ReMoBot: Mobile Manipulation with Vision-based Sub-goal Retrieval

Anonymous Author(s)
Affiliation
Address
email

Abstract: Imitation learning (IL) algorithms typically distill experience into parametric behavior policies to mimic expert demonstrations. With a limited set of demonstrations, previous methods often cannot accurately align the current state with expert demonstrations, especially under partial observability. 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 a sub-goal, considering visual similarity w.r.t. both single observations and trajectories. A motion generation policy subsequently guides the robot toward each selected sub-goal, iteratively progressing until the task is successfully completed. We design three mobile manipulation tasks and evaluate ReMoBot on these tasks with a Boston Dynamics Spot robot. With only 20 demonstrations, ReMoBot outperforms baseline methods, achieving high success rates in Table Uncover (70%) and Gap Cover (80%) tasks, while showing promising performance on the more challenging Curtain Open task (35%). Moreover, ReMoBot generalizes to varying robot positions, object size, and material type. Additional details are available at: https://sites.google.com/view/remobot/home

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

1 Introduction

Learning mobile manipulation purely from egocentric visual inputs is particularly 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, RL typically demands extensive exploration [3, 4], making it impractical for real-world applications without additional guidance. In contrast, imitation learning (IL) has succeeded in various complex robotic tasks [5], enabling robots to quickly acquire skills from expert demonstrations. However, the performance of IL methods highly depends on the quantity and diversity of demonstrations, posing significant challenges. Furthermore, IL methods such as behavior cloning suffer from compounding errors over long task horizons [6]. 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 similar observations from a collected dataset of demonstrations, enabling robust performance with a few expert trajectories.

ReMoBot introduces two key innovations to enable data-efficient skill acquisition with strong generalization capabilities: (1) it leverages vision foundation models to extract object-centric state representations, and (2) it incorporates history-aware retrieval by enforcing trajectory similarity constraints, enabling consistent sub-goal generation to guide the robot through complex mobile manipulation tasks in the real world directly.

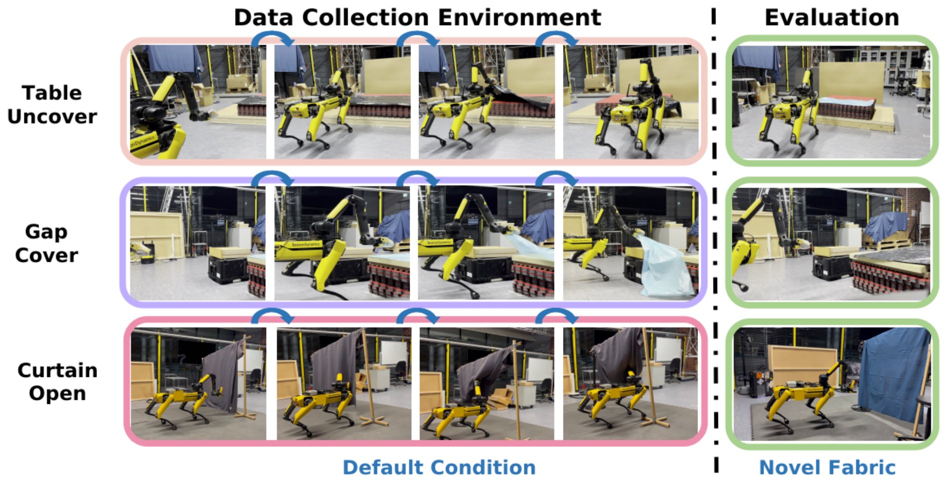

Figure 1: **Three Deformable Mobile Manipulation tasks.** Table Uncover (top), Gap Cover (middle), and Curtain Open (bottom) are shown in both the data collection (left) and novel fabric evaluation (right) settings.

2 Related Work

Vision-based mobile manipulation: Recent progress in visual-input-based mobile manipulation has enabled more generalizable and scalable robotic skill acquisition [7, 8]. However, ego-centric viewpoints present persistent challenges due to frequent occlusions, dynamically shifting perspectives, and a limited field of view. These factors lead to partial observations, which significantly complicate both perception and planning. Several existing methods address these challenges through end-to-end reinforcement learning [9, 10, 11] or modular architectures [10, 12, 13]. Furthermore, manipulation of deformable objects adds an additional layer of complexity [14, 15], demanding robustness to complex dynamics and high visual variability [16, 17].

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 [18, 19, 20, 21]. Previous studies, such as VINN [20], 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 [22] 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 to make decisions based on trajectories [23, 24] or trajectory distributions [25, 26] 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 [27], our approach introduces a non-parametric retrieval mechanism guided by trajectory similarity constraints. This design enables our method to operate effectively in the context of partial observation environments with only a few demonstrations and without 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 retrieval-based method designed to efficiently solve complex 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-based perception module; 2) Sub-goal generation, which encodes current observations to identify a state from the expert demonstrations as a sub-goal, guiding the robot towards task completion; and 3) A goal-conditioned behavior retrieval policy, which selects the appropriate action to be executed. Fig. 2 shows an overview of our framework.

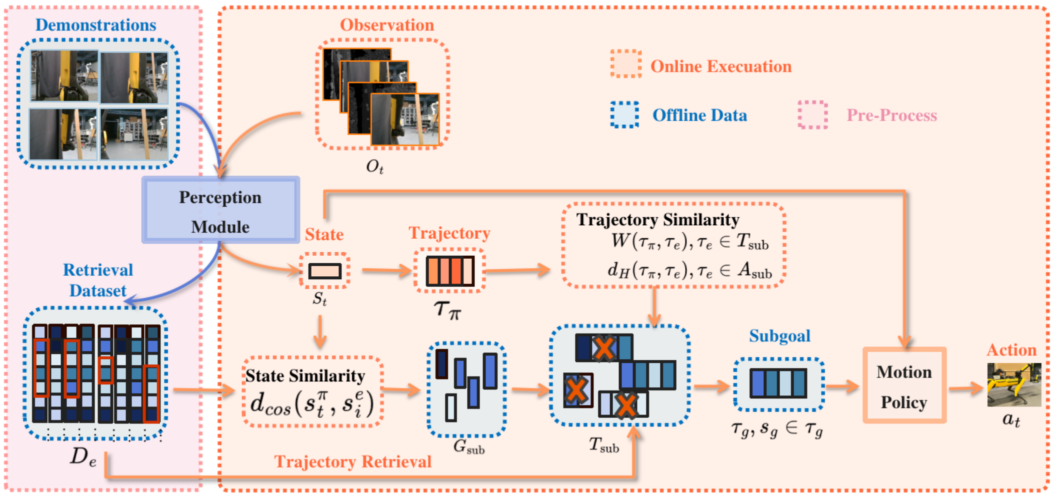

Figure 2: **Overview of Our Pipeline:** We first process an offline dataset using a pre-trained perception module to create a retrieval dataset. During execution, ReMoBot maps RGB-D observations into the same feature space as the retrieval dataset to retrieve similar states. ReMoBot selects a sub-goal based on trajectory similarity, and the motion generation module produces control commands conditioned on the current state to reach the retrieved sub-goal.

3.1 Retrieval Feature Generation

ReMoBot initiates the inference process with a retrieval feature generation module, which encodes high-dimensional visual input from the camera into compact, task-relevant representations for downstream inference and training. The perception module leverages pre-trained vision foundation models, eliminating the need for additional training while enabling generalization to novel objects.

To generate retrieval features, we first construct prompts for task-relevant entities, for example, the robot embodiment and the manipulated objects. These prompts are used as input to the segmentation module [28, 29, 30] to continuously segment and track relevant objects over time. The resulting segmentation masks are passed through a frozen, pre-trained DINO model [31] to extract a visual feature vector s_t^π with 384 dimensions. The complete feature extraction and inference pipeline operates 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. Further details on the perception pipeline are provided in Section 7.1. Before the online execution process, we first leverage the generation module to construct a retrieval dataset from the offline expert demonstrations. For each trajectory in the demonstrations, we encode the observation into the feature space while retaining the original action. The resulting retrieval dataset is:

$$D_e = \{\tau_1^e, \tau_2^e, \ldots, \tau_n^e\}$$

where each $\tau_i^e = \{(s_1^e, a_1^e), (s_2^e, a_2^e), \ldots, (s_n^e, a_n^e)\}$ consists of encoded feature and their corresponding actions, with each action representing a control command executed by the robot. Further details on data collection, observation spaces, and action spaces are provided in Section 7.3.1.

3.2 Sub-goal Retrieval

Sub-goal generation focuses on identifying relevant data items from expert demonstrations to serve as intermediate targets, thereby aiding decision-making. Given current robot state s_t^π and the observed trajectory so far, $\tau_\pi = \{(s_1^\pi, a_1^\pi), (s_2^\pi, a_2^\pi), \ldots, (s_t^\pi)\}$, ReMoBot determines the appropriate sub-goal from D_e using two constraints: 1) state similarity, 2) trajectory similarity as detailed in Algorithm 1.

Algorithm 1 Sub-goal Retrieval Strategy

```
 1: Initialize: expert trajectory dataset $D_e$; online visited trajectory $\tau_\pi$; an empty buffer $G_{\text{sub}}$; and
    an empty buffer $T_{\text{sub}}$.
 2: Step 1: Retrieve Sub-goal Candidates
 3: Based on the current state $s_t^\pi$, retrieve a batch of candidate states $G_{\text{sub}}$ using Eq. 1.
 4: Step 2: Extract Corresponding Trajectories
 5: for Each $s_i^e \in G_{\text{sub}}$ do
 6:     Retrieve $s_i^e$'s corresponding expert trajectory $\tau_i^e$, terminate it at $s_i^e$, and store it in $T_{\text{sub}}$.
 7: Step 3: Evaluate Candidate Trajectories
 8: for Each $\tau_i^e \in T_{\text{sub}}$ do
 9:     Compute the observation trajectory similarity $W(\tau_\pi, \tau_i^e)$ using Eq. 3.
10:     if $\tau_i^e$ is among the top-M most similar trajectories then
11:         Add $\tau_i^e$ to the refined candidate set $A_{\text{sub}}$.
12:         Compute the action trajectory similarity $d_H(\tau_\pi, \tau_i^e)$.
13: Step 4: Select Final Sub-goal
14: Select the sub-goal $\tau_g, s_g$ using Eq. 4.
```

State Similarity Constraint To identify a feasible, near-future sub-goal from expert demonstrations, we begin by constructing an initial set of sub-goal candidates based on state similarity. Given the current observed state feature s_t^π, we first perform a nearest neighbor search based on cosine similarity $d_{cos}(s_t^\pi, s_i^e)$ to sample the top-k most similar states to construct a sub-goal candidate set G_{sub}:

$$G_{\text{sub}} = \text{top-k}_{(s_i^e, a_i^e) \in D_e}\big(d_{cos}(s_t^\pi, s_i^e)\big), \quad (1) \qquad d_{cos}(s_t^\pi, s_i^e) = 1 - \frac{s_t^\pi \cdot s_i^e}{\|s_t^\pi\| \cdot \|s_i^e\|} \quad (2)$$

Trajectory Similarity Constraint Due to the partial observability inherent in ego-centric visual perception, effective decision-making requires leveraging historical context. To take advantage of historical information, we prioritize among the generated sub-goal candidates G_{sub} those whose associated historical trajectories closely align with the robot's actual trajectoryτ_π.

Specifically, we evaluate similarity across both observation and action trajectories. For observation trajectories, we use the Wasserstein distance [32], which captures distributional similarity and has shown effectiveness in imitation learning tasks [33]. For action trajectories, we adopt a reversed Hamming distance [34], which counts the number of matching action positions between two sequences, favoring candidates with higher alignment to the robot's past actions.

For each candidate state-action pair (s_n^e, a_n^e) from G_{sub} (where n denotes the timestamp of the retrieved state-action pair in the expert trajectory), we retrieve its corresponding expert sub-trajectory from start to timestamp n consist as $T_{\text{sub}} = \tau_1^e, \tau_2^e, \ldots, \tau_k^e$, where each trajectory τ_i^e is defined as $\tau_i^e = \{(s_1^e, a_1^e), (s_2^e, a_2^e), \ldots, (s_n^e, a_n^e)\}$. The Wasserstein distance is then computed between the robot's current trajectory τ_π and each expert sub-trajectory τ_i^e:

$$W(\tau_\pi, \tau_i^e) = \min_{c \in C(\tau_\pi, \tau_i^e)} \sum_{k=1}^{m} \sum_{j=1}^{n} c_{kj} \cdot d(s_k^\pi, s_j^e)\ , \quad (3)$$

Where $C(\tau_\pi, \tau_i^e)$ includes all $m \times n$ transportation matrices c that fulfill the marginal conditions, with each row summing to $\frac{1}{m}$ and each column summing to $\frac{1}{n}$. Here, c_{kj} represents the amount of mass transported from s_k^π to s_j^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 τ_e, allowing us to filter out dissimilar candidates.

We then rank sub-goal candidates based on Wasserstein distance and form the refined candidate set $A_{\text{sub}} = \{\tau_{\text{e}} \in \mathrm{T}_{\text{sub}} : W(\tau_e, \tau_\pi) \leq \tau_{\text{thresh}}\}$. For each trajectory in$A_{\text{sub}}$, we compute the matching score and select the trajectory τ_g with the highest matching score:

$$\tau_g = \arg\max_{\tau_e \in A_{\text{sub}}} d_H(\tau_e, \tau_\pi) \quad (4) \qquad d_H(\tau_e, \tau_\pi) = \sum_{i=1}^{n} \mathbf{1}\left(a_i^e, a_i^\pi\right) \quad (5)$$

where a_i^e and a_i^π are the actions of trajectory τ_e and τ_π at the i-th time step. $\mathbf{1}$ is an indicator function that is 1 for identical actions and 0 otherwise.

3.3 Motion Generation

We implement a goal-conditioned retrieval policy to calculate the actions to reach the generated sub-goal. Given s_g and its associated sub-demonstration $\tau_g = \{(s_1^e, a_1^e), (s_2^e, a_2^e), \ldots, (s_n^e, a_n^e) \mid s_g\}$, ReMoBot identifies the state-action pair (s_n^e, a_n^e) from τ_g, where s_n^e is the expert state most similar to the robot's current state s_t^π, based on the sub-goal retrieval policy. ReMoBot then executes the expert action a_n^e to reach s_g. Formally, the retrieval policy chooses

$$(s_n^e, a_n^e) = \arg\min_{(s_i^e, a_i^e) \in \tau_g} d(s_i^e, s_t^\pi) \; .$$

Combined with the sub-goal generation mechanism, this framework enables the robot to complete tasks efficiently without additional training.

4 Experiments

We evaluate ReMoBot on real-world mobile manipulation tasks with complex visual observations and compare it against several state-of-the-art baselines. Our experiments are designed to answer the following key questions:

- How does ReMoBot compare to learning-based and retrieval-based baselines?
- Can ReMoBot generalize to variations in initial pose, object size, and material?
- How well does ReMoBot perform under limited data conditions?
- What are the contributions of state and trajectory constraints?

4.1 Mobile Manipulation Tasks

To demonstrate the capability of ReMoBot to deal with complex eco-centric observation, we designed three real-world mobile manipulation tasks: Table Uncover, Gap Cover, and Curtain Open. These tasks introduce perception challenges due to fabric deformability and partial observability from a front-mounted RGB-D camera, highlighting the need for decision-making under uncertainty. Task illustrations are shown in Figure 1, with further details in Section 7.3.1.

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. Beyond the deformability of the table cover, the ego-centric setting introduces significant visual challenges in this task. The pre- and post-grasp stages often appear visually similar, making it difficult to determine the correct action from single visual input alone. Moreover, once the cloth is lifted, it frequently occludes the front-mounted camera, further complicating perception and planning. As this task involves a relatively short interaction horizon, it increases the risk of overfitting in learning-based methods, particularly when trained with limited data.

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. Similar to the Table Uncover task, this scenario presents visual challenges due to the similarity

Table 1: **Baseline Performance Comparison.** Success rates (*success/total trials*) of all methods across the three tasks under dataset collection conditions. (Bolded entries indicate no statistically significant difference from ReMoBot, based on a two-tailed test for Binomial distributions with a 95% confidence interval).

	BC	**Diffuser**	**TT**	**VINN**	**DinoBot**	**ReMoBot**
Table Uncover	0/20	0/20	0/20	0/20	0/20	**14/20**
Gap Cover	3/20	0/20	0/20	0/20	0/20	**16/20**
Curtain Open	0/20	1 /20	**2/20**	0/20	0/20	**7/20**

between pre- and post-grasp stages, and frequent occlusions caused by the lifted arm and cloth. Moreover, Gap Cover involves a longer interaction horizon and an additional object, decreasing the risk of overfitting but increasing the complexity of decision-making and planning.

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 new challenges. Collision avoidance becomes critical as the robot must maneuver in a confined space, and the curtain's slippery surface can cause inconsistent motion during interaction. Furthermore, if the robot approaches the curtain from the center, the fabric can obstruct the front-mounted camera, leading to severe visual occlusion and increased uncertainty in planning. Coordinating the robot's body and arm movement is also non-trivial, occasionally resulting in inverse kinematics issues where the arm cannot reach the desired position due to constraints in the robot's configuration.

4.2 Baselines

We compare ReMoBot against several representative learning-based and retrieval-based methods. For fairness, all baselines are implemented using the same retrieval-based feature representations unless otherwise stated. Each learning-based method is trained using 20 demonstrations.

Behavior Cloning (BC): A classical supervised learning approach [35, 36], where a policy is trained to directly map observations to actions using expert demonstrations. In our setup, we train the policy to predict one-step actions.

Trajectory Transformer (TT): A transformer-based behavior cloning method that models long-horizon decision-making by generating entire state-action trajectories [23]. During evaluation, a full trajectory is predicted, and only the first action is executed at each timestep.

Diffuser (Diffuser): Diffuser leverages diffusion probabilistic models to generate trajectories that mimic expert behavior [25, 37]. While previous work has focused on large-scale datasets and point cloud inputs [38], we implement a version based solely on RGB-D observations following the Diffusion Policy framework [25].

DinoBot (DinoBot): A few-shot imitation learning framework that utilizes vision foundation models for representation encoding [22]. DinoBot retrieves demonstrations by aligning the robot's current observation with the initial observation from a demonstration through relative pose estimation, then replays the corresponding expert actions in an open-loop fashion. This serves as a retrieval-based baseline under few-shot settings.

Visual Imitation through Nearest Neighbours (VINN): VINN performs nearest neighbor search over demonstration observations to retrieve the most similar states [20], and computes an action as a Euclidean kernel-weighted average of those associated with the retrieved neighbors. We use the original VINN encoder [39] to compare against our visual perception pipeline.

4.3 Baseline Comparisons

We first compare ReMoBot with the five baselines outlined above in environments that are identical to the ones used for demonstration collection. Table 1 shows the results of 20 trials with a real Boston Dynamics Spot robot. ReMoBot outperforms all comparison methods, achieving success rates of 70% for Table Uncover, 80% for Gap Cover, and 35% for Curtain Open.

Learning-based approaches such as BC, TT, and Diffuser are unable to complete the tasks reliably when trained with only 20 expert demonstrations. Both TT and Diffuser are designed to model long-horizon behaviors, which require significantly larger datasets to effectively learn temporal dependencies. BC, on the other hand, lacks the ability to deal with previously unseen states and imbalanced training data, which we analyze in more detail further below.

Retrieval-based methods, including VINN and DinoBot, also perform poorly in our ego-centric mobile manipulation setting. VINN is constrained by the limited dataset size, which hinders the training of a robust image encoder and leads to incorrect action selection. DinoBot relies on accurate pose alignment between the robot's current observation and the demonstration frames by estimating the relative pose. This is impractical in ego-centric, partially observed scenarios in mobile manipulation.

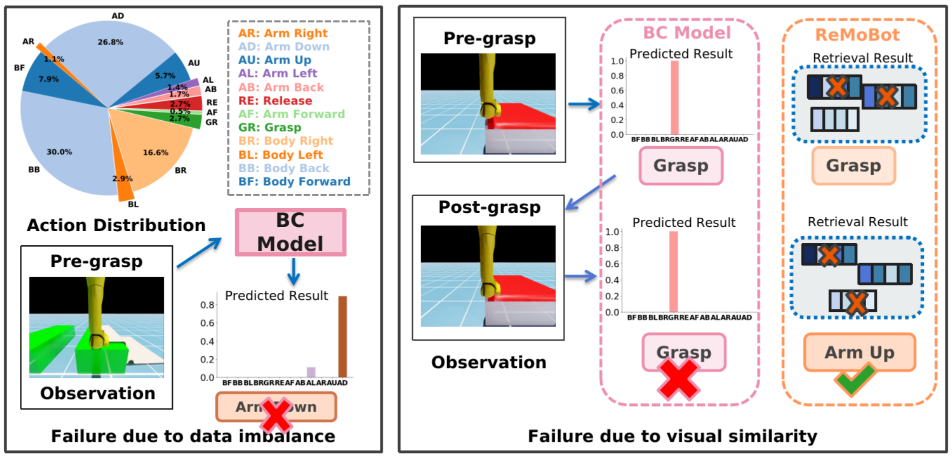

Figure 3: **Failure Analysis of Behavior Cloning (BC).** Left: Failure in the Gap Cover task caused by data imbalance. Right: Failure in the Table Uncover task due to visual similarity leading to incorrect action selection.

To analyze the limitations of the baselines and validate ReMoBot's robustness, we replicate the tasks in the Isaac Sim simulation. These simulated experiments are used solely for analysis, with no sim-to-real transfer involved. Additional implementation details and results are provided in Section 7.2. Figure 3 uses BC as an example to illustrate two main failure cases for learning-based methods. One key issue is the heavy data imbalance in the dataset, which biases the models toward frequently occurring actions and reduces their ability to learn rare but essential behaviors. The action distribution of each task can be found in Section 7.5. The second challenge arises from the visual similarity between pre- and post-grasp states. To further support this analysis, we manually modified BC's output by replacing the second grasp action with an arm-up command, resulting in 33 successful trials out of 40. In contrast, ReMoBot demonstrates robust performance in this challenging, data-constrained setting.

4.4 Generalizability Evaluation

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. Detailed configurations are provided in Section 7.3.3.

Table 2: **Generalizability Evaluation.** Success rates (*success/total trials*) of ReMoBot across variations in the object size, fabric materials, and the robot's initial position.

	Table Uncover	Gap Cover	Curtain Open
Size	10/20	10/20	6/20
Material	12/20	11/20	6/20
Position	15/20	12/20	7/20

Table 2 shows that ReMoBot maintains robust performance across diverse generalization scenarios. For each scenario, a single factor is varied while all other conditions remain consistent with the data collection environment. Minor performance drops are primarily due to incorrect sub-goal retrieval, which can occur 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, ReMoBot demonstrates generalization across varying fabric materials, object sizes, and initial robot positions.

4.5 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. As shown in Table 3, ReMoBot achieves success rates exceeding 50% with as few as 15 expert trajectories across all three simulation tasks. Overall, ReMoBot is able to learn effective manipulation strategies with relatively small amounts of data.

Table 3: **Data Efficiency Evaluation.** Success rate (*success/total*) for varying numbers of demonstration trajectories in the simulation. The numbers in the first row correspond to the number of trajectories in the dataset.

# Demo	1	5	10	15	20
Table Uncover	7/40	10/40	27/40	32/40	36/40
Gap Cover	2/40	12/40	14/40	23/40	31/40
Curtain Open	2/40	4/40	12/40	20/40	32/40

4.6 Ablation Study

We conduct an ablation study to evaluate the impact of two different similarity constraints used in sub-goal retrieval. The evaluation is performed in a simulated environment identical to the environment in data collection, ensuring a controlled comparison. Table 4 shows that incorporating trajectory similarity consistently improves performance across all tasks, highlighting its importance.

Table 4: **Ablation Study.** Success rate (*success/total*) for different similarity constraints.

Task	State	State + Trajectory
Table Uncover	30/40	36/40
Gap Cover	17/40	31/40
Curtain Open	26/40	32/40

5 Conclusion

Learning mobile manipulation skills for complex tasks, such as deformable 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 sub-goal generator, enabling 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 outperforms both learning-based and retrieval-based baselines, effectively acquiring manipulation skills from a limited dataset. Furthermore, ReMoBot demonstrates generalization to varying environmental conditions. In future work, we aim to extend ReMoBot by explicitly incorporating collision handling and integrating online fine-tuning mechanisms to improve adaptability during deployment.

6 Limitations

Despite the promising results of ReMoBot, several limitations remain that point to directions for future work.

First, for example, in the Curtain Open task, the robot occasionally collides with the curtain hanger. This is primarily due to the absence of obstacle information, which causes the robot to overlook it. Furthermore, the segmentation tracker may drift or switch attention to other objects, leading to incorrect tracking and subsequent failure cases.

Second, the sub-retrieval module sometimes selects visually distinct states as sub-goals. This limitation arises from the restricted representation power of the foundation model and the limited diversity of the offline dataset. Such issues are common in imitation learning scenarios that lack online adaptation or fine-tuning, making the model less robust to novel or ambiguous situations. To address this, we consider adding an online adaptation process or incorporating a failure detection and recovery mechanism in the future to enhance the system's robustness.

In the Table Uncover and Gap Cover tasks, we observe that pre- and post-grasp observations are often nearly identical, which can result in the agent converging to local optima during evaluation. This stems from the nature of the task design. One potential solution is to introduce a grasp flag to discourage redundant grasp attempts and improve action diversity.

Another limitation related to our control scheme, that the robot's body and arm are operated independently. At each timestep, the policy outputs a command for either the body or the arm. Arm movements rely on inverse kinematics (IK) to translate target poses into joint configurations. However, this decoupled control approach can result in IK failures during evaluation, which directly contributes to task failures. Future work may explore more integrated control strategies or redefine the action space in terms of joint configurations to improve reliability.

Regarding motion generation, we currently adopt a simple retrieval-based strategy that is efficient in the discrete action space. In future work, we plan to explore more expressive motion generation techniques, including models trained with supervision or constraints tailored to action semantics. Extending the system to continuous action spaces is also a promising direction by retrieving skill-level actions.

We believe that many of these limitations can be mitigated through the collection of more diverse data, the inclusion of additional contextual signals (e.g., grasp flags), and the incorporation of online learning or fine-tuning mechanisms to adapt the model to new situations.

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] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[29] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.

[30] Z. Yang and Y. Yang. Decoupling features in hierarchical propagation for video object segmentation. *Advances in Neural Information Processing Systems*, 35:36324–36336, 2022.

[31] 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.

[32] N. Bonneel, M. Van De Panne, S. Paris, and W. Heidrich. Displacement interpolation using lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference*, pages 1–12, 2011.

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

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

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

[36] 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.

[37] 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`.

[38] 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.

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

7 Appendix

7.1 Perception Pipeline

In this section, we describe the details of the perception module. Notably, the same perception module is used in both the real-world and simulation settings.

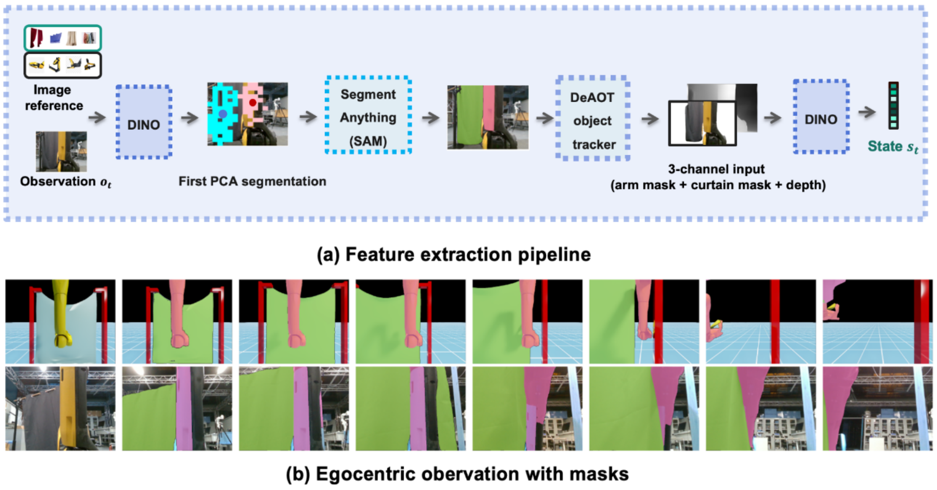
Image reference
Observation o_t
DINO
First PCA segmentation
Segment Anything (SAM)
DeAOT object tracker
3-channel input (arm mask + curtain mask + depth)
DINO
State s_t
(a) Feature extraction pipeline
(b) Egocentric obervation with masks

Figure 4: **Perception Pipeline.** (a) Retrieval Feature Generation Pipeline: We use DINOv2, along with object reference images, to generate prompts for the SAM model to segment the task-relevant objects. These segments are then tracked throughout the task using the DeAOT tracker. Finally, DINO encodes the task-relevant segmentation into a compact representation. (b) Observation with Mask: Visualization of robot observation sequences both in simulation (top row) and the real world. (bottom row).

We use the input from the front-mount camera as the observersion, therefore $O_t = \{I_{\text{rgb}}, I_{\text{depth}}\}$. As shown in Fig. 4, we first generate retrieval features using a pre-trained DINO model as a dense visual descriptor [31] to perform co-segmentation. This step produces prompt points for task-related objects, e.g., the robot embodiment and the fabric. Using these prompts, we apply the Segment Anything Model (SAM) [28, 29] to segment task-relevant objects from the visual input. To ensure temporal consistency, we then use the AOT tracker [30], which maintains stable segmentation across frames by continuously tracking the arm and curtain using the refined SAM masks. The result of segmentation is presented in the bottom row in Fig. 4.

Following segmentation, we construct a three-channel input feature composed of the robot maskI_{robot}, manipulated objects maskI_{object}, and depth image, denoted as $I = \{I_{\text{robot}}, I_{\text{object}}, I_{\text{depth}}\}$. This input is passed through the frozen pre-trained DINO model to extract a neural feature representation of the scene, yielding a 384-dimensional state vector s_t^{π}.

7.2 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. 5, 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.

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

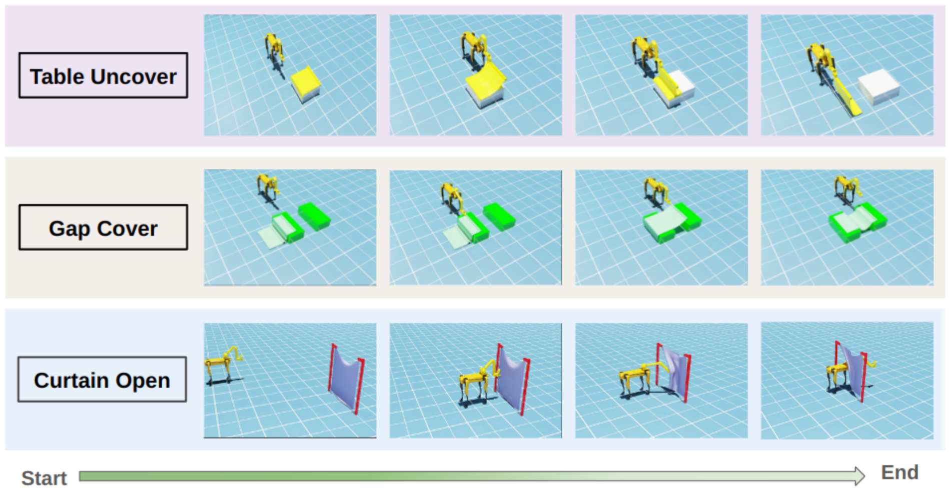

Figure 5: **Three Deformable Mobile Manipulation tasks in Simulation.** Table Uncover (top), Gap Cover (middle), and Curtain Open (bottom) tasks. To solve these three tasks, a Spot robot with a front-mounted RGB-D camera must coordinate body movement and arm operation to navigate and manipulate fabric. Note that we collect demonstrations and evaluate all the methods directly in the simulation.

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 to reduce 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.

7.3 Experiment

7.3.1 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 $O_t = \{I_{\text{rgb}}, I_{\text{depth}}\}$. 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 7.3.2. 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 × 110 cm blue plastic cloth and set the gap size as 50 cm.
- **Curtain Open:** The curtain is a 130 cm × 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 × 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 × 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 × 110 cm, with a distance of 130 cm between the two hangers.

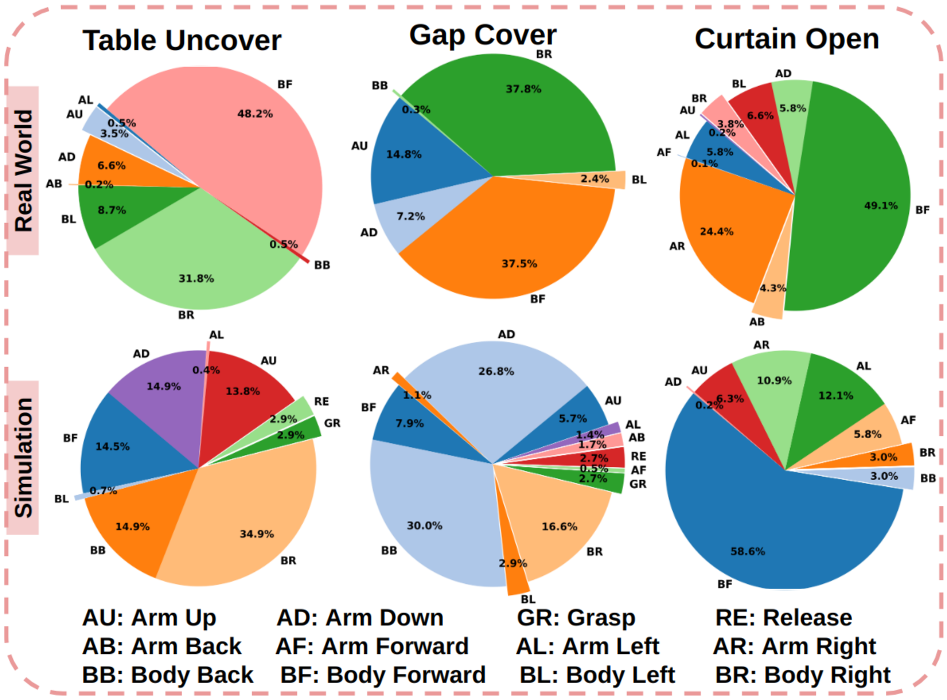

Figure 6: **Action Distribution in Dataset for Each Task:** Each pie chart represents the frequency of discrete actions in the expert dataset for three tasks in both simulation and real-world settings.

7.3.2 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.

7.3.3 Generalizability Analysis

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.

Table 5: **Baseline comparisons.** The table shows the number of successful attempts out of the total evaluation runs (success/total rounds) for all methods in the three tasks under the dataset collection environmental conditions.

	BC	Diffuser	TT	VINN	DinoBot	ReMoBot
Table Uncover	0/40	0/40	0/40	0 /40	0/40	**36/40**
Gap Cover	15/40	0/40	0/40	0/40	0/40	**31/40**
Curtain Open	16/40	0/40	0/40	0/40	0/40	**32/40**

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.

7.4 Experimental Simulation Results

7.4.1 Baseline Comparison

The deformation behavior of the fabric observed in the simulation closely aligns with results from the real world. However, learning-based methods such as TT and Diffuser still fail to solve the deformable manipulation task in the simulated environment, further highlighting the challenges this setting poses for data-driven approaches. Similarly, the consistent failures of retrieval-based methods like DinoBot and VINN further support the conclusion that the primary cause of failure is not attributable to noise in the real-world evaluation environment.

However, once noise from the real-world setting is removed, the success rate of behavior cloning (BC) increases significantly. We attribute this improvement primarily to a reduction in distribution shift. In simulation, the cloth behaves similarly to how it does in the expert demonstrations, which minimizes this shift. For the Table Uncover task, the primary failure mode stems from the visual similarity between different stages of the task. After manually addressing this bottleneck, we observed a notable increase in the success rate.

7.4.2 Generalization Evaluation

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

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. 'Origin' 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
Origin	32/40	35/40	32/40

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.

7.5 Dataset Analysis

Figure 6 shows the action distribution for each task in our expert dataset in the real-world setting. Each task involves a distinct distribution of discrete robot actions, reflecting the action imbalance. For instance, Table Uncover and Curtain Open are dominated by the BF (body-front) and BR (body-right) actions, whereas Gap Cover demonstrates a more balanced distribution between BR and BF, along with significant occurrences of AU (arm-up) and AD (arm-down). These distributions highlight the diversity and task-specific nature of action trajectories in our collected demonstrations.

In the real-world setting, the task is defined such that the fabric is considered grasped as long as the end-effector touches it. This makes the AU (arm-up) and AD (arm-down) actions particularly important for successful execution. However, these two actions are underrepresented in the dataset for both the Table Uncover and Gap Cover tasks, which likely contributes to the failure of the baseline methods. Meanwhile, grasp and release actions are treated as separate and appear in less than 3% of demonstrations for both tasks, indicating the alignment between the real-world and simulated settings.