

# MoTo Supplementary

## 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.

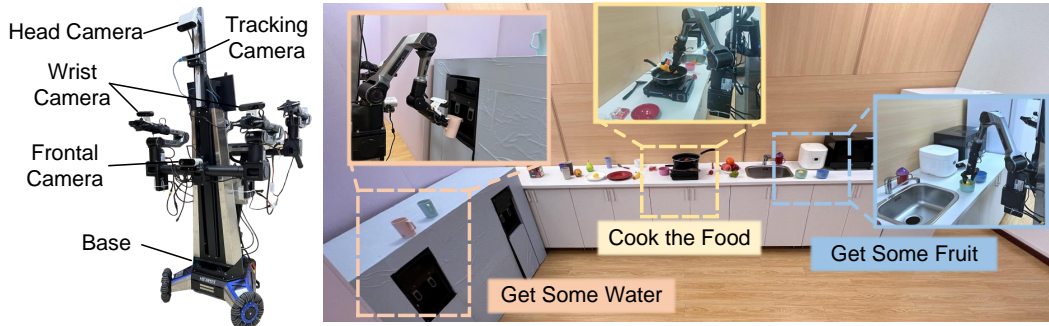


Figure 1: Real-world experimental platforms and deployment environments.

### 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 [1] and GPT-4o, respectively, which ensure the zero-shot generalization ability of the overall framework. Figure 1 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 [2] 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 [3] and iDP3 [4] 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.

Table 1: 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	Strawberry, Banana or Lemon	
	I am hungry I want to eat a strawberry	Strawberry	
Serve me water	I want a cup of water	Cup	1. Grasp the target
	I want to drink water I need drink Give me some water, please!	Cup, Mug or Bowl	2. Fetch hot water 3. Grasp target filled with cold water 4. Mix hot and cold water 5. Put down the targets
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

**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  $1 \times 10^{-4}$ , weight decay  $1 \times 10^{-2}$ ), following the default RDT-1B configuration.

**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  $1 \times 10^{-4}$ , weight decay  $1 \times 10^{-6}$ ). 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 1.

### B.2 Metrics

We follow the metrics of [5] 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 1 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

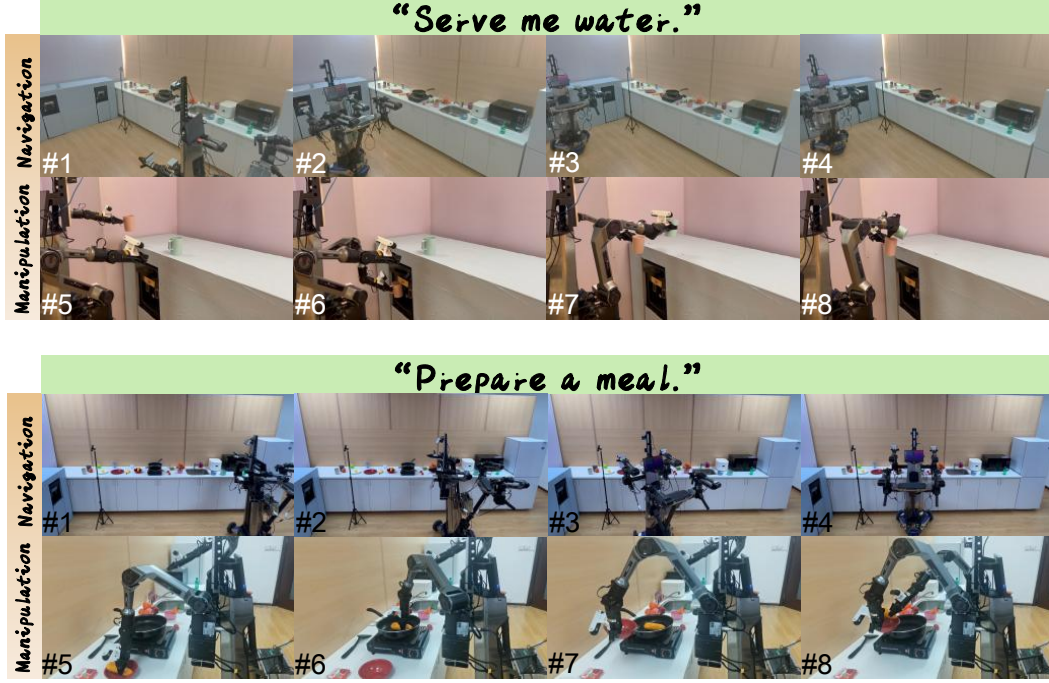


Figure 2: Visualization of mobile manipulation trajectories for real-world experiments.

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 object to reduce the optimized search space, searching for the optimal arm joint angle  $\{\theta_t^{arm}\}$  and base pose  $\theta_t^{base}$  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  $\{\theta_t^{base}, \{\theta_t^{arm}\}\}_{t=0}^T$

```

1 for  $t = 0$  to  $T$  do
2   Generate candidate proposals  $\{(\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})_{k=1}^K$ ;
3   Evaluate cost for each proposal:  $C_k = \mathcal{O}(\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})$ ;           // Use Eq. (4)
4   Select best candidate  $(\theta_t^{base}, \{\theta_t^{arm}\}) = \arg \min_k C_k$ ;
5   repeat
6     Propose new candidate  $(\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})$  using Dual Annealing schedule  $T_{anneal}$ ;
7     Compute cost  $C_{new} = \mathcal{O}(\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})$ ;
8     if acceptance criterion satisfied then
9       Update current solution:  $(\theta_t^{base}, \{\theta_t^{arm}\}) \leftarrow (\hat{\theta}_t^{base}, \{\hat{\theta}_t^{arm}\})$ ;
10    Update annealing temperature  $T_{anneal} \leftarrow \text{cool}(T_{anneal})$ ;
11  until cost improvement  $< \varepsilon$  or max iterations reached;
12 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 2 to demonstrate the performance

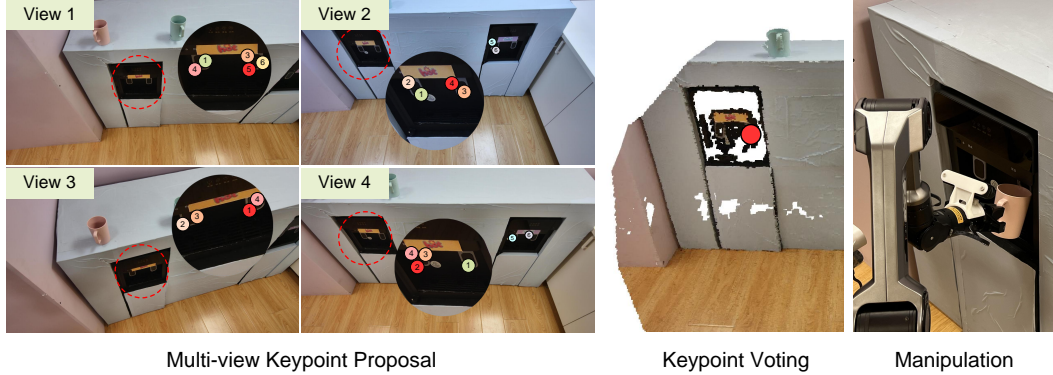


Figure 3: Visualization results for keypoint generation. MoTo selects keypoint proposals (red points) from multi-views, projects them into 3D space and votes to generate keypoints for manipulation.

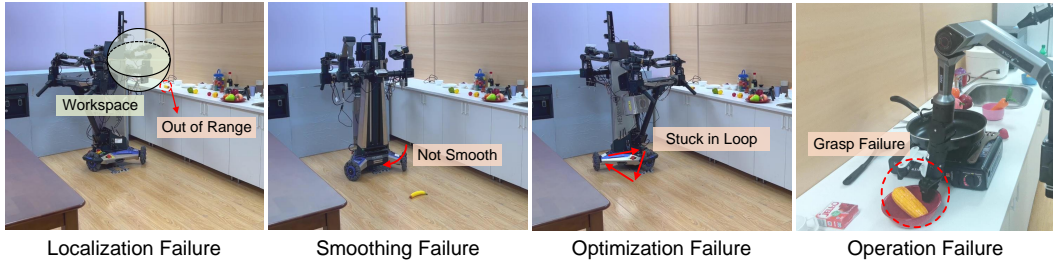


Figure 4: Failure Cases in real-world experiments.

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.

## D.1 Manipulation Visualization

Figure 3 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 4 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 4 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.

## References

- [1] 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.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] 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.