MoTo: A Zero-shot Plug-in Interaction-aware Navigation for General Mobile Manipulation

Zhenyu Wu[1,3*], **Angyuan Ma**[2,3,4*], **Xiuwei Xu**[2,3,4‡], **Hang Yin**[2,3,4]
Yinan Liang[2,3,4], **Ziwei Wang**[5], **Jiwen Lu**[2,3,4], **Haibin Yan**[1†]
[1]School of IEA, Beijing University of Posts and Telecommunications
[2]Department of Automation, Tsinghua University
[3]Beijing Key Laboratory of Embodied Intelligence Systems
[4]Beijing National Research Center for Information Science and Technology
[5]School of Electrical and Electronic Engineering, Nanyang Technological University

Abstract: Mobile manipulation is the fundamental challenge for robotics in assisting humans with diverse tasks and environments in everyday life. Conventional mobile manipulation approaches often struggle to generalize across different tasks and environments due to the lack of large-scale training. However, recent advances in manipulation foundation models demonstrate impressive generalization capability on a wide range of fixed-base manipulation tasks, which are still limited to a fixed setting. Therefore, we devise a plug-in module named MoTo, which can be combined with any off-the-shelf manipulation foundation model to empower them with mobile manipulation ability. Specifically, we propose an interaction-aware navigation policy to generate robot docking points for generalized mobile manipulation. To enable zero-shot ability, we propose an interaction keypoints framework via vision-language models (VLM) under multi-view consistency for both target object and robotic arm following instructions, where fixed-base manipulation foundation models can be employed. We further propose motion planning objectives for the mobile base and robot arm, which minimize the distance between the two keypoints and maintain the physical feasibility of trajectories. In this way, MoTo guides the robot to move to the docking points where fixed-base manipulation can be successfully performed, and leverages VLM generation and trajectory optimization to achieve mobile manipulation in a zero-shot manner, without any requirement on mobile manipulation expert data. Extensive experimental results on OVMM and real-world demonstrate that MoTo achieves success rates of 2.68% and 16.67% higher than the state-of-the-art mobile manipulation methods, respectively, without requiring additional training data.

Keywords: Mobile Manipulation, Vision Language Action Models

1 Introduction

Mobile manipulation empowers robots with the ability to perform complex manipulation tasks across large spaces, which requires controlling both the mobile base and the robotic arm [1, 2, 3]. With the increasing popularity of intelligent robotic systems, several fields such as household service [4, 5], manufacturing [6], and logistics [7] are in urgent demand of mobile manipulation capabilities since robots need to perform cross-space manipulations autonomously. However, the requirements to perform diverse tasks in unstructured environments (e.g., assisting humans in their daily lives) present significant challenges. Therefore, designing a general mobile manipulation framework for real-world deployments is desirable.

Existing mobile manipulation approaches [8, 9] propose end-to-end frameworks to jointly search the navigation and manipulation action spaces to improve performance. However, the end-to-end

*Equal contribution. ‡Project Leader. †Corresponding author: Haibin Yan (eyanhaibin@bupt.edu.cn).

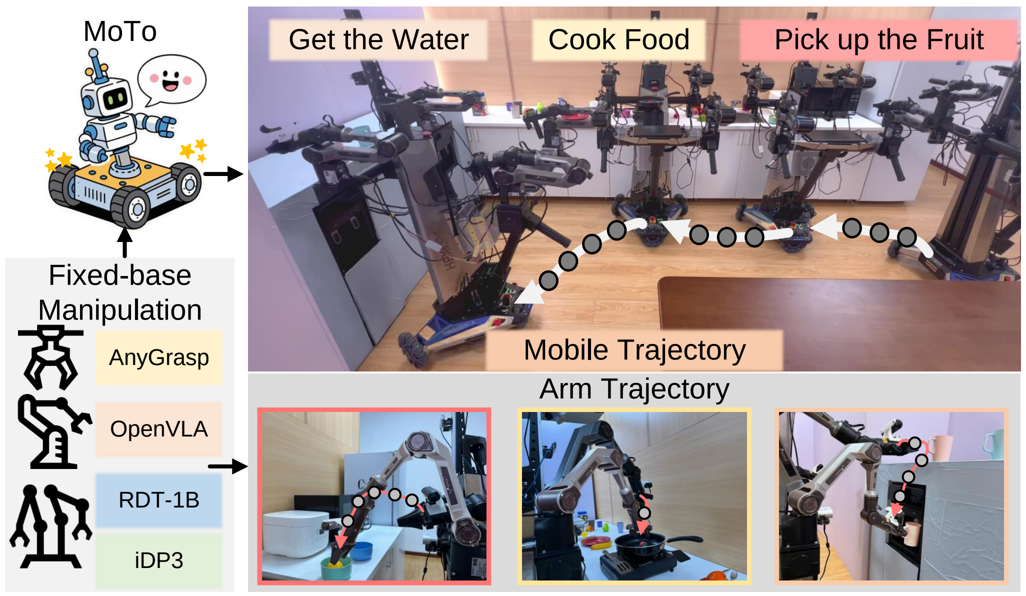

Figure 1: MoTo can be plugged into any fixed-base manipulation model and transferred to mobile manipulation tasks in a zero-shot manner, enabling generalized mobile manipulation.

mobile manipulation policy models lack training on large-scale mobile manipulation datasets due to extremely expensive collection costs for mobile manipulation demonstration, which results in low generalization across tasks and environments. On the other side, the emerging manipulation foundation models [10, 11, 12] have demonstrated high generalization capabilities across different manipulation tasks and diverse deployment scenes. However, these models are not able to predict the movement of the base and thus are limited to fixed-base manipulation tasks. A natural idea is to build a mobile manipulation framework by directly patching together the LLM or VLM with the manipulation foundation models. Conventional mobile manipulation frameworks employ navigation [13, 14] and fixed-base manipulation [15, 16] modules separately to accomplish mobile manipulation tasks. However, naive combining navigation and manipulation results in compounding errors since the large gap between the goals of navigation and manipulation [17]. Finding suitable navigation docking points for interaction remains a huge challenge in mobile manipulation tasks.

In this paper, we propose to solve the problem of mobile manipulation with an interaction-aware navigation policy, namely **Mo**ve and **To**uch (MoTo). MoTo is a zero-shot plug-in that can be combined with any off-the-shelf fixed-base manipulation model and empower it with mobile manipulation ability. The proposed interaction-aware navigation policy generates suitable base docking points for the robot to successfully perform manipulation, which are inspired by the fact that the robot can perform subsequent fixed-base manipulation actions on the docking points where its arms can reach the target object interaction region. We prompt the vision-language model (VLM) to generate task-related keypoints on the target object and robotic arm to enable zero-shot mobile manipulation. We can control the robot by solving an optimization problem that minimizes the distance between the two keypoints and considers several additional constraints, including collision avoidance, smoothness, and arm margin. Figure 1 illustrates the proposed approaches. MoTo efficiently transfers manipulation foundation models to diverse mobile manipulation tasks, where the mobile base and the robot arm coordinately perform actions with physically feasible trajectories. Extensive experimental results on OVMM [18] and real-world demonstrate that MoTo achieves 2.68% and 16.67% higher success rates compared to state-of-the-art mobile manipulation techniques, respectively.

2 Related Works

Mobile Manipulation Frameworks: Existing mobile manipulation frameworks can be categorized into end-to-end and modular based on the model structure. The end-to-end approaches [19, 20, 21, 22] directly map visual observations to the mobile manipulation action space. M^{2}Diffuser [23] guides the diffusion process with various energy terms to ensure that the trajectory of the mobile manipulation follows the physical constraints. However, end-to-end approaches require extensive training data leading to high costs [24]. The modular mobile manipulation framework [18, 25, 26] benefits from off-the-shelf navigation and manipulation frameworks exhibiting outstanding data efficiency. IALP [27] leverages LLM reasoning capability to generate manipulation actions in a closed-loop interactive manner. MoManipVLA [3] achieves data-efficient generalized mobile manipulation via optimization of the base waypoints to ensure that the VLA prediction trajectories are feasible. AMR [28] and LIMP [29] leverage language instructions to guide robots to target poses, demonstrating the potential of vision-language navigation for mobile manipulation. However, naively combining navigation and manipulation complicates instructions and leads to compounding errors, as humans may not know optimal base positions for manipulation.

Trajectory Optimization: Trajectory optimization is important for robots to perform precise manipulation actions. Early research [30, 31, 32] due to the hand-crafted cost terms restricts deployment in diverse and complex environments. To improve scalability, researchers focus on data-driven learnable methods. Contact-GraspNet [33] and O2O-Afford [34] generate trajectories with learned grasping poses and object affordances, respectively. Diffusion policy-based methods [35, 36] learn collision-free trajectories directly from expert episodes and achieve promising performance. More recently, foundation-model–based frameworks like VoxPoser [37] and ReKep [38] leverage pre-trained priors to infer physical constraints, which significantly improve generalization. Inspired by ReKep, we propose a multi-view voting strategy to generate scene-level interaction keypoints to fine-grain guide mobile manipulation trajectory generation.

3 Problem Statement

Our goal is to enable robots to perform long-horizon mobile manipulation tasks with strong generalization ability to unseen environments and goals. The task is described by a free-form language instruction $\mathcal{T}$. For example, "I need to drink water", the robot should first pick up a bottle, put the bottle on a water dispenser, add water to the bottle, and place the bottle on a table near the human.

We consider a wheeled robot equipped with single or dual arms and RGB-D cameras. At each time step t, the robot receives RGB-D observations $(s_t^e, \{s_t^w\})$, and executes actions $(a_t^{base}, \{a_t^{arm}\})$, where s_t^e is the observation from the exterior camera which captures the scene in front of the robot, $\{s_t^w\}$ is the observation from wrist camera, a_t^{base} is a 2-DoF robot base movement and $\{a_t^{arm}\}$ is 6-DoF arm pose and gripper status for each robotic arm.

Note that this work focuses on a general plug-in framework for mobile manipulation. We assume off-the-shelf fixed-base manipulation policies are available, which can complete low-level language instruction $\mathcal{T}_{fix}$. Here, the task specified in $\mathcal{T}_{fix}$ must be completed with the fixed robotic base.

This work addresses the problem of determining base docking points from which the robotic arm can reach and manipulate the target object successfully.

$$\begin{aligned} &\min \sum_{t=0}^{T} \mathcal{O}(\theta_t^{base}, \{\theta_t^{arm}\}) \\ &\text{s.t.} \quad \theta_t^{base} \in \mathbb{R}^3, \quad \theta_t^{arm} \in \mathbb{R} \end{aligned} \tag{1}$$

This equation represents the overall optimization objective $\mathcal{O}$ for the mobile manipulation trajectory.

With the fast development of manipulation foundation models [37, 11, 12, 38], we believe this assumption is reasonable and feasible. And what we should accomplish is a general plug-in that can be combined with any off-the-shelf fixed-base manipulation model and empower it with mobile manipulation ability.

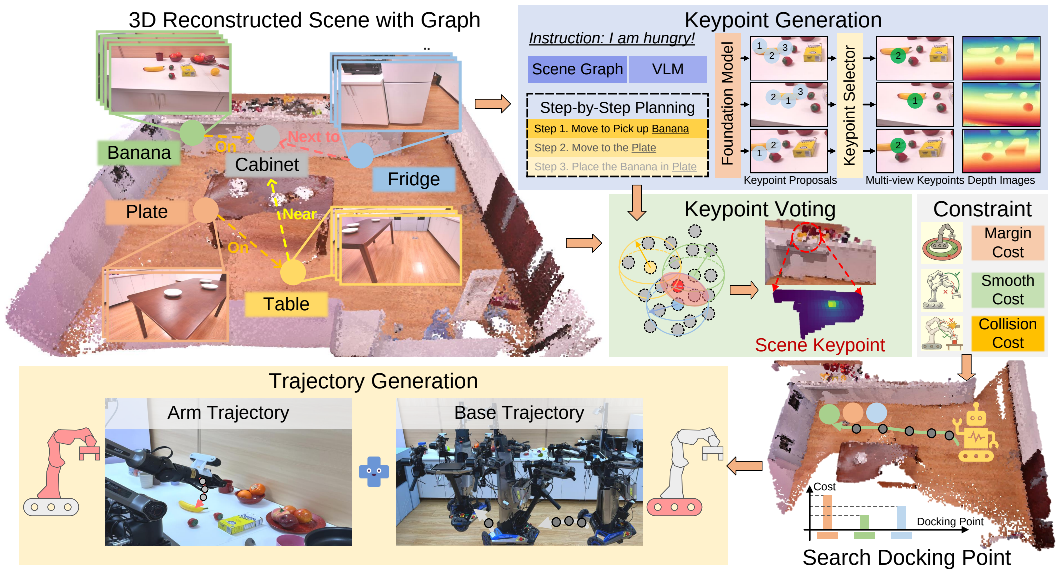

Figure 2: The pipeline of MoTo. Based on robot scanning RGB-D observation to get 3D scene point clouds and graphs, we utilize VLM and multi-view consistency voting to get interaction keypoints, and generate mobile manipulation trajectories via proposed cost constraint optimization.

4 Approach

4.1 Interaction-aware Navigation

Most mobile manipulation tasks can be decomposed into subtasks, each involving a single target object. For example, $\mathcal{T}$: "I want to drink milk." can be divided into: (1) pick up a **bottle**; (2) place the bottle on the **table**; (3) pick up the **milk**, etc. Each subtask has one target object (in **bold**). Two objects are involved in subtask (2): one in the robot's gripper and another to be interacted with, the latter being defined as the target object. We assume each subtask can be completed by first navigating and then manipulating with a fixed base[1]. Since fixed-base manipulation policies are available, the main challenge for our work is how and where to move, i.e., the last-mile challenge.

Determining how and where to move is a non-trivial problem that cannot be solved by a pure navigation policy. A robot must stop at a docking point for feasible manipulation, where the target object is at a suitable orientation and distance with no significant obstacles. Since high-quality mobile manipulation data is limited, we tackle this problem in a zero-shot manner by introducing interaction-aware navigation. We observe that a subtask becomes easy with a fixed base if (1) the robot can reach the interaction-relevant keypoints of the target object and (2) the arm retains sufficient motion margin. The keypoints must be part-level. Imagine the robot is going to open a fridge. The robot cannot open without moving the base if it reaches the back of the fridge. Similarly, the robot must choose context-specific interaction points. If the gripper is empty, the end-effector can be used; if holding a tool, such as a broom, the relevant part (e.g., broom tip) must engage with the environment (e.g., contacting trash on the floor). Based on this insight, interaction-aware navigation decomposes the movement problem into two subproblems: (1) identifying the target keypoint (TK) on the object and the arm keypoint (AK) on the manipulator, and (2) controlling the robot to move until TK and AK are aligned for interaction.

Approach Overview. To accurately ground instruction $\mathcal{T}$ to objects in the scene, we first let the robot scan the whole scene and reconstruct a 3D point cloud $P \in \mathbb{R}^{N\times 3}$ along with a 3D scene graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Each object node in $\mathcal{N}$ stores the 3D coordinate, 3D instance mask and several

[1]Navigation is optional; if the robot can operate the target object with its arm without moving the base, this step is unnecessary. We do not consider tasks requiring simultaneous base and arm movement, such as opening a door with a rotating handle.

object-centric images $\{I_1^k, ..., I_m^k\}$ of an object o_k. The edge in $\mathcal{E}$ indicates the spatial relationship between two objects, such as “inside” and “on top of”. With $\mathcal{G}$, we can prompt an LLM to perform task planning based on the existing objects and their relationship:

$$\{(\mathcal{T}_1, o_1), ..., (\mathcal{T}_n, o_n)\} = \mathrm{LLM}(\mathcal{T}, \mathcal{G}) \tag{2}$$

where $\mathcal{T}_k$ is the k-th subtask. $o_k \in \mathcal{N}$ is the corresponding target object, which should be explicitly referred to in the language description of $\mathcal{T}_k$. We can also query the object-centric image set $\{I_1^k, ..., I_m^k\}$ from $\mathcal{G}$ with o_k. Based on these, we leverage vision-language models (VLM) to obtain target keypoint $\mathcal{P}_k^T$ and arm keypoint $\mathcal{P}_k^A$ in the world coordinate system:

$$\begin{aligned} \mathcal{P}_k^T &= \mathcal{V}(\mathrm{VLM}(\mathcal{T}_k, \{I_1^k, ..., I_m^k\}), P) \\ \mathcal{P}_{k,t}^A &= \Gamma(\mathrm{VLM}(\mathcal{T}_k, \{s_t^w\}), E_t) \end{aligned} \tag{3}$$

where we consider the scene point clouds P in the world coordinate system. Γ represents the mapping function from the robot arm coordinate system to the camera coordinate system. $\mathrm{VLM}(\mathcal{T}_k, \{I_1^k, ..., I_m^k\})$ generates target keypoint proposals in different images, which are then aggregated with a voting module $\mathcal{V}$. $\mathrm{VLM}(\mathcal{T}_k, \{s_t^w\})$ first generates arm keypoint in the robot's coordinate system, which is then transformed to the world coordinate system via the mapping function Γ using the robot-to-world transformation E_t acquired with off-the-shelf SLAM method. Note that $\mathcal{P}_{k,t}^A$ is a function of the position and pose of the robot base and arm. Therefore, the robot's action $(a_t^{base}, \{a_t^{arm}\})$ can be solved as an optimization problem, which aims to minimize the distance between TK and AK as well as following several constraints:

$$\underset{a_{1:T}^{base}, \{a_{1:T}^{arm}\}}{\operatorname{argmin}} \sum_{t=1}^{T} ||\mathcal{P}_k^T - \mathcal{P}_{k,t}^A(\theta_t^{base}, \{\theta_t^{arm}\})|| + \mathcal{C}_t \tag{4}$$

where θ_t^{base} and $\{\theta_t^{arm}\}$ represent the position and pose of robot base and arm respectively. $\mathcal{C}_t$ is the additional constraints. We will detail Eq (3) and (4) in Section 4.2 and 4.3 respectively.

4.2 VLM-based Keypoint Generation

Since high-quality data on instruction-affordance pairs is limited, training a model to generate reasonable TK and AK according to text description $\mathcal{T}_k$ is very challenging. However, we notice that although current models are unable to directly generate task-related keypoints on an object, VLMs demonstrate great ability to select a proper keypoint among a set of candidates given the task description. Therefore, we propose a two-stage VLM-based method to generate keypoints for an image, which is divided into keypoint proposal stage and keypoint selection stage.

Keypoint Proposal. In this stage, we exploit visual cues in the given image to generate several keypoint proposals. Without any language information, we aim to find out all representative actionable points in an image as proposals. Note that this is more challenging than part segmentation, because actionable points may not have significant color or texture differences. For example, to close a laptop we should push the back of the screen, to lift a table we should hold both centers of its opposite edges. These points to be interacted with do not belong to a specific part. Current segmentation models can only segment a laptop into screen and keyboard, and a table into surface and legs, which cannot provide detailed, actionable locations. Inspired by ReKep [38], we adopt DINOv2 [39] to extract pixel-wise representation of the image, where semantically meaningful regions show larger activations. Then we apply SAM [40] to extract fine-grained part-level masks. Finally, in each segmented mask, we apply clustering on the DINOv2 features normalized with depth information, and select the center of each cluster as the keypoint proposal[2].

Keypoint Selection. Given the extracted keypoint proposals, we index and plot them on the image. We then prompt VLM with this modified image as well as the task description $\mathcal{T}_k$. VLM will select the most suitable keypoint from the proposals.

[2]If no keypoints are provided, the end-effector's proprioceptive position is used as the default.

We adopt a two-stage pipeline to obtain AK. Firstly, extracting the wrist keypoint from the RGB-D observation s_t^w, then projecting it to 3D space using E_t. If the gripper is empty, we skip this process and directly set the end-effector's 3D position as AK. For dual-arm robots, both arms may output keypoints, which confuses the following optimization process. To address this, we adopt VLM as a binary classifier to select a specific arm for the current subtask, ensuring only one AK contacts TK. The VLM is prompted with $\mathcal{T}_k$, the Euclidean distances from each end-effector to the target, and the gripper contents ("empty" or object category) to determine the appropriate arm.

Multi-view Voting for TK. Different from AK, TK should be generated on the target object, which may be far away from the robot and is harder to be backprojected accurately. So we further propose a voting module $\mathcal{V}$ to aggregate multi-view predictions into the final TK in 3D space.

To enable robust voting, we modify the keypoint selection prompt to let VLM select the top V most relevant keypoint proposals. In this way, we generate $V \times m$ keypoints for the image set $\{I_1^k, ..., I_m^k\}$ of target object. These keypoints are projected to 3D space with corresponding depth images and camera parameters, which are denoted as $P_K \in \mathbb{R}^{(V\times m)\times 3}$. We then utilize P_K to vote for the points P in the scene:

$$\mathcal{V}(P_K, P) = \operatorname*{argmax}_i \sum_{j=1}^{V\times m} \mathbb{I}(||P(i) - P_K(j)||_2 - \tau) \tag{5}$$

where $\mathbb{I}$ is an indicator function that converts negative values to 1 and others to 0, and τ is a Euclidean distance threshold. We use the voting result to index P and obtain the 3D point of TK.

4.3 Keypoint-guided Optimization

With the generated TK and AK, now we solve an optimization problem to make them touch with each other to control the robot moving to a right point. Since the robot-to-world transformation E_k depends on the position and pose of robot base and arm, the arm keypoint $\mathcal{P}_{k,t}^A$ can be written as a function of θ_t^{base} and $\{\theta_t^{arm}\}$. Note that we have:

$$a_t^{base} = \Delta\theta_t^{base}, \ \ \{a_t^{arm}\} = \Delta\{\theta_t^{arm}\} \tag{6}$$

Therefore by formulating $\mathcal{C}_t$ as the function of θ_t^{base} and $\{\theta_t^{arm}\}$, we can solve Eq (4) to obtain the optimal trajectories of robot base and arm, from which the actions can be easily derived with Pinocchio IK solver [41].

In addition to making TK and AK touch each other, we also need to design constraints $\mathcal{C}_t$ to ensure the smooth execution of the control, including collision avoidance, control smoothness, and the margin of the robotic arm. We formulate three cost functions to achieve this:

$$\mathcal{C}_t(\theta_t^{base}, \{\theta_t^{arm}\}) = \mathcal{F}_t^c + \mathcal{F}_t^s + \mathcal{F}_t^m \tag{7}$$

Collision Cost. The robot is required to avoid any collision with objects in the scene to ensure safety during the mobile manipulation process. We uniformly sample N_q query points on the robot surface Ω to evaluate the collision cost:

$$\mathcal{F}_t^c = \sum_{j=1}^{N_q} \max(0, \ \epsilon_0 - \mathcal{D}(q_t^j, P)), \quad q_t^j \in \Omega(\theta_t^{base}, \{\theta_t^{arm}\}) \tag{8}$$

where q_j^t means the j-th query point, ϵ_0 is a hyperparameter that controls the safety margin for collision avoidance. $\mathcal{D}(q_j^t, P)$ means the distance between q_j^t and the scene point clouds P. We expect the distance between the robot surface and the scene can be maximized if it is within the safety margin. Otherwise, the collision cost will not influence the control of the robot.

Smoothness Cost. This cost keeps continuous and smooth changes for joint angles of the robot arm and translation and rotation of the base, where sudden changes are avoided to maintain the safety of the motor and the embodiment. We define trajectory smoothness as the difference of joint angles $\boldsymbol{\theta}^t$ of the arm and proprioception of the base between consecutive poses in the candidate trajectory:

$$\mathcal{F}_t^s = ||\theta_{t+1}^{base} - \theta_t^{base}||_2 + \sum ||\theta_{t+1}^{arm} - \theta_t^{arm}||_2 \tag{9}$$

Table 1: Comparison results on the OVMM benchmark. Partial success rates indicate the execution of each stage, conditioned on the success of the preceding one. "RL" and "Heuristic" refer to object placement methods based on RL and heuristics, respectively. The RL method is used by default.

Method	**Partial Success Rates**			**Overall SR**	**Average SR**	**Step**
	FindObj (↑)	Pick (↑)	FindRec (↑)			
Home-Robot (RL)	66.60%	61.10%	50.90%	14.80%	48.30%	1132.5
Home-Robot (Heuristic)	65.40%	54.80%	43.70%	7.30%	42.80%	1009.8
UniTeam [43]	66.13%	62.65%	54.69%	17.96%	50.36%	1027.7
Home-Robot w/ L3MVN	68.10%	60.14%	57.46%	15.28%	50.25%	1084.3
OpenVLA w/ L3MVN	68.28%	60.68%	58.00%	16.09%	50.76%	1191.5
Home-Robot w/ MoTo (Ours)	65.24%	60.23%	50.40%	18.32%	48.55%	1152.2
OpenVLA w/ MoTo (Ours)	66.67%	60.95%	49.87%	20.64%	49.53%	1195.1

Table 2: Ablation experiments for optimization cost terms and keypoint generation variants.

Method		**Partial Success Rates**			**Overall SR**	**Average SR**
		FindObj (↑)	Pick (↑)	FindRec (↑)		
Cost	w/o Collision	66.93%	60.95%	49.24%	18.50%	48.91%
	w/o Smoothness	66.76%	61.48%	49.60%	19.75%	49.40%
	w/o Margin	65.95%	60.77%	50.04%	17.87%	48.66%
Keypoint	SAM+CLIP	66.31%	58.36%	50.58%	20.29%	48.88%
	SAM+ViT	65.95%	57.19%	49.87%	20.46%	48.37%
	w/o Fusion	66.13%	56.21%	49.96%	15.19%	46.87%
	Single View	66.49%	58.27%	50.04%	17.61%	48.19%

We leverage the joint angle solved via IK for smoothness constraint instead of the proprioception of the arm, because small changes in the arm pose cannot guarantee slight difference in joint angles.

Margin Cost. When AK and TK make contact, a large margin is required for the robotic arm to enable subsequent fixed-base manipulation. We define the horizontal distance between the end-effector and the robot base center as the arm radius r, bounded by constants r_{min} and r_{max}. Given the robot's design, r is a function of the arm pose $r = g(\theta_t^{arm})$. The margin cost is defined as:

$$\mathcal{F}_t^m = ||\frac{r_{min} + r_{max}}{2} - g(\theta_t^{arm})||_2 \tag{10}$$

In this way, the robotic arm will not be too curved or too extended, ensuring that it has a large margin to operate.

In the process of optimization solving, the decision variables include the robotic arm joint angle $\{\theta_t^{arm}\}$ and base pose θ_t^{base}. Specifically, we generate initial solution proposals for the next time step from the search space and calculate the cost relative to the optimization objective for each proposal. Using the Dual Annealing [42] algorithm, we iteratively update the initial solutions based on the change in cost, continuing this process until the cost is reduced below a predefined threshold.

5 Experiment

5.1 Comparison with State-of-the-art Methods

Table 1 demonstrates the performance of MoTo on the OVMM [18] validation set compared to the baseline, decomposing it into four sequential stages: finding the target (FindObj), grasping (Pick), finding the container (FindRec), and placing (Overall SR). Our training and testing methods fully comply with the OVMM baseline settings (Home-Robot). The implementation details are in Appendix A.1. The similar success rates in stages FindObj and Pick are due to MoTo's focus on interaction-aware navigation, which is invoked only after finding a container without exploring it. As shown in Table 1, Home-Robot w/ MoTo outperforms Home-Robot (RL), achieving a 3.52% higher overall success rate. With a stronger manipulation foundation (e.g., OpenVLA [11]), MoTo's

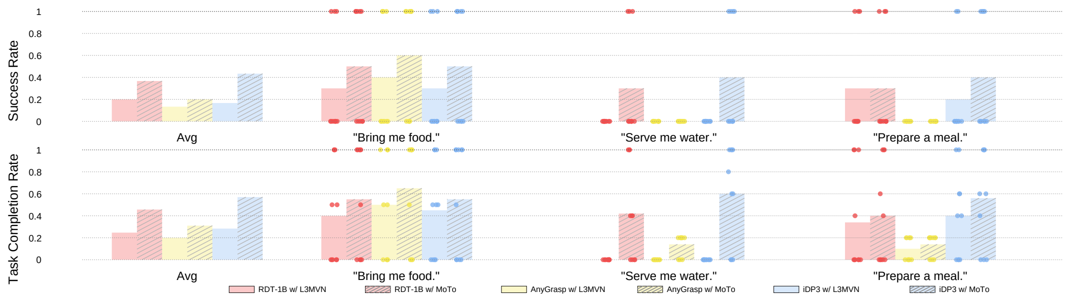

Figure 3: Real-world experimental results. All methods are run 10 times on the three types of mobile manipulation tasks, where the dots represent the performance of each test (Best view in color).

success rate further improves by 2.32%. We also compare MoTo with the conventional navigation framework L3MVN [44]. Since L3MVN only ensures proximity to the target while ignoring the feasibility of subsequent manipulations, it yields only a marginal overall success rate gain of 0.48%, despite a 6.65% improvement in the FindRec stage.

5.2 Ablation Study

Table 2 reports a systematic ablation of our full MoTo pipeline in OVMM, isolating the contribution of each optimization cost term and keypoint design choice, where w/o Fusion denotes that no voting is applied on the generated multi-view keypoints, and Single View means only the view with the most target pixels is utilized. Experimental results demonstrate that each constraint cost in MoTo can improve performance. Since the OVMM involves only the pick and place task, the margin cost is more important than the others and determines whether the arm can reach the target position or not. We further investigated variants of the keypoint extraction and fusion pipeline. The inconsistency of multi-view keypoints in the "w/o Fusion" setting results in a serious performance drop (2.42% lower success rate compared to Single View), because the cluttered keypoints fail to provide effective guidance. Wrong interaction keypoints will result in the robot moving elsewhere due to physical constraints that prevent it from performing fixed-base manipulation actions. The choice of visual foundation model for keypoint extraction has little impact on MoTo; replacing DINOv2 with CLIP or ViT still yields high performance.

5.3 Real World Experiments

The OVMM baseline cannot be directly deployed in the real world due to the sim-to-real gap. To validate the plug-in scalability of MoTo, we integrated fixed-base manipulation models (AnyGrasp, iDP3, RDT-1B) covering heuristics, diffusion policy, and foundation models (Fig. 3). Compared to conventional navigation with fixed-base manipulation, MoTo consistently improves performance across all tasks, demonstrating both efficiency and scalability. AnyGrasp, limited to grasping, performs poorly on complex interaction tasks. Interestingly, iDP3 and AnyGrasp outperform RDT-1B in mobile manipulation, owing to stronger viewpoint generalization from 3D point cloud observations. These results highlight 3D egocentric manipulation policies as a promising direction for future mobile manipulation.

6 Conclusion

In this paper, we introduce MoTo, a plug-in interaction-aware navigation module that enables zero-shot transfer of fixed-base manipulation models to mobile manipulation tasks. MoTo employs vision foundation models to generate interaction keypoints and leverages multi-view consistency to fuse them, providing accurate guidance for base and arm motion planning. We further design motion planning objectives based on arm and goal keypoints with cost constraints to enhance trajectory feasibility. Extensive experiments in both simulation and the real world demonstrate the effectiveness of MoTo and its scalability across diverse manipulation models.

7 Limitation

Although the proposed MoTo can be plug-and-play with any off-the-shelf fixed-base manipulation model for efficient mobile manipulation, real-world deployments still have some limitations. First, MoTo lacks the ability to reconstruct the scene online, which prevents deployment in dynamic environments. Second, MoTo relies on interaction-aware navigation to generate docking points for fixed-base manipulation models but does not incorporate sufficient whole-body control policies, making it struggle with complex tasks such as door opening. Third, MoTo makes a strong assumption that all relevant objects are fully observable within the agent's current view and at sufficient resolution to allow reliable keypoint extraction (e.g., 'we first let the agent scan the whole scene'). This assumption limits robustness in partially observable or cluttered settings.

In future work, we plan to equip MoTo with incremental online scene reconstruction and perception algorithms to handle dynamic and partially observable environments. Additionally, we aim to integrate cost constraints directly into policy generation (e.g., during the denoising step of Diffusion Policy) to produce fine-grained whole-body control trajectories, thereby enabling more challenging mobile manipulation tasks.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62376032 and 624B2076, in part by the Beijing Natural Science Foundation under Grant No. L257008, and in part by the MoE AcRF Tier 1 Seed Grant RS17/24 and NTU EEE Ignition Research Grant 024920-00001.

References

[1] H. Xiong, R. Mendonca, K. Shaw, and D. Pathak. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024.

[2] R. Yang, Y. Kim, R. Hendrix, A. Kembhavi, X. Wang, and K. Ehsani. Harmonic mobile manipulation. *arXiv preprint arXiv:2312.06639*, 2023.

[3] Z. Wu, Y. Zhou, X. Xu, Z. Wang, and H. Yan. Momanipvla: Transferring vision-language-action models for general mobile manipulation. *arXiv preprint arXiv:2503.13446*, 2025.

[4] A. Xiao, N. Janaka, T. Hu, A. Gupta, K. Li, C. Yu, and D. Hsu. Robi butler: Remote multimodal interactions with household robot assistant. *arXiv preprint arXiv:2409.20548*, 2024.

[5] B. Abbatematteo, E. Rosen, S. Thompson, T. Akbulut, S. Rammohan, and G. Konidaris. Composable interaction primitives: A structured policy class for efficiently learning sustained-contact manipulation skills. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7522–7529. IEEE, 2024.

[6] Y. Peng, Z. Wang, Y. Zhang, S. Zhang, N. Cai, F. Wu, and M. Chen. Revolutionizing battery disassembly: The design and implementation of a battery disassembly autonomous mobile manipulator robot (beam-1). *arXiv preprint arXiv:2407.06590*, 2024.

[7] P. Štibinger, G. Broughton, F. Majer, Z. Rozsypálek, A. Wang, K. Jindal, A. Zhou, D. Thakur, G. Loianno, T. Krajník, et al. Mobile manipulator for autonomous localization, grasping and precise placement of construction material in a semi-structured environment. *RA-L*, 6(2):2595–2602, 2021.

[8] X. Huang, D. Batra, A. Rai, and A. Szot. Skill transformer: A monolithic policy for mobile manipulation. In *ICCV*, pages 10852–10862, 2023.

[9] N. Yokoyama, A. Clegg, J. Truong, E. Undersander, T.-Y. Yang, S. Arnaud, S. Ha, D. Batra, and A. Rai. Asc: Adaptive skill coordination for robotic mobile manipulation. *RA-L*, 9(1): 779–786, 2023.

[10] X. Li, M. Zhang, Y. Geng, H. Geng, Y. Long, Y. Shen, R. Zhang, J. Liu, and H. Dong. Manipllm: Embodied multimodal large language model for object-centric robotic manipulation. In *CVPR*, pages 18061–18070, 2024.

[11] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[12] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[13] H. Yin, X. Xu, Z. Wu, J. Zhou, and J. Lu. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. *arXiv preprint arXiv:2410.08189*, 2024.

[14] D. Choi, A. Fung, H. Wang, and A. H. Tan. Find everything: A general vision language model approach to multi-object search. *arXiv preprint arXiv:2410.00388*, 2024.

[15] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *T-RO*, 2023.

[16] Y. Ze, G. Yan, Y.-H. Wu, A. Macaluso, Y. Ge, J. Ye, N. Hansen, L. E. Li, and X. Wang. Gnfactor: Multi-task real robot learning with generalizable neural feature fields. In *CoRL*, pages 284–301, 2023.

[17] J. Gu, D. S. Chaplot, H. Su, and J. Malik. Multi-skill mobile manipulation for object rearrangement. *arXiv preprint arXiv:2209.02778*, 2022.

[18] S. Yenamandra, A. Ramachandran, K. Yadav, A. Wang, M. Khanna, T. Gervet, T.-Y. Yang, V. Jain, A. W. Clegg, J. Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023.

[19] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

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

[21] R.-Z. Qiu, Y. Hu, G. Yang, Y. Song, Y. Fu, J. Ye, J. Mu, R. Yang, N. Atanasov, S. Scherer, et al. Learning generalizable feature fields for mobile manipulation. *arXiv preprint arXiv:2403.07563*, 2024.

[22] P. Zhang, X. Gao, Y. Wu, K. Liu, D. Wang, Z. Wang, B. Zhao, Y. Ding, and X. Li. Momakitchen: A 100k+ benchmark for affordance-grounded last-mile navigation in mobile manipulation. *arXiv preprint arXiv:2503.11081*, 2025.

[23] S. Yan, Z. Zhang, M. Han, Z. Wang, Q. Xie, Z. Li, Z. Li, H. Liu, X. Wang, and S.-C. Zhu. M2diffuser: Diffusion-based trajectory optimization for mobile manipulation in 3d scenes. *arXiv preprint arXiv:2410.11402*, 2024.

[24] R. Mendonca, E. Panov, B. Bucher, J. Wang, and D. Pathak. Continuously improving mobile manipulation with autonomous real-world rl. *arXiv preprint arXiv:2409.20568*, 2024.

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

[26] E. Rosen, S. James, S. Orozco, V. Gupta, M. Merlin, S. Tellex, and G. Konidaris. Synthesizing navigation abstractions for planning with portable manipulation skills. In *Conference on Robot Learning*, pages 2278–2287. PMLR, 2023.

[27] F. Wang, S. Lyu, P. Zhou, A. Duan, G. Guo, and D. Navarro-Alarcon. Instruction-augmented long-horizon planning: Embedding grounding mechanisms in embodied mobile manipulation. 2025.

[28] X. Meng, X. Yang, S. Jung, F. Ramos, S. S. Jujjavarapu, S. Paul, and D. Fox. Aim my robot: Precision local navigation to any object. *arXiv preprint arXiv:2411.14770*, 2024.

[29] B. Quartey, E. Rosen, S. Tellex, and G. Konidaris. Verifiably following complex robot instructions with foundation models. *arXiv preprint arXiv:2402.11498*, 2024.

[30] M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[31] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *CRAS*, 3(1):269–296, 2020.

[32] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[33] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *ICRA*, pages 13438–13444, 2021.

[34] K. Mo, Y. Qin, F. Xiang, H. Su, and L. Guibas. O2o-afford: Annotation-free large-scale object-object affordance learning. In *CoRL*, pages 1666–1677, 2022.

[35] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *IJRR*, 2023.

[36] S. Huang, Z. Wang, P. Li, B. Jia, T. Liu, Y. Zhu, W. Liang, and S.-C. Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *CVPR*, pages 16750–16761, 2023.

[37] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[38] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[39] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[40] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *ICCV*, pages 4015–4026, 2023.

[41] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard. The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 614–619. IEEE, 2019.

[42] Y. Xiang, D. Sun, W. Fan, and X. Gong. Generalized simulated annealing algorithm and its application to the thomson model. *Physics Letters A*, 233(3):216–220, 1997.

[43] A. Melnik, M. Büttner, L. Harz, L. Brown, G. C. Nandi, A. PS, G. K. Yadav, R. Kala, and R. Haschke. Uniteam: Open vocabulary mobile manipulation challenge. *arXiv preprint arXiv:2312.08611*, 2023.

[44] B. Yu, H. Kasaei, and M. Cao. L3mvn: Leveraging large language models for visual target navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3554–3560. IEEE, 2023.

[45] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5021–5028. IEEE, 2024.

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

[47] Y. Ze, Z. Chen, W. Wang, T. Chen, X. He, Y. Yuan, X. B. Peng, and J. Wu. Generalizable humanoid manipulation with improved 3d diffusion policies. *arXiv preprint arXiv:2410.10803*, 2024.

[48] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

A Implementation Details

A.1 Simulator Experiment

The OVMM benchmark consists of 60 extensive indoor scenes and contains more than 18k 3D models of everyday objects.OVMM utilizes Hello Robot as an agent to perform the “Move a target object from container A to container B” mobile manipulation task. We utilize an OVMM-heuristic baseline to collect manipulation expert trajectories that include robot proprioception, action, and visual observations to fine-tune off-the-shelf manipulation foundation models. The simulator experiments are training and testing on 8 RTX 3090 GPUs. MoTo leverages ConceptGraph [45] to build 3D scene point clouds and scene graphs from RGB-D inputs. The object nodes and edges in the scene graph are converted into a structured description to query the VLM for target identification.

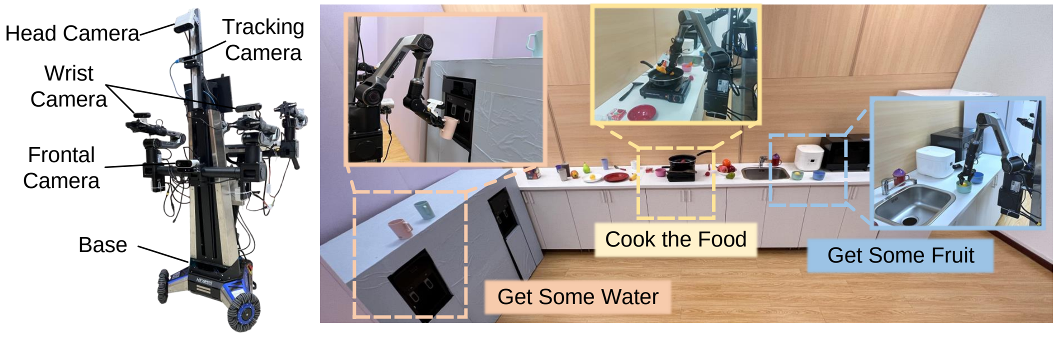

Figure 4: Real-world experimental platforms and deployment environments.

A.2 Real World Experiment

For the real-world experiments, we use HEXMOVE as the base and two PiPER robotic arms to build a dual-arm mobile manipulation robot, which is equipped with a Femto Bolt RGB-D sensor as the head camera to acquire high-quality scene point clouds, and two Gemini 336L RGB-D sensors as the wrist cameras to assist in the manipulation task execution. Meanwhile, an Intel tracking camera T265 is employed to acquire the robotic camera poses to reconstruct the deployment scene. We recognize scene target objects and task planning using GroundingSAM [46] and GPT-4o, respectively, which ensure the zero-shot generalization ability of the overall framework. MoTo also leverages ConceptGraph [45] to build 3D scene point clouds and scene graphs from RGB-D inputs. Figure 4 further demonstrates the real-world experiment platform and environment. The real-world experiments are all performed on a single RTX 4060 GPU.

A.3 Training Details

In the OVMM simulator, we collect expert demonstration data for pick-and-place during mobile manipulation with heuristic baselines. To better fine-tune OpenVLA [11] to mitigate cross-robot ontology differences, we collected a total of 20k data and fine-tuned 10k epoch on 8 RTX 3090 GPUs using the LoRA strategy. We ensured the diversity of viewpoints in the expert trajectories during the collection process in order to achieve a higher viewpoint generalization.

For both RDT-1B [12] and iDP3 [47] manipulation policies, we collected fine-tuning data in a realistic kitchen environment, spanning three long-horizon tasks—*Arrange various food items*, *Dispense hot water from a water dispenser and mix with cold water*, and *Cook food*—each decomposed into 2-5 subtasks (e.g., “pick up ingredient,” “align container,” “adjust water temperature”) with 50 expert demonstrations per task.

RDT-1B: The RDT-1B policy uses dual-arm 6-DOF joint positions, gripper open/close angles, and synchronized RGB streams (640×480 @ 30 Hz) from three cameras—one frontal and two mounted on the left and right grippers—as inputs. We fine-tuned the model for 150,000 gradient steps on 8 NVIDIA RTX 4090 GPUs (total batch size 128) using the AdamW optimizer (learning rate 1×10^{-4}, weight decay 1×10^{-2}), following the default RDT-1B configuration.

Table 3: Real-world task instructions. The target is the object to be interacted with, and the subgoal is the condition that needs to be completed.

Type	Instruction	Target	Subgoal
Bring me food	Give me a banana	Banana	1. Grasp the target 2. Place the target in the plate
	Get some food for me I want to eat some food I am hungry	Strawberry, Banana or Lemon	
	I want to eat a strawberry	Strawberry	
Serve me water	I want a cup of water	Cup	1. Grasp the target 2. Fetch hot water 3. Grasp target filled with cold water 4. Mix hot and cold water 5. Put down the targets
	I want to drink water I need drink Give me some water, please!	Cup, Mug or Bowl	
Prepare a meal	Prepare a launch Make some salad Warm up the food Cook for a meal	Banana, Strawberry or Corn	1. Grasp the target 2. Put the target in the Pan 3. Grasp the bowl 4. Grasp the cooked target 5. Put the target in the Bowl

iDP3: The iDP3 policy takes dual-arm 6-DOF joint positions, gripper angles, and point-cloud frames (640×480 depth $\rightarrow$ 3D XYZRGB) from a frontal Orbbec Femto Bolt (20 Hz) as inputs. We reduced both the agent-state vector and action vector to 14 dimensions (7 per arm) to match our dual-arm kinematics. Training was performed for 3,000 epochs on a single RTX 4090 GPU (batch size 64) using Adam (learning rate 1×10^{-4}, weight decay 1×10^{-6}). Validation performance plateaued after approximately 2,500 epochs, and we selected the checkpoint with the lowest validation loss.

B Real World Tasks and Metrics

B.1 Task Description

In real-world experiments, we have tested three types of task instructions to verify that MoTo can fully utilize the off-the-shelf fixed manipulation model to accomplish the task requirements. Specifically, each task corresponds to multiple instructions, and the robot needs to reason about the best interaction goal as well as action planning based on the scene information. Details of the test tasks are demonstrated in Table 3.

B.2 Metrics

We follow the metrics of [48] and use the success rate and task completion rate to measure real-world mobile manipulation performance. For each instruction, the robot must complete the corresponding subtasks in Table 3 for the manipulation to succeed. Task completion rate is the ratio of completed subgoals to those necessary to complete a task, which reflects the progress of the robot in completing the task. For example, in the "Serve me water" task, if the robot only successfully grasps and places the target (e.g., a cup or bowl), but there is no water in the target, the task completion rate is 0.4.

C Optimization Algorithm

Navigation policy is one of the fundamental challenges of mobile manipulation, and it must ensure that the generated base docking points are feasible for subsequent actions. Conventional navigation methods only provide coarse docking point proposals that cannot be effectively applied to mobile manipulation. For this reason, we propose the optimization objective of Eq. 1 with the cost constraint of Eq. 3 to ensure the feasibility of the docking point selection utilizing an optimization search approach. Specifically, it is very difficult to search for mobile manipulation trajectories directly in large-scale scenes, and we use an existing navigation algorithm to move closer to the target

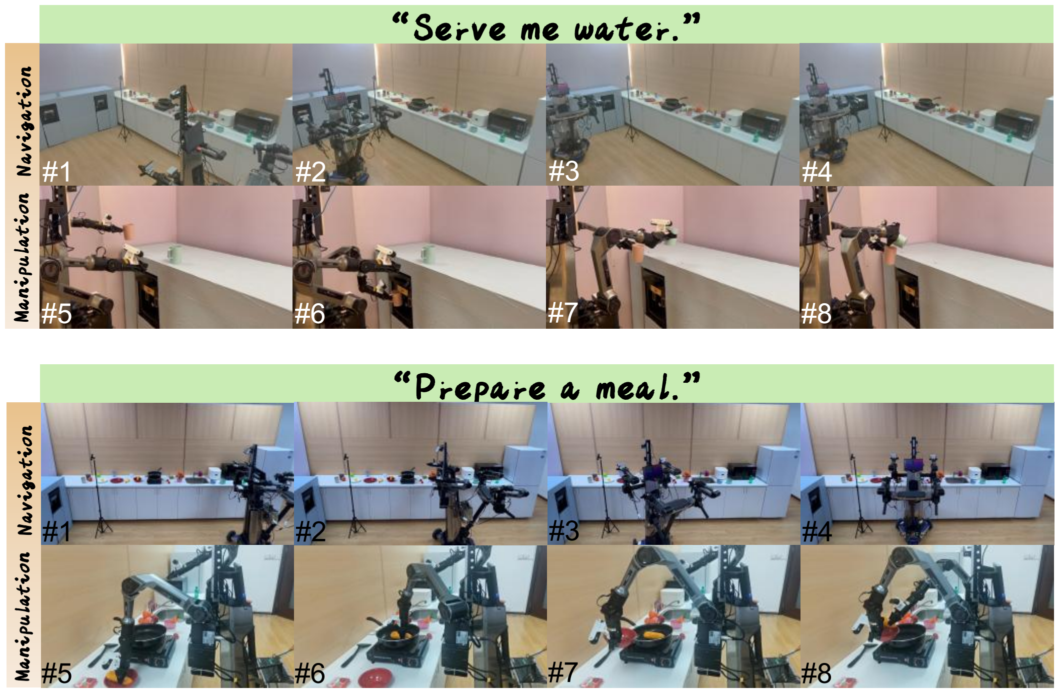

Figure 5: Visualization of mobile manipulation trajectories for real-world experiments.

object to reduce the optimized search space, searching for the optimal arm joint angle $\{\theta_t^{arm}\}$ and base pose θ_t^{base} iteratively with the Double Annealing Algorithm, as illustrated in Algorithm 1.

Algorithm 1: Dual Annealing-Based Trajectory Optimization for Mobile Manipulation

```
Input: Initial base pose $\theta_0^{base}$, arm joint configuration $\{\theta_0^{arm}\}$, time horizon $T$, cost
       function $\mathcal{O}$, temperature schedule $T_{anneal}$
Output: Optimized trajectory $\{\theta_t^{base}, \{\theta_t^{arm}\}\}_{t=0}^{T}$
for $t = 0$ to $T$ do
    Generate candidate proposals $\{(\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})^k\}_{k=1}^{K}$;
    Evaluate cost for each proposal: $C_k = \mathcal{O}(\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})$ ;        // Use Eq. (4)
    Select best candidate $(\theta_t^{base}, \{\theta_t^{arm}\}) = \arg\min_k C_k$;
    repeat
        Propose new candidate $(\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})$ using Dual Annealing schedule $T_{anneal}$;
        Compute cost $C_{new} = \mathcal{O}(\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})$;
        if acceptance criterion satisfied then
            Update current solution: $(\theta_t^{base}, \{\theta_t^{arm}\}) \leftarrow (\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})$;
        Update annealing temperature $T_{anneal} \leftarrow \text{cool}(T_{anneal})$;
    until cost improvement $< \varepsilon$ or max iterations reached;
return $\{\theta_t^{base}, \{\theta_t^{arm}\}\}_{t=0}^{T}$
```

D More Results

In this section, we present additional experimental results. We visualize mobile manipulation trajectories for tasks "Serve me water" and "Prepare a meal" in Figure 5 to demonstrate the performance of MoTo on mobile manipulation. We further show failure results to identify the limitations of the existing work and better help relevant researchers to follow this work.

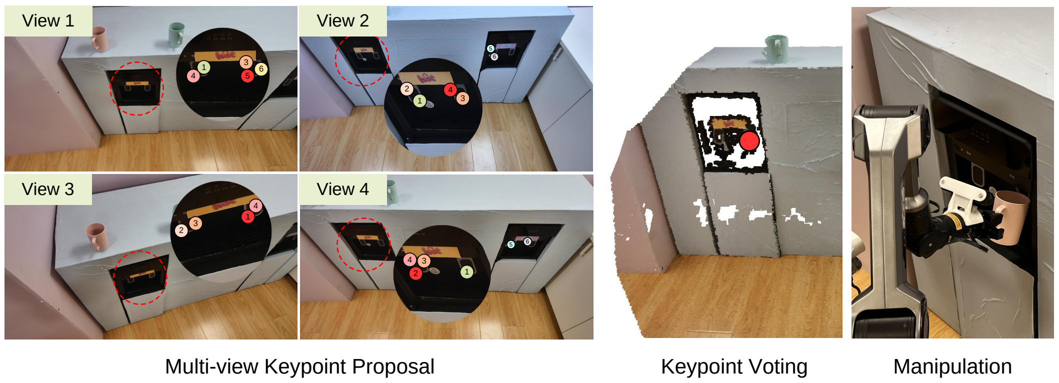

Figure 6: Visualization results for keypoint generation. MoTo selects keypoint proposals (red points) from multi-views, projects them into 3D space and votes to generate keypoints for manipulation.

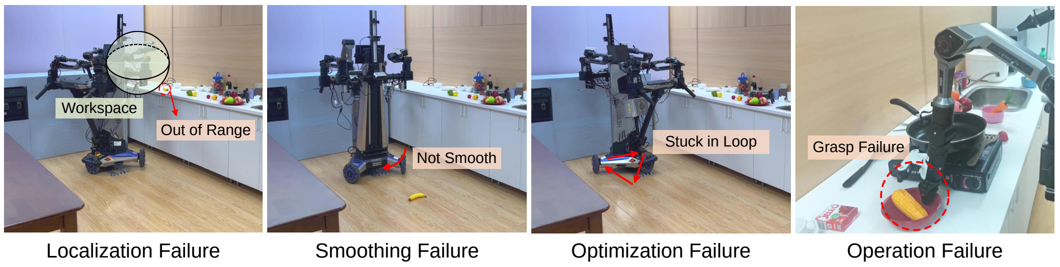

Figure 7: Failure Cases in real-world experiments.

D.1 Manipulation Visualization

Figure 6 demonstrates the scene keypoint generation and mobile trajectory in task "Serve me water". MoTo first extracts keypoint candidates from multi-view RGB images with VLM, and then projects them into the 3D space to vote for specific regions to be interacted with utilizing the consistency of the viewpoints. Based on the key point guidelines, MoTo searches for mobile trajectories using a zero-shot optimization algorithm, and generates feasible docking points for subsequent action, to execute the subsequent fixed-base manipulation.

D.2 Failure Cases

Figure 7 illustrates 4 common failures in the experiment, which are analyzed as follows:

Smoothing Failure: In real-world experiments, the unsmoothed motion trajectory of the base results in high acceleration, triggering the protection mechanism, which leads to the failure of mobile manipulation. The smoothing failure in Figure 7 illustrates that the neighboring waypoints on the mobile manipulation trajectory are distant from each other, and the excessive acceleration of the mobile base activates the protection mechanism, causing the banana to fall.

Localization Failure: SLAM is the key for the robot to reconstruct the scene, and also provides real-time position information to the robot. Errors due to SLAM can cause the robot to stay in a position far away from the specific target object.

Optimization Failure: Using the Dual Annealing algorithm sometimes gets stuck in a loop, causing the robot to keep adjusting the base position repeatedly.

Operation Failure: Due to a series of errors such as SLAM, the robot stops at a different position each time, which presents a challenge to the viewpoint generalization of the fixed-base manipulation model. How to improve the viewpoint generalization of the manipulation policy is one of the key bottlenecks in mobile manipulation.