomposition and skill composition for zero-shot generalization in long-horizon 3d manipulation. *IEEE Robotics and Automation Letters*, 2026.

[8] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[10] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[11] Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024.

[12] Daniel Ekpo, Mara Levy, Saksham Suri, Chuong Huynh, and Abhinav Shrivastava. Verigraph: Scene graphs for execution verifiable robot planning. *arXiv preprint arXiv:2411.10446*, 2024.

[13] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pages 158–168. PMLR, 2022.

[14] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[15] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293, 2021.

[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[17] ByungOk Han, Jaehong Kim, and Jinhyeok Jang. A dual process vla: Efficient robotic manipulation leveraging vlm. *arXiv preprint arXiv:2410.15549*, 2024.

[18] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[19] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[20] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[21] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[22] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

[23] Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-

robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

[24] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[25] Zhe Ni, Xiaoxin Deng, Cong Tai, Xinyue Zhu, Qinghongbing Xie, Weihang Huang, Xiang Wu, and Long Zeng. Grid: Scene-graph-based instruction-driven robotic task planning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13765–13772. IEEE, 2024.

[26] Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pages 188–204. PMLR, 2021.

[27] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. *arXiv preprint arXiv:2307.06135*, 2023.

[28] Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*, 2023.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[30] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[31] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36: 8634–8652, 2023.

[32] Arth Shukla, Stone Tao, and Hao Su. Maniskill-hab: A benchmark for low-level manipulation in home rearrangement tasks. *arXiv preprint arXiv:2412.13211*, 2024.

[33] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

[34] Marta Skreta, Zihan Zhou, Jia Lin Yuan, Kourosh Darvish, Alán Aspuru-Guzik, and Animesh Garg. Replan: Robotic replanning with perception and language models. *arXiv preprint arXiv:2401.04157*, 2024.

[35] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on robot learning*, pages 477–490. PMLR, 2022.

[36] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[37] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[38] Haoming Ye, Yunxiao Xiao, Cewu Lu, and Panpan Cai. Uniplan: Vision-language task planning for mobile manipulation with unified pddl formulation. *arXiv preprint arXiv:2602.08537*, 2026.

[39] Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023.

[40] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[41] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

[42] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[43] Enshen Zhou, Qi Su, Cheng Chi, Zhizheng Zhang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and He Wang. Code-as-monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6919–6929, 2025.

[44] Y Zhu et al. Vima: General robot manipulation with multimodal prompts. In *International Conference on Learning Representations (ICLR)*, 2023.

APPENDIX A
METHOD DETAILS

A. Framework Modules

The complete design consists of three modules. **Structured Skill Grounding** constructs a Hierarchical Skill Library and a Skill-State Graph to model skill preconditions and transitions, providing a formal grounding space that prevents logically invalid composition. **Graph-Constrained Planning and Verification** conditions the VLM on the Skill Graph and then performs a post-hoc Skill-State Graph check for global consistency. **Closed-Loop Execution and Replanning** continuously monitors ego-state transitions and triggers graph-grounded recovery when outcomes deviate from expectations.

The original framework description is retained here: given multi-modal inputs, the system integrates graph-constrained planning with closed-loop execution. (a) The VLM-based planner decomposes long-horizon instructions into semantic skill sequences, restricted by the topological constraints of the Skill Graph. (b) A post-hoc feasibility check is performed using the Skill-State Graph to ensure global state consistency. (c) During execution, the system monitors ego-state transitions and triggers graph-grounded replanning to autonomously recover from failures such as grasping or navigation errors.

B. Hierarchical Skill Library

The library $\mathcal{S}$ has two levels of granularity. **Action-level skills** $\mathcal{S}_{action}$ are fine-grained primitives such as joint-level control and basic motion. They are decoupled from scene semantics and serve as foundational execution units without predefined execution orders. **Semantic-level skills** $\mathcal{S}_{semantic}$ are coupled to scene semantics and have explicit contextual preconditions and sequential relationships. A semantic skill may be instantiated through end-to-end training, sequential composition of action primitives, or post-training refinement.

C. Skill-State Graph and State Operations

The Skill-State Graph is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node v_i represents a semantic skill s_i, and a directed edge (v_i, v_j) denotes a feasible local transition. A topological graph is insufficient because skill executability depends on cumulative embodiment state; a pick is invalid when the relevant gripper is already occupied even if the graph contains an adjacent pick edge.

Each node is augmented with a precondition state and state variation. For the dual-arm platform,

$$\Delta = (f_{scene}, f_{left}, f_{right}). \tag{3}$$

L_{scene} is the robot location; O_{left} and O_{right} are held objects, with $\emptyset$ for empty and "_" as a wildcard. Navigation applies $MOVE(A, B)$ or leaves the location unchanged. Manipulation applies $ADD(a)$ to change an empty gripper to object a, or $SUB(a)$ to change a gripper holding a back to empty; a null operation leaves state unchanged.

Recursively applying Δ from the initial state verifies global consistency along an edge sequence. The graph is not intended to be a full symbolic world model. It is a lightweight execution-consistency interface between high-level VLM reasoning and embodied execution, retaining only variables critical to long-horizon executability.

D. Topology-Aware Planning and State Verification

Given instruction I and visual observation O_{vis}, the objective is to produce a semantic sequence $\mathbf{s} = [s_1, \ldots, s_T]$ that is task-consistent while satisfying graph connectivity and cumulative state feasibility. The two-stage procedure consists of topology-aware semantic planning followed by state-driven logical verification.

The reasoning engine receives I, O_{vis}, and a topological subgraph $\mathcal{G}_{topo} \subset \mathcal{G}$. $\mathcal{G}_{topo}$ exposes only skill nodes and adjacency, omitting C and Δ. With explicit 3D scene models and symbolic object descriptions withheld, the VLM relies on environmental affordances and robot capabilities encoded by graph connectivity to generate a topologically valid candidate sequence $\hat{\mathbf{s}}$.

The full Skill-State Graph then evaluates the cumulative transition chain. Starting from C_0, the verifier applies each Δ_i and checks the next skill's location and gripper preconditions. A conflict, such as requesting a pick with an occupied gripper, rejects the candidate and triggers VLM re-decomposition or adjustment. The final sequence is therefore both topologically and logically sound.

E. State-Aware Monitoring and Dynamic Replanning

Execution uses two monitoring layers. **Ego-state monitoring** tracks proprioceptive data, including joint encoders and gripper tactile signals, to confirm primitive outcomes such as successful ADD or SUB operations. **Semantic verification** uses the VLM to check whether the scene matches the expected state variation. If the observed state satisfies the expected outcome, execution advances; otherwise a deviation is flagged.

After a pick failure, blocked navigation path, or other deviation, MoMaStage does not restart the complete task. The observed state C_{obs} becomes the new initial state. A localized search or VLM query identifies a corrective path in $\mathcal{G}$ from C_{obs} to a state satisfying the preconditions of the unfinished subtasks. Constraining recovery to the Skill-State Graph preserves logical consistency and physical embodiment constraints throughout autonomous failure recovery.

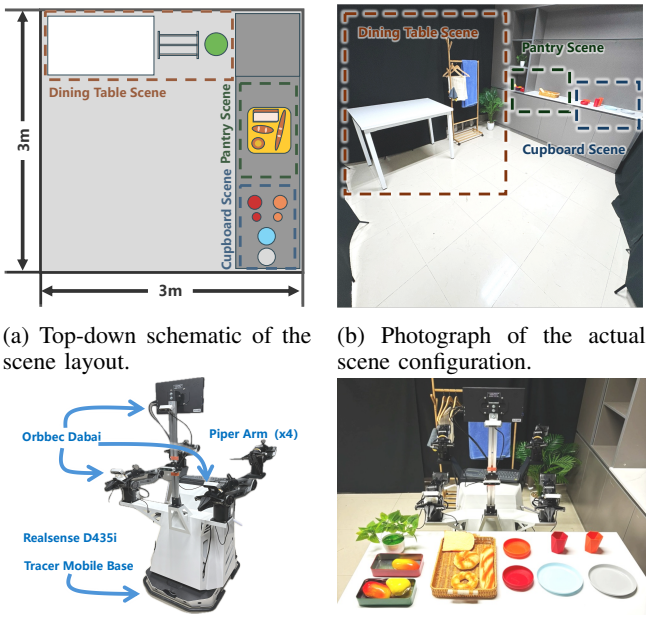
APPENDIX B
ADDITIONAL EXPERIMENTAL MATERIAL

A. Real-World Setup Illustration

Figure 4 complements the experimental protocol in the main paper with the top-down layout, physical workspace, robot hardware, and manipulated objects.

B. Evaluation Questions

The original evaluation addresses three questions. First, can MoMaStage be deployed in real-world long-horizon mobile manipulation, and does closed-loop replanning mitigate compounding execution errors relative to existing baselines?



(a) Top-down schematic of the scene layout.

(b) Photograph of the actual scene configuration.

(c) Hardware configuration of the robotic platform.

(d) Objects and furniture involved in the experiments.

Fig. 4: Overview of the real-world experimental setup.

Second, how robust is planning across diverse simulated environments, and how do low-level skill configurations affect long-horizon success? Third, how does explicit topological grounding contribute to planning efficiency, and what do failures reveal about the boundary between semantic reasoning and physical execution?

C. Additional Platform and Protocol Details

The Cobot Magic platform contains four Agilex Piper arms and a Tracer mobile base. Three Orbbec Dabai cameras—two wrist-mounted and one overhead—serve manipulation, while an Intel RealSense D435i supports both manipulation and navigation. The 100 real-world tasks vary in horizon and repeatedly alternate navigation and manipulation while requiring dual-arm coordination. The evaluation is designed to test whether state-grounded planning and replanning remain consistent over extended execution, rather than to benchmark scene diversity. GT sequencing is an execution upper bound that isolates low-level physical errors from high-level planning errors. DeCo* is the primary planning baseline because it offers recent VLM-driven skill decomposition without requiring a heavy 3D scene representation. VoxPoser [19] and SayPlan [27] depend on explicit spatial maps or benchmark-specific infrastructure, making direct comparison less aligned with our map-free setting. The real-world Skill-State Graph contains 16 semantic skills, including six pick, six place, and four navigation skills, together with ten action-level recovery skills. Semantic manipulation skills use ACT trained on 50 spatially generalized demonstrations, while navigation and recovery primitives use predefined rules. End-to-End ACT is trained from 50 complete-task expert demonstrations.

D. Complete Simulation Protocol

*mshab** extends ManiSkill-Hab [32] over *tidy_house* with four stages and 20 subtasks, *prepare_groceries* with three stages and 12 subtasks, and *set_table* with two stages and 16 subtasks. Original open-source weights are retained: SAC [16] and BC [2] for pick/place, PPO [29] and BC for open/close, and PPO for navigation. The benchmark extension makes three changes: (1) it adds a VLM interface for task decomposition and skill composition, task generation, and translation from planned skills to simulation APIs; (2) it enables seamless chaining of heterogeneous skill types instead of restricting an episode to one skill type; and (3) it supports dynamic recovery by changing the skill sequence directly at the interruption point. Gemini 2.5 Flash is the reasoning engine. GT, DeCo*, and MoMaStage are evaluated across three scenes and three graph configurations, with 100 episodes in each of three independent trial groups, totaling 8,100 trajectories.

E. Detailed Real-World Interpretation

The original description of the main real-world figure is retained here: it provides a quantitative and qualitative evaluation of MoMaStage on the real-world robot platform. Panel (a) shows that MoMaStage maintains a higher success rate over long-horizon subtasks than the baselines. Panel (b) provides trajectory-level execution details demonstrating robustness through successful replanning. Panel (c) gives a step-by-step long-horizon execution and illustrates adaptive replanning after an unexpected failure. The main paper reports ten trials of the hardest 17-subtask setting, involving cross-scene navigation and dual-arm manipulation; the original draft designates shorter-horizon results as supplementary material. Initial planning failures immediately terminate an episode to expose compounding errors. End-to-End ACT fails immediately because a non-modular imitation policy suffers covariate shift. DeCo* initiates only 10% of trials and reaches 0% by subtask 7 because it lacks state verification and recovery. Even perfect GT planning reaches 0% by step 16 because open-loop execution cannot recover from physical anomalies such as grasp slips. MoMaStage achieves perfect initial planning and 60% cumulative final success. The ten trajectory traces show that this survival rate comes from active error correction rather than flawless execution. After physical anomalies, the replanner synthesizes localized recovery actions; failures remain only for severe states such as objects dropping outside the reachable workspace. The qualitative trajectory highlights recovery at subtask 7 and subsequent completion of the 17-subtask task.

F. Detailed Simulation Interpretation

Across scenes and low-level policies, DeCo* degrades strongly on multi-step reasoning, particularly on *set_table*, whereas MoMaStage maintains 79–94% planning success. Grounding the VLM in the Skill-State Graph removes many logical hallucinations and produces valid sequences independent of the selected low-level policy. All methods exhibit strong attrition over long physical horizons. MoMaStage can

initially trail GT because early performance includes planning failures, but open-loop GT degrades rapidly under physical anomalies while MoMaStage recovers and can surpass GT in middle and late phases. RL_Per_Obj illustrates a design trade-off: object-specific skills increase vocabulary size and planning burden, but their stronger physical robustness can compensate later. This motivates balancing skill-library size against the competence of each skill.

G. Simulation Failure Interpretation

The simulation results expose a gap between correct semantic recovery and executable physical recovery. Even GT sequencing approaches zero success at the longest horizons because mesh clipping, force-limit events, dropped objects, and out-of-distribution controller states repeatedly prevent low-level completion. The reasoning engine can detect an anomaly and propose a valid corrective skill chain, but the same simulated policy may fail again when executing it. The stronger real-robot recovery result therefore supports the planning mechanism while also showing that current simulation skill quality can understate the benefit of semantic replanning.

H. Additional Ablation and Failure Analysis

TABLE III: Planning performance and computational cost comparison. Metrics are reported as mean ± standard deviation across all trials.

Methods	Succ. (%) ↑	Time (s) ↓	Tokens ↓	Think. Tok. ↓
DeCo*	0.0 ±0.0	21.4 ±5.7	5814 ±436	3853 ±581
MoMaStage*	100.0 ±0.0	24.6 ±4.2	9106 ±678	2927 ±421
MoMaStage	**100.0** ±0.0	**16.0** ±1.1	**6633** ±218	**2485** ±230

TABLE IV: Failure analysis. FLE, TLE, and PTF denote Force Limit Exceeded, Time Limit Exceeded, and Preceding Task Failed, respectively.

Scene	**Graph**	**FLE(%)**	**TLE(%)**	**PTF(%)**
Tidy_House	BC	83.7 ±2.5	16.3 ±2.5	0.0 ±0.0
	RL	81.0 ±3.0	19.0 ±3.0	0.0 ±0.0
	RL_Per_Obj	80.0 ±5.0	20.0 ±5.0	0.0 ±0.0
Prepare_Groceries	BC	52.0 ±5.6	48.0 ±5.6	0.0 ±0.0
	RL	59.3 ±4.5	40.7 ±4.5	0.0 ±0.0
	RL_Per_Obj	52.3 ±5.0	47.7 ±5.0	0.0 ±0.0
Set_Table	BC	72.7 ±3.5	27.3 ±3.5	0.0 ±0.0
	RL	70.7 ±1.5	29.3 ±1.5	0.0 ±0.0
	RL_Per_Obj	70.7 ±4.7	29.3 ±4.7	0.0 ±0.0

Table III evaluates planning efficiency using Gemini 2.5 Flash. DeCo* completely fails with 0% planning success while still consuming substantial resources, including 3853 thinking tokens on average. This suggests that without explicit topological constraints, the VLM spends computation exploring invalid or inconsistent reasoning paths. Both MoMaStage variants achieve 100% planning success. However, MoMaStage* supplies the VLM with the entire unpruned Skill-State Graph, introducing redundant complexity. In contrast, the standard MoMaStage extracts a focused Skill Graph according to state-driven constraints. This pruning reduces inference time from 24.6 s to 16.0 s, total tokens from 9106 to 6633, and thinking tokens from 2927 to 2485. These results show that graph pruning preserves planning validity while reducing latency and API cost. Table IV categorizes terminal execution failures into Force Limit Exceeded (FLE), Time Limit Exceeded (TLE), and Preceding Task Failed (PTF). FLE is the dominant failure mode in most settings and typically arises from simulation artifacts such as mesh clipping or unstable physics during contact-rich manipulation. Since these failures originate from physically invalid simulated states rather than incorrect semantic planning, they are often unrecoverable by a high-level planner alone. TLE accounts for the remaining failures and usually occurs when a low-level skill exceeds its time bound. Log inspection shows that this often happens after an object drops or when the policy enters an out-of-distribution state. For example, if an object slips during navigation and no localized recovery primitive can restore it, the policy may repeatedly attempt to reach an unmanipulable target until timeout. PTF remains 0.0% across all scenes and graph configurations. This indicates that MoMaStage does not inadvertently undo completed subtasks during recovery. The result supports the monotonicity of the state-verification mechanism and shows that graph-grounded replanning improves logical consistency, even though final long-horizon reliability remains bounded by low-level physical skill competence.