ReMoBot: Retrieval-Based Few-Shot Imitation Learning for Mobile Manipulation with Vision Foundation Models

Yuying Zhang[*1] **Wenyan Yang**[*1], **Francesco Verdoja**[1], **Ville Kyrki**[1], **Joni Pajarinen**[1]
[1]Department of Electrical Engineering and Automation
[1]Aalto University, Finland
{yuying.zhang,wenyan.yang,francesco.verdoja,ville.kyrki,joni.pajarinen}@aalto.fi

Abstract: Imitation learning (IL) algorithms typically distill experience into parametric behavior policies to mimic expert demonstrations. However, with limited demonstrations, existing methods often struggle to generate accurate actions, particularly under partial observability. To address this problem, we introduce a few-shot IL approach, *ReMoBot*, which directly *Re*trieves information from demonstrations to solve *Mo*bile manipulation tasks with ego-centric visual observations. Given the current observation, ReMoBot utilizes vision foundation models to identify relevant demonstrations, considering visual similarity w.r.t. both individual observations and history trajectories. A motion selection policy then selects the proper command for the robot until the task is successfully completed. The performance of ReMoBot is evaluated on three mobile manipulation tasks with a Boston Dynamics Spot robot in both simulation and the real world. With only 20 demonstrations, ReMoBot outperforms the baselines, achieving high success rates in Table Uncover (70%) and Gap Cover (80%), while also showing promising performance on the more challenging Curtain Open task in the real-world setting. Furthermore, ReMoBot demonstrates generalization across varying robot positions, object sizes, and material types.

Keywords: Few-shot Imitation Learning, Mobile Manipulation, Partial Observability

1 Introduction

Learning mobile manipulation purely from egocentric visual inputs is challenging due to partial observability arising from a limited camera field of view and the complexity of the environment [1, 2]. While reinforcement learning (RL) has shown promise in certain complex scenarios, it typically requires extensive exploration [3, 4], making it impractical for real-world applications without additional guidance. In contrast, imitation learning (IL) has enabled robots to efficiently acquire skills from expert demonstrations in various complex tasks [5]. However, the performance of IL methods depends greatly on the quantity and diversity of demonstrations, and approaches such as behavior cloning are prone to compounding errors over long task horizons [6]. To mitigate these issues, retrieval-based imitation learning [7, 8] has been proposed, which leverages expert demonstrations directly rather than relying solely on parametric policies. However, most existing approaches focus on static manipulation [7] and retrieve actions based only on individual state information, which is insufficient for egocentric mobile manipulation tasks where partial observability poses additional challenges.

To address these limitations, we introduce ReMoBot, a retrieval-based few-shot imitation learning framework to solve mobile manipulation tasks using only visual input. Unlike traditional parametric skill learning approaches, ReMoBot imitates demonstrated behaviors by retrieving visually simi-

9th Conference on Robot Learning (CoRL 2025) Workshop RemberRL, Seoul, Korea.

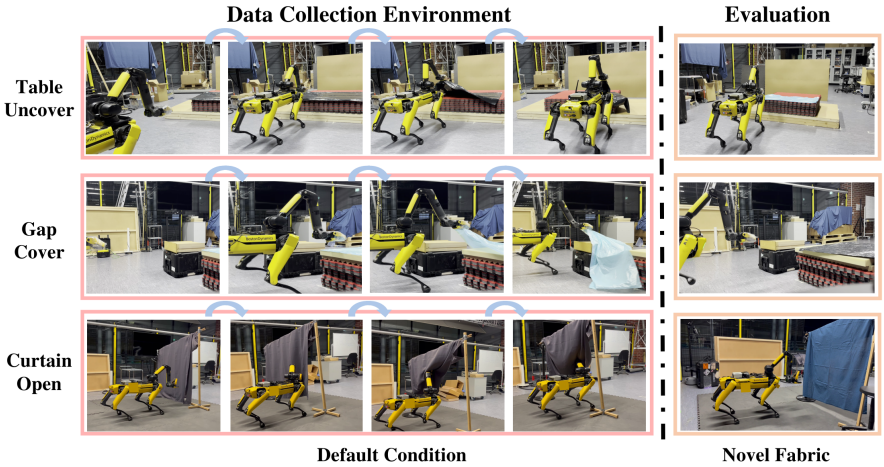

Figure 1: **Mobile Manipulation tasks.** Table Uncover (top), Gap Cover (middle), and Curtain Open (bottom) are shown in both the data collection (left) and novel fabric evaluation (right) settings.

lar trajectories while incorporating historical context from a dataset of expert demonstrations. This design enables robust performance from just a few expert trajectories, without requiring additional training. ReMoBot introduces two key innovations to enable data-efficient skill acquisition with strong generalization capabilities: (1) it leverages vision foundation models to extract state representations, and (2) it incorporates history-aware retrieval by enforcing trajectory similarity constraints, enabling the robot to perform complex mobile manipulation tasks in the real world. Additional details are available at: https://sites.google.com/view/remobot/home

2 Related Work

Vision-based mobile manipulation: Recent advances in visual-input-based mobile manipulation have enabled more generalizable robotic skill acquisition [9, 10]. Despite these advances, ego-centric viewpoints pose persistent challenges due to frequent occlusions, shifting perspectives, and partial observations that complicate perception and planning. Several existing methods based on end-to-end reinforcement learning [11, 12, 13], transformer [14], or modular architectures [15, 16] often struggle to generalize, handle long-horizon tasks, or explicitly address partial observability. Manipulation of deformable objects further increases these challenges due to complex dynamics and high visual variability [17, 18, 19, 20]. Consequently, robust ego-centric mobile manipulation under limited demonstrations remains an open gap.

Retrieval-based imitation learning: Retrieval-based imitation learning is a non-parametric approach where a robot learns to perform tasks by retrieving and reusing relevant data from expert demonstrations instead of learning an explicit policy. The core idea is intuitive: upon perceiving a new observation, the agent searches for the most similar observation within the dataset and executes the corresponding expert action [21, 22, 8, 23]. Previous studies, such as VINN [8], explore direct retrieval of actions using additional representation learning. In contrast, we leverage the capabilities of visual foundation models to eliminate the need for extra training. While DinoBot [7] also utilizes a visual foundation model, their method relies on pose estimation followed by visual servoing, which is impractical in mobile manipulation settings. In ego-centric views, accurate pose estimation from visual inputs is particularly challenging due to occlusions and dynamic viewpoints.

Inspired by recent efforts on decision-making based on trajectories [24, 25] or trajectory distributions [26, 27] in long-horizon tasks, we also incorporate trajectory-level information to mitigate the challenges posed by partial observations. While prior methods typically rely on learning parametric models from large-scale datasets or extensive training in simulation [28], our approach introduces a non-parametric retrieval mechanism guided by trajectory similarity constraints. This design enables

our method to operate effectively in partial observation environments with only a few demonstrations, without requiring additional model training. To the best of our knowledge, no prior work has applied a retrieval-based, training-free strategy to visual, ego-centric mobile manipulation tasks.

3 ReMoBot

In this work, we propose ReMoBot, a learning-free retrieval-based imitation method designed to efficiently solve mobile manipulation tasks with few expert demonstrations. To achieve this, we outline three main steps: 1) retrieval dataset generation, which creates a dataset by extracting visual features from the demonstrations using a vision-foundation model (VFM) based perception module; 2) retrieval process, where the agent identifies the similar expert observations and selects trajectories based on the robot executed trajectory; and 3) behavior retrieval stage, where the agent refines the retrieved behavior candidates to find the appropriate action for execution. Fig. 2 shows an overview of our framework.

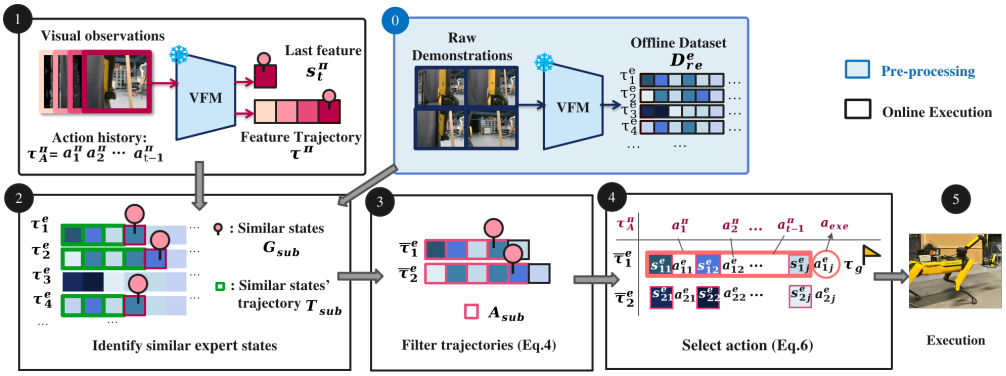

Figure 2: **Overview of ReMoBot:** An offline dataset is first processed using a pre-trained visual foundation model (VFM) to build a retrieval dataset. During execution, ReMoBot encodes RGB observations into the same feature space to identify similar expert states. These candidates are then filtered based on trajectory similarity, and the agent selects the final action by matching action history, enabling efficient training-free task execution.

3.1 Retrieval Dataset Generation

To construct the retrieval dataset, we first manually collect n demonstrations.$D_I^e = \{\tau_1^e, \tau_2^e, \ldots, \tau_n^e\}$, where each trajectory $\tau_i^e = \{(I_{i1}^e, a_{i1}^e), (I_{i2}^e, a_{i2}^e), \ldots, (I_{im_i}^e, a_{im_i}^e)\}$ contains raw RGB observations from the front-mounted camera $I_{ij}^e \in \mathbb{R}^{H\times W\times 3}$ and the corresponding discrete actions $a_{ij}^e \in \mathbb{A}$, where $\mathbb{A}$ is a set of primitive discrete actions that may involve either the body or the arm (*e.g.*, body forward, arm forward, arm grasp, etc.). We then encode these high-dimensional visual inputs into compact, task-relevant representations for downstream inference and training. The perception module leverages pre-trained vision foundation models Dinov2 [29] with frozen parameters, removing the need for additional training and enabling generalization to novel objects.

For each trajectory, we map observations into the feature space while retaining their associated actions. Let the encoder be denoted as ϕ. The resulting retrieval dataset is $D_{re}^e = \{\tau_1^e, \tau_2^e, \ldots, \tau_n^e\}$, where each trajectory $\tau_i^e = \{(s_{i1}^e, a_{i1}^e), (s_{i2}^e, a_{i2}^e), \ldots, (s_{im_i}^e, a_{im_i}^e)\}$consists of encoded features $s_{ij}^e = \phi(I_{ij}^e)$ with 384 dimensions, paired with the corresponding actions $a_{ij}^e \in \mathbb{A}$ with 1 dimension.

3.2 Retrieval Process

The retrieval stage focuses on identifying relevant trajectories from expert demonstrations, thereby aiding in imitating expert behavior for tasks. Given the current RGB observation I_t^π, the corresponding robot visual feature $s_t^\pi = \phi(I_t^\pi)$ and the historical feature trajectory $\tau^\pi =$

$\{(s_1^\pi, a_1^\pi), (s_2^\pi, a_2^\pi), \ldots, (s_t^\pi)\}$ ReMoBot filters the similar trajectory candidates from D_{re}^e using two constraints: 1) state similarity and 2) trajectory similarity, as detailed in Alg. 1.

Identify Similar States To identify the most similar demonstration from the expert dataset, we begin by constructing an initial set of individual states based on state similarity. Given the current observation feature s_t^π, we perform a nearest neighbor search based on cosine similarity $d_{cos}(s_t^\pi, s_{ij}^e)$ where we use the notation s_{ij}^e to refer to the state s_j^e from τ_i^e. We then sample the top-K most similar states to construct a state subset G_{sub}:

$$G_{\text{sub}} = \text{top-K}_{s_{ij}^e \in D_{re}^e}\big(d_{cos}(s_t^\pi, s_{ij}^e)\big), \quad (1) \qquad d_{cos}(s_t^\pi, s_{ij}^e) = 1 - \frac{s_t^\pi \cdot s_{ij}^e}{\|s_t^\pi\| \cdot \|s_{ij}^e\|} \quad (2)$$

Trajectory Similarity Filtering Due to the partial observability inherent in ego-centric visual perception, effective decision-making requires leveraging historical context. Therefore, we prioritize within the generated state subset G_{sub} those whose associated historical trajectories closely align with the robot's actual trajectory τ^π. First, for each state s_{ij}^e from G_{sub}, we retrieve its corresponding expert trajectory τ_i^e from the start upto s_j^e. More specifically, each sub-trajectory $\bar{\tau}_i^e$ is defined as: $\bar{\tau}_i^e = \{(s_{i1}^e, a_{i1}^e), (s_{i2}^e, a_{i2}^e), \ldots, (s_{ij}^e, a_{ij}^e)\}$ We denote this retrieved trajectory set as T_{sub}, which is visualized in Fig. 2 (2).

We then evaluate the similarity between the observed trajectory τ^π and the set of expert sub-trajectories T_{sub} in order to filter out dissimilar candidates. As the similarity metric, we adopt the Wasserstein distance [30], computed solely over trajectory states. This metric captures distributional alignment and has demonstrated effectiveness in imitation learning tasks [31]. Formally, the Wasserstein distance between the robot's current trajectory τ^π and an expert sub-trajectory $\bar{\tau}_i^e$ is given by Eq. 3. Where $C(\tau^\pi, \bar{\tau}_i^e)$ includes all $t \times j$ transportation matrices c that fulfill the marginal conditions, with each row summing to $\frac{1}{t}$ and each column summing to $\frac{1}{j}$. Here, c_{pq} represents the amount of mass transported from s_p^π to s_{iq}^e. d is an L2 distance function that evaluates the similarity between the robot's state in τ^π and the expert state in the expert trajectories $\bar{\tau}_i^e$, allowing us to filter out dissimilar candidates. Once we have estimated all the distance combinations $W(\tau^\pi, \bar{\tau}_i^e)$, we keep the top-$L$ similar trajectories and form the refined skill trajectory subset A_{sub}.

$$W(\tau^\pi, \bar{\tau}_i^e) = \min_{c \in C(\tau^\pi, \bar{\tau}_i^e)} \sum_{p=1}^{t} \sum_{q=1}^{j} c_{pq} \cdot d(s_p^\pi, s_{iq}^e) \quad (3) \qquad A_{\text{sub}} = \text{top-L}^{\min}_{\bar{\tau}_i^e \in T_{sub}}\big(W(\bar{\tau}_i^e, \tau^\pi)\big) \quad (4)$$

3.3 Action Selection

Now we have the refined set A_{sub}, where each trajectory's ending state s_{ij}^e matches the robot's current state s_t^π and whose sequences align with its observations. Their final actions a_{ij}^e define feasible candidates. Under ego-centric partial observability, identical observations may yield different motions. To resolve this, ReMoBot exploits the intuition that similar trajectories share similar action histories, selecting actions via reversed Hamming distance [32], which rewards alignment with the robot's past actions.

More specifically, denote the robot's action sequence as $\tau_A^\pi = \{a_1^\pi, a_2^\pi, ..., a_{t-1}^\pi\}$. For each $\bar{\tau}_i^e \in A_{sub}$, we formulate its corresponding action history sequence as $\tau_{iA}^e = \{a_{i(j-t+1)}^e, a_{i(j-t+2)}^e, ..., a_{i(j-1)}^e\}$ where j is the index of the last action in $\bar{\tau}_i^e$. We then compute the matching score and select the trajectory τ_g with the highest matching score:

$$\tau_g = \arg\max_{\bar{\tau}_i^e \in A_{\text{sub}}} d_H(\tau_{iA}^e, \tau_A^\pi) \quad (5) \qquad d_H(\tau_{iA}^e, \tau_A^\pi) = \sum_{k=1}^{t-1} \mathbf{1}\left(a_{i(j-t+k)}^e, a_k^\pi\right) \quad (6)$$

where a_k^π is the action of trajectory τ^π at the k-th timestep and $a_{i(j-t+k)}^e$ is the action of trajectory τ_i^e at timestep $(j-t+k)$. $\mathbf{1}$ is an indicator function that equals 1 for identical actions and 0 otherwise.

Consequently, τ_g is the retrieved expert sub-trajectory that 1) its last visual observation matches the robot's current observation, 2) has visually similar historical observations, and 3) makes similar historical action decisions as the robot. ReMoBot then retrieves the last action of τ_g as the feasible action a_{exe} to execute. We present the ReMoBot algorithm in Alg. 1

4 Experiments

We evaluate ReMoBot on complex mobile manipulation tasks and compare it against several state-of-the-art baselines. Our experiments are designed to answer the following key questions: (1)How does ReMoBot compare to learning-based and retrieval-based baselines?(2)Can ReMoBot generalize to variations in initial pose, object size, and material? (3)How well does ReMoBot perform under limited data? (4)What are the effects of state and trajectory constraints, and how do the hyperparameters K and L influence performance?

4.1 Mobile Manipulation Tasks

To demonstrate ReMoBot's capability to handle complex ego-centric observations, we designed three mobile manipulation tasks: Table Uncover, Gap Cover, and Curtain Open. These tasks present perception challenges due to fabric deformability and partial observability from a front-mounted RGB camera, highlighting the need for decision-making under uncertainty. For the Table Uncover and Gap Cover tasks, the main difficulty arises from dataset imbalance, as each trajectory contains only a single bottleneck GRASP action. The discrete action space further increases the challenge for learning-based methods. Task illustrations are shown in Fig. 1. The complete feature extraction and inference pipeline runs at 15 frames per second on an NVIDIA RTX 3080 GPU and an AMD Ryzen 5000-series CPU, enabling real-time decision-making during deployment. Real experiments are performed on a Boston Dynamics Spot robot, while simulation experiments are performed on the same robot in an Isaac Sim environment. We used the visual foundation model Dinov2 [29] with frozen parameters as the perception module.

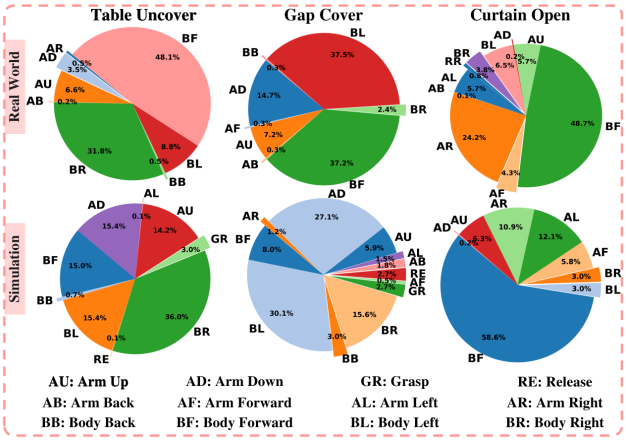

Figure 3: **Action distribution in the dataset .** Each pie chart illustrates the frequency of discrete actions. The rows correspond to real-world and simulation results, respectively. Tasks are ordered as Table Uncover, Gap Cover, and Curtain Open.

Table Uncover: In this task, the robot approaches a table and removes a cloth covering it by folding and pulling it sideways. The task is considered complete when the folded cloth's edge crosses the center of the table.

Gap Cover: In this task, the robot first approaches a cloth, grasps it, and then uses it to cover a gap between two objects. The gap is positioned such that successful coverage requires coordinated body movement. The task is considered complete when one edge of the cloth fully surpasses the gap.

Curtain Open: In this task, the robot approaches a curtain, uses its arm to push the curtain aside, and then navigates its body through the opening. The task is considered successful when the curtain is sufficiently opened and the robot moves past the curtain hanger. Although this task does not involve grasping, it introduces collision risk.

Dataset Collection: We first collect expert demonstrations to teach the robot to complete the target tasks. Demonstrations are obtained via human teleoperation using a discrete action space com-

posed of body and arm movement primitives. Additional implementation details are provided in Section A.3.3

For each task, we collect 20 demonstrations in simulation and real-world settings, respectively. As shown in Fig. 3, the action distribution is highly imbalanced. This imbalance poses a significant challenge for policy learning, as the agent must acquire competence in infrequent but essential actions (*e.g.*, Grasp) despite limited training data. Table 1 summarizes dataset statistics, highlighting both the variability across tasks and the challenges imposed by data imbalance.

Table 1: **Dataset Details.** Average trajectory length (mean $\pm$ standard deviation) over 20 demonstrations per task, along with the total number of samples.

	Simulation		**Real world**	
Task	**Length**	**Samples**	**Length**	**Samples**
Uncover	33.75 ± 7.50	675	30.95 ± 3.46	619
Cover	37.1 ± 6.99	742	13.95 ± 1.79	279
Curtain	42.85 ± 6.90	875	41.1 ± 5.97	822

4.2 Baseline Comparisons

For baseline selection, we include both learning-based and retrieval-based approaches to provide a comprehensive comparison. Learning-based baselines consist of classical Behavior Cloning (BC) [6] as well as state-of-the-art architectures such as the Action-Chunk Transformer [33] and Diffusion models [34], which represent the current frontier of parametric imitation learning. Retrieval-based baselines include GSR [35] and Visual Imitation through Nearest Neighbors (VINN) [8]. Additional implementation details and results are provided in Section A.3.4.

We first set up the three tasks in the simulator to evaluate our method and all five baselines. We then train selected methods on a real-world dataset and compare their performance with ReMoBot in real-world deployment. Notably, all methods are evaluated in simulation and in the real-world separately, without any sim-to-real transfer.

Table 2: **Baseline Comparisons in Simulation.** Success rates (*success/total trials*) across three tasks. The underlined entry denotes the best-performing method for each task. Bold indicates methods whose performance is not significantly different from ours (Fisher's exact test, $p \geq 0.05$).

	VINN	**Diffusion**	**GSR**	**BAKU**	**BC**	**Ours**
Uncover	0/40	0/40	6/40	**38/40**	5/40	**36/40**
Cover	0/40	0/40	0/40	**33/40**	15/40	**31/40**
Curtain	0/40	15/40	12/40	13/40	16/40	**32/40**

Baseline Comparisons in Simulation: Although all six methods generally navigate the robot close to the target (e.g., the curtain or sheet), Diffusion and VINN perform worse across most tasks (Table 2). We hypothesize that 20 demonstrations($\sim$ 600 samples per task) are insufficient, compared to prior work: Diffusion [34] uses over 200 demonstrations, and VINN [8] uses 71 for training.

Beyond dataset limitations, we observe two main failure modes of learning-based methods. First, severe action imbalance biases models toward frequent actions and hinders rare but crucial ones, such as GRASP (only 2.7% and 3.0% in Cover and Uncover). Second, strong visual similarity between pre- and post-grasp states confuses models. Supporting this, manually replacing BC's second grasp with an arm-up command yielded 33/40 successes, confirming our assumption.

In contrast, both ReMoBot and BAKU leverage historical information, enabling them to achieve robust performance in this data-constrained setting. ReMoBot achieves competitive performance across all tasks, including the more visually complex Curtain scenario, demonstrating that retrieval-based imitation can generalize beyond specific bottleneck actions.

Baseline Comparisons in the Real World: For real-world evaluation, we retrain BAKU and Behavior Cloning (BC) on the real-world dataset. BAKU is included as the strongest simulation baseline (Tab. 2), while BC serves as a lightweight supervised baseline. Although BC lacks sequential rea-

Table 3: **Baseline Comparisons in Real World.** Success rates (*success/total trials*) of all baselines across three tasks. Underline indicates the best-performing method for each task. Bold indicates methods that are not significantly different from ours (Fisher's exact test, $p \geq 0.05$).

Tasks	**Table Uncover**			**Gap Cover**			**Curtain Open**		
Task Stages:	Approach	Grasp	Uncover	Approach	Grasp	Cover	Approach	Open	Pass
BC	10/20	1/20	0/20	13/20	4/20	2/20	**16/20**	2/20	0/20
BAKU	**19/20**	0/20	0/20	10/20	0/20	0/20	6/20	0/20	0/20
ReMoBot	**20/20**	15/20	14/20	20/20	17/20	16/20	**20/20**	16/20	9/20

soning, it provides a useful lower bound for performance under limited supervision. Together, these baselines offer a balanced comparison: BAKU as the best-performing advanced method and BC as the simplest direct imitation approach.

Similar to the simulation, as shown in Tab. 3, both baselines are generally able to navigate the robot close to the target. However, despite being trained on the real-world dataset with the same network structure, BAKU's performance drops significantly. We also identify the same two failure modes as in the simulation. First, in the Cover and Uncover tasks, the model often fails to issue the correct GRASP action, as the observations are nearly indistinguishable (Fig. 4). This results in repeated body motions without executing the grasp, preventing task completion.Second, we hypothesize that the decline in performance is primarily caused by environmental noise and a perception distribution shift between the data collection and evaluation environments. In addition, once the robot moves, the perception of the scene is not identical across trials, as small differences in the robot's trajectory lead to variations in viewpoint and observation.

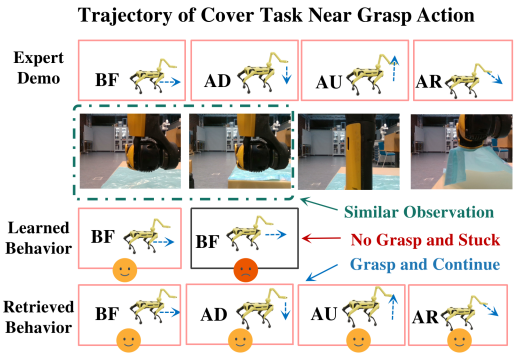

Figure 4: **Visual ambiguity.** Rows show (top to bottom): expert demonstration, egocentric observations, learned behavior, and retrieved behavior (BF: move forward, AD: arm down, AU: arm up, AR: arm right).

In contrast, ReMoBot consistently outperforms the baselines, achieving success rates of 70% for Table Uncover, 80% for Gap Cover, and 45% for Curtain Open. A detailed breakdown in Table 3 shows that ReMoBot is not only able to reach the target reliably (100% success in the Close stage across all tasks), but also executes rare but essential grasping and following actions with high accuracy. The Curtain Open task remains the most challenging due to fabric deformability, partial observability, and collision, yet ReMoBot still demonstrates improvements over the baselines.

4.3 Generalizability Evaluation

Table 4: **Generalizability Evaluation.** Success rates (*success/total trials*). Underline values indicate the best-performing condition.

	Uncover	**Cover**	**Curtain**
Size	10/20	10/20	6/20
Material	12/20	11/20	6/20
Position	15/20	12/20	7/20
Default	14/20	16/20	9/20

We evaluate ReMoBot in three different settings to demonstrate its generalization capability: (1) varying object sizes, (2) different fabric materials, and (3) varying initial positions of the robot. We conduct this evaluation on the real robot with 20 demonstrations. For each scenario, we vary one factor while keeping all other configurations consistent with the data collection environment. Detailed configurations are provided in Section A.3.2. Table 4 shows that ReMoBot maintains robust performance across diverse generalization scenarios. Minor performance drops are primarily due to incorrect expert trajectory retrieved, when the target

object is partially or entirely outside the camera's field of view, especially with larger materials that occlude the scene. Despite these challenges, across all tasks, none of the evaluated conditions show statistically significant differences from the default condition (Fisher's exact test, $p \geq 0.05$), indicating that ReMoBot demonstrates good generalization capabilities across varying fabric materials, object sizes, and initial robot positions.

4.4 Data Efficiency Evaluation

To investigate the data efficiency of ReMoBot, we conducted experiments using varying dataset sizes of 1, 5, 10, 15, and 20 demonstrations in simulation only. The evaluation environment is identical to the demonstration collection environment. Results are reported as the success rate of 40 trials. As shown in Table 5, for simpler tasks such as Table Uncover, ReMoBot reaches around 80% success with only 15 demonstrations, while for more challenging tasks, performance exceeds 75% with 20 demonstrations.

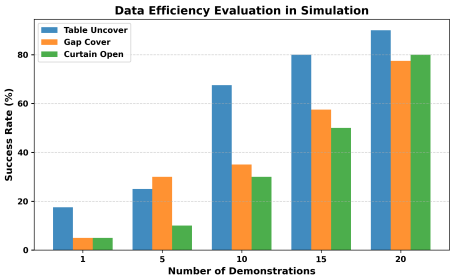

Figure 5: **Data Efficiency Evaluation.** Success rates for 40 trials with different numbers of demonstrations.

4.5 Ablation Study

Table 5: **Ablation study in simulation.** Success rates (*success/total trials*) under different hyperparameter settings. Underline values indicate the best-performing condition for each task. NA indicates no trajectory constraints.

Task	K=10 L=NA	K=10 L=5	K=10 L=3	K=5 L=5	K=5 L=3
Uncover	30/40	<u>36/40</u>	30/40	21/40	32/40
Cover	17/40	<u>31/40</u>	30/40	28/40	30/40
Curtain	26/40	<u>32/40</u>	24/40	27/40	25/40

We conduct an ablation study to evaluate the impact of two different similarity constraints hyperparameters used in the retrieval process. K denotes the size of the subset G_{sub} in Eq. 1, L is the size of A_{sub} in Eq. 4. The evaluation is performed in the simulated environment identical to the data collection environment. Table 5 shows that incorporating trajectory similarity consistently improves performance across all tasks, highlighting its importance. The best results are obtained with $K = 10$ and $L = 5$, which we used in all previous experiments.

5 Conclusion

Learning mobile manipulation skills for complex tasks, such as partial observation mobile manipulation, from a few demonstrations is a challenging problem. This work introduces ReMoBot, a few-shot imitation learning framework that leverages a retrieval strategy with visual similarity constraints to solve tasks without additional training. ReMoBot integrates a visual foundation model as a feature extractor with a trajectory-aware action identification, enabling training-free imitation of expert demonstrations even under partial observability. To evaluate ReMoBot, we designed three real-world mobile manipulation tasks involving deformable fabrics with the Boston Dynamics Spot robot. Across all tasks, ReMoBot consistently outperformed both learning-based and retrieval-based baselines, effectively acquiring manipulation skills from a limited dataset. Furthermore, ReMoBot demonstrated generalization to varying environmental conditions, including robot initial position, object size, and materials. Moving forward, extending ReMoBot with explicit mechanisms for collision handling and incorporating online fine-tuning strategies could further enhance its adaptability and safety during deployment, while preserving data efficiency and generalizability.

References

[1] M. Luo, Z. Xue, A. Dimakis, and K. Grauman. Put myself in your shoes: Lifting the egocentric perspective from exocentric videos. In *European Conference on Computer Vision*, pages 407–425. Springer, 2024.

[2] Y. Hu, B. Chen, and H. Lipson. Egocentric visual self-modeling for autonomous robot dynamics prediction and adaptation. *arXiv preprint arXiv:2207.03386*, 2022.

[3] C. Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022.

[4] T. Ni, K. Ehsani, L. Weihs, and J. Salvador. Towards disturbance-free visual mobile manipulation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5219–5231, 2023.

[5] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.

[6] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[7] N. D. Palo and E. Johns. Dinobot: Robot manipulation via retrieval and alignment with vision foundation models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[8] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto. The surprising effectiveness of representation learning for visual imitation, 2021.

[9] Y. Gong, G. Sun, A. Nair, A. Bidwai, R. CS, J. Grezmak, G. Sartoretti, and K. A. Daltorio. Legged robots for object manipulation: A review. *Frontiers in Mechanical Engineering*, 9: 1142421, 2023.

[10] S. Thakar, S. Srinivasan, S. Al-Hussaini, P. M. Bhatt, P. Rajendran, Y. Jung Yoon, N. Dhanaraj, R. K. Malhan, M. Schmid, V. N. Krovi, et al. A survey of wheeled mobile manipulation: A decision-making perspective. *Journal of Mechanisms and Robotics*, 15(2):020801, 2023.

[11] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese. Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation. *arXiv preprint arXiv:2008.07792*, 2020.

[12] A. Gupta, M. Zhang, R. Sathua, and S. Gupta. Opening articulated objects in the real world, 2025. URL https://arxiv.org/abs/2402.17767.

[13] Z. Fu, X. Cheng, and D. Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023.

[14] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

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

[16] 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. *IEEE Robotics and Automation Letters*, 9(1):779–786, 2024. doi:10.1109/LRA.2023.3336109.

[17] J. Hietala, D. Blanco-Mulero, G. Alcan, and V. Kyrki. Learning visual feedback control for dynamic cloth folding. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1455–1462. IEEE, 2022.

[18] F. Zhang and Y. Demiris. Visual-tactile learning of garment unfolding for robot-assisted dressing. *IEEE Robotics and Automation Letters*, 2023.

[19] B. Frank, C. Stachniss, R. Schmedding, M. Teschner, and W. Burgard. Real-world robot navigation amongst deformable obstacles. In *2009 IEEE International Conference on Robotics and Automation*, pages 1649–1654, 2009. doi:10.1109/ROBOT.2009.5152275.

[20] J. Hu, W. Liu, H. Zhang, J. Yi, and Z. Xiong. Multi-robot object transport motion planning with a deformable sheet. *IEEE Robotics and Automation Letters*, 7(4):9350–9357, 2022.

[21] D. Sharon and M. van de Panne. Synthesis of controllers for stylized planar bipedal walking. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2387–2392. IEEE, 2005.

[22] E. Mansimov and K. Cho. Simple nearest neighbor policy method for continuous control tasks. 2018.

[23] E. Valassakis, G. Papagiannis, N. Di Palo, and E. Johns. Demonstrate once, imitate immediately (dome): Learning visual servoing for one-shot imitation learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8614–8621. IEEE, 2022.

[24] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[25] S. Haldar, V. Mathur, D. Yarats, and L. Pinto. Watch and match: Supercharging imitation with regularized optimal transport. In *Conference on Robot Learning*, pages 32–43. PMLR, 2023.

[26] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[27] S. Yan, Z. Zhang, M. Han, Z. Wang, Q. Xie, Z. Li, Z. Li, H. Liu, X. Wang, and S.-C. Zhu. M 2 diffuser: Diffusion-based trajectory optimization for mobile manipulation in 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[28] M. Lauri, D. Hsu, and J. Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

[29] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2(3):4, 2021.

[30] C. Villani. The wasserstein distances. In *Optimal transport: old and new*, pages 93–111. Springer, 2009.

[31] R. Dadashi, L. Hussenot, M. Geist, and O. Pietquin. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*, 2020.

[32] Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[33] S. Haldar, Z. Peng, and L. Pinto. Baku: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024.

[34] 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. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[35] Z.-H. Yin and P. Abbeel. Offline imitation learning through graph search and retrieval. *arXiv preprint arXiv:2407.15403*, 2024.

[36] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[37] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[38] X. Hu, qiang liu, X. Liu, and B. Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ugXKInqDCC.

[39] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[40] J. Pari, M. Shafiullah, S. Arunachalam, and L. Pinto. Visual imitation through nearest neighbors (vinn) implementation. https://github.com/jyopari/VINN/tree/main, 2021.

A Appendix

A.1 Task in Simulation

To further analyze failure cases and reduce noise present in real-world experiments, we replicate the same task in the simulation with the Isaac Sim simulator. This enables the comparison of different baselines under controlled conditions. As shown in Fig. 6, we recreate the tasks using the Spot robot in simulation with alignment to the real-world setup. Specifically, we position the camera to match its placement in the real-world setting, ensuring consistent ego-centric observations across both domains. The cloth is simulated using a particle-based system, which provides photorealistic visuals and physically accurate cloth dynamics.

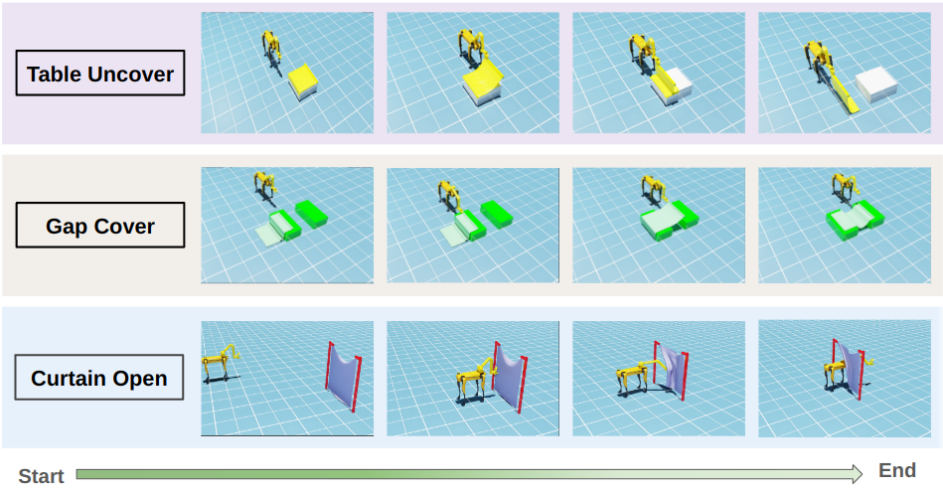

Figure 6: **Three Deformable Mobile Manipulation tasks in Simulation.** Table Uncover (top), Gap Cover (middle), and Curtain Open (bottom) tasks. To solve these three tasks, a Spot robot with a front-mounted RGB-D camera must coordinate body movement and arm operation to navigate and manipulate fabric. Note that we collect demonstrations and evaluate all the methods directly in the simulation.

Table Uncover: In this task, the robot must approach the table and remove the table cover by pulling it in a specific direction, causing the cloth to fold first. The task is considered successful when the cloth is folded and its edge crosses the center of the table. To reduce irrelevant sources of difficulty, we assume that all grasp actions succeed without slippage and collision.

Gap Cover: This task requires the robot to get close to the deformable target first and then use it to cover the gap between the two objects. Completion of the task is defined by one of the cloth's edge surpassing the entire gap. We also assume that all grasp actions succeed in reducing irrelevant difficulty.

Curtain Open: In this task, the robot is required to approach the curtain, use its arm to move the curtain aside, and then navigate through the opening. The task introduces additional difficulty by incorporating a collision-avoidance requirement. Success is defined as the curtain being opened and the robot's body moving past the curtain hanger. Notably, grasp actions are excluded from the action space for this task.

A.2 ReMoBot Algorithm

The algorithm of ReMoBot is described in Alg. 1

Algorithm 1 ReMoBot

Initialize:
Given expert retrieval dataset D^e_{re};
Online visited trajectory τ^π, current feature state s^π_t;
Empty buffers G_{sub} and T_{sub};
Step 1: Identify Similar States
Retrieve top-K similar expert states as G_{sub} (Eq.1).
Step 2: Extract Corresponding Trajectories
for Each $s^e_{ij} \in G_{\text{sub}}$ **do**
Retrieve the expert trajectory τ^e_i where $s^e_{ij} \in \tau^e_i$.
Truncate τ^e_i at timestamp j.
Store truncated $\bar{\tau}^e_i$ in T_{sub}.
Step 3: Evaluate Candidate Trajectories
Select top-L similar expert trajectories from T_{sub} to form a refined set A_{sub} (Eq.4).
Step 4: Action Selection
for Each $\tau^e_{iA} \in A_{\text{sub}}$ **do**
Compute the action similarity score (Eq. 6).
Select the $\bar{\tau}^e_i$ with highest score as τ_g.
Execute the last action a^e_{ij} from τ_g.

A.3 Experiment

A.3.1 Task Configuration

For both simulation and real-world settings, we use the Boston Dynamics Spot robot to conduct tasks in a laboratory environment. Observations consist of RGB-D inputs from a front-mounted camera on the robot body $O_t = I_{\text{rgb}}$. The action space is discrete and consists of body and arm movement primitives. At each timestep, the policy selects one action from a fixed set of commands. These include body-level motions that translate the robot base by a fixed distance in the environment. Similarly, arm-level actions that move the end-effector in Cartesian space by a fixed step size along the corresponding axis. The policy is restricted to issuing either a body or an arm command at each step, resulting in a decoupled control scheme. This discretization enables efficient policy learning and simplifies integration with retrieval-based planning. The detailed action definitions for each task can be found in A.3.3. All manipulated objects are deformable, increasing the complexity of visual observations. The properties of deformable objects for each task are listed below:

- **Table Uncover:** In this setting, the table is big enough to require both body and arm movements to complete the task. We use a 75 cm $\times$ 110 cm black plastic cloth as the table cover and place it on a bigger stage.
- **Gap Cover:** In this setup, we use a 55 cm $\times$ 110 cm blue plastic cloth and set the gap size as 50 cm.
- **Curtain Open:** The curtain is a 130 cm $\times$ 240 cm gray polyester cloth, with a distance of 1.5 meters between the two hangers.

To analyze the performance in the controllable environment, we replicate the tasks in the simulator. The deformable fabric is simulated with the particle system. The resolution is 50 with 0.01 kg for each particle.

- **Table Uncover:** In this setting, the table is big enough to require both body and arm movements to complete the task. We use a 100 cm $\times$ 100 cm cloth with a rigid handle as the table cover and place it on the same size stage.
- **Gap Cover:** In this setup, we use an 80 cm $\times$ 120 cm cloth with a rigid handle as the table cover and set the gap size as 90 cm.
- **Curtain Open:** The curtain is 110 cm $\times$ 110 cm, with a distance of 130 cm between the two hangers.

A.3.2 Generalizability Configuration

We conduct a comprehensive evaluation of ReMoBot's generalization capabilities under three conditions in both simulation and the real world: (1) varying initial robot positions, (2) different deformable object materials, and (3) diverse object sizes. For each scenario, we vary only one factor at a time while keeping all other configurations consistent with the data collection environment. Detailed experimental settings are provided below.

- **Position:** We expand the range of robot initial positions: keep the distances from 1.5 to 1.8 meters from the curtain, change the lateral displacements up to 1 meter from the curtain's center, and angular variations between -20 and 20 degrees. For these position-based generalization tests, all other environmental parameters remained consistent with the demonstration collection setup.
- **Material:** This experiment assessed the system's adaptability to different fabric characteristics. In the real-world environment, we evaluated performance using a mixed fiber (cotton and polyester) cloth and a blue plastic cover, neither of which is utilized during demonstration collection.
- **Curtain Size:** To evaluate the influence of curtain dimensions on ReMoBot's performance, we conducted tests with two additional curtain sizes not used in the demonstrations. For the curtain-open task, we tested a smaller curtain measuring 80 cm × 110 cm and a larger one measuring 160 cm × 90 cm. For the Cover and Uncover tasks, we employed a smaller curtain of 80 cm × 80 cm and a larger variant of 80 cm × 160 cm.

In the simulation, the detailed experimental setting is listed below: Detailed experimental settings are provided below.

- **Position:** We expand the range of robot initial positions: keep the distances from 0 to 1.8 meters from the curtain, change the lateral displacements up to 1 meter from the curtain's center, and angular variations between -20 and 20 degrees. For these position-based generalization tests, all other environmental parameters remained consistent with the demonstration collection setup.
- **Material:** This experiment assessed the system's adaptability to different fabric characteristics. We randomized the damping parameters in the computed springs for the fabric between 0.05 and 0.35 to model various material properties.
- **Curtain Size:** To evaluate the influence of curtain dimensions on ReMoBot's performance, we tested a smaller curtain measuring 80 cm × 110 cm and a larger one measuring 160 cm × 90 cm. For the Cover and Uncover tasks, we employed a smaller curtain of 80 cm × 80 cm and a larger variant of 80 cm × 160 cm.

For all tasks, the Spot robot's initial position is randomized within a range of 0 to 1.8 meters away from the deformable objects, with lateral deviations of up to 0.8 meters to the left or right of the fabric or gap center. Additionally, angular deviations range from -15 to 15 degrees relative to the center. We use this experimental setup as the default condition during the comparison. The demonstrations are conducted by a human operator via remote control.

A.3.3 Demonstration Collection

Real-world Setting: We collected 20 demonstrations for each task separately in real-world with the keyboard controller. An RGB-D camera is mounted on the front of the robot's body to receive egocentric observations. The robot has the following discrete actions implemented: 1) body move-forward, 2) body move-left, 3) body move-right, 4) body move-backward, 5) body turn-left, 6) body turn-right, 7) hand move-forward, 8) hand move-backward, 9) hand move-left, 10) hand move-right, 11) hand move-up, 12) hand move-down. Note that in the Gap Cover and Table Uncover tasks, the object is considered to be grasped when the end-effector touches the handle of the deformable object. For all tasks, the Spot robot's initial position is randomized within a range of 1.5 to 1.8

meters away from the deformable objects, with lateral deviations of up to 1 meter to the left or right of the fabric or gap center. Additionally, angular deviations range from -15 to 15 degrees relative to the center. We use this experimental setup as the default condition during the comparison. The demonstrations are conducted by a human operator via remote control.

Simulation setting: We also collected 20 demonstrations for each task separately in the simulation with the keyboard controller. An RGB-D camera is mounted on the front of the robot's body to receive egocentric observations. The robot has the following discrete actions implemented: 1) body move-forward, 2) body move-left, 3) body move-right, 4) body move-backward, 5) body turn-left, 6) body turn-right, 7) hand move-forward, 8) hand move-backward, 9) hand move-left, 10) hand move-right, 11) hand move-up, 12) hand move-down, 13) hand grasping, and 14) hand release. Note that in the Gap Cover and Table Uncover tasks, the object is considered to be grasped as long as the grasp action is executed. For all tasks, the Spot robot's initial position is randomized within a range of 0 to 1.8 meters away from the deformable objects, with lateral deviations of up to 0.8 meters to the left or right of the fabric or gap center. Additionally, angular deviations range from -15 to 15 degrees relative to the center. We use this experimental setup as the default condition during the comparison. The demonstrations are conducted by a human operator via remote control.

A.3.4 Baseline Implementation

BC: A classical supervised learning approach [36, 37], where a policy is trained to directly map observations to actions using expert demonstrations. In our setup, to make the comparison fair, we train the policy to predict one-step actions just like our methods.

GSR: A retrieval-based method [35] that organizes the dataset into a graph and performs graph search to estimate the values of different behaviors. A retrieval procedure is then applied to identify the best behavior (action) for each state, followed by behavior cloning to learn that behavior. For simplicity, we adapt the original diffusion-based behavior cloning approach to a multi-layer perceptron (MLP).

BAKU: A transformer-based behavior cloning method that inputs the history of the last h observations $s_{t-h:t}$ and predicts a chunk of h actions with a Gaussian mixture model. Following [33], we calculate the multi-step action loss with $h = 5$ but only execute the first one during evaluation.

Diffusion: Diffuser leverages diffusion probabilistic models to generate trajectories that mimic expert behavior [26, 38]. While previous work has focused on large-scale datasets and point cloud inputs [39], we implement a version based solely on RGB observations following the Diffusion Policy framework [34].

VINN: VINN performs nearest neighbor search over demonstration observations to retrieve the most similar states [8], and computes an action as a Euclidean kernel-weighted average of those associated with the retrieved neighbors. We use the original VINN encoder structure [40], a visual representation model BYOL, with our dataset to compare against our visual perception pipeline.

A.3.5 Generalization Evaluation in Simulation

We also evaluate ReMoBot in three different settings in simulation to demonstrate its generalization capability without the real-world noise.

For each scenario, we vary only one factor at a time while keeping all other configurations consistent with the data collection environment. The results in Table 6 demonstrate that ReMoBot maintains robust performance across diverse generalization scenarios, aligning well with the real-world experimental results and further validating the robustness of ReMoBot.

B Limitations

Despite the promising results of ReMoBot, several limitations remain. First, the absence of collision-free motion planning led to frequent failures in the curtain-opening task, where the robot occasion-

Table 6: **Generalizability evaluation of DeMoBot.** The success rates of the three tasks are evaluated across varying initial positions of the robot, materials and sizes of the deformable fabric in the simulation. 'Default' means the default environmental condition.

	Gap Cover	**Table Uncover**	**Curtain Open**
Materials	32/40	32/40	32/40
Position	27/40	32/40	28/40
Size	25/40	28/40	30/40
Default	32/40	35/40	32/40

ally collided with the curtain hanger. Second, the retrieval module sometimes selected visually distinct states due to the limited representation power of the foundation model and the restricted diversity of the offline dataset—an issue common in imitation learning without online adaptation. Third, in the Table Uncover and Gap Cover tasks, highly similar observations occasionally caused local optima, suggesting the need for additional contextual signals such as a grasp flag. Finally, the decoupled control of the robot's body and arm introduced IK-related failures. Future work may address these limitations by integrating collision-free planning, incorporating online adaptation and failure recovery mechanisms, adding contextual signals (*e.g.*, grasp flags), and exploring more unified control schemes or continuous skill-level action spaces.