Data-Efficient Learning for Mobile Manipulation using Vision-Based Topological Planning

Zipeng Fu Chelsea Finn
Stanford University

***Abstract*— Mobile manipulation systems require tremendous engineering efforts to ensure that different modules (e.g. manipulation, navigation, perception) work in coordination. End-to-end imitation learning can alleviate this problem by learning a policy from expert demonstrations. However, the diverse perception inputs and large action spaces of mobile manipulation tasks impose burdens on the size of collected training data for the imitation learning to perform well. Underperforming policies will inevitably drift away from the data distribution due to compounding errors. In this work, we propose a data-efficient method for learning from mobile manipulation demonstrations by utilizing topological planning. By building a topological graph on both head and wrist camera views, planning on the graph, and acting using a local goal-conditioned policy, our method prevents the robot from drifting away from the training distribution during deployment, and thus alleviates the need for large amounts of training data. We evaluate our method both in simulation and on a real LoCoBot. We find that with only 20 demonstrations, the robot can solve room tidying tasks such as opening a drawer, picking up a sock inside, and then relocating it with around 76% success rate. Project website at https://mobile-manipulation.github.io/topological**

I. Introduction

Building robust mobile manipulation systems is a long-standing challenge in robotics, as they require each component of the robot systems, including manipulation, navigation, locomotion, and perception, to function correctly not just in individual manners, but in tight coordination with each other. For example, consider the room tidying tasks shown in Figure 1, such as retrieving a sock from a drawer. Proper positioning of the mobile base is crucial; if the robot is too close, the arm can't open the drawer; if too far, it can't reach. The manipulation of objects also influences the robot's mobility. For instance, when cleaning a spill on a high surface, the arm must avoid heavy forces and collisions to prevent destabilization of the robot. We aim to allow a robot to complete mobile manipulation tasks that require such tight coordination.

Many existing mobile manipulation systems use model-based control by carefully injecting human knowledge and experience into system modeling and engineering [1]–[4]. One of the hallmarks of model-based control for mobile manipulation is the DARPA Robotics Challenge [5]. However, such systems can be difficult to be model, requiring large teams to design and maintain, and small perception modeling errors can lead to catastrophic control failures [6], [7]. In contrast, end-to-end imitation learning circumvents the need for such models by directly learning a policy that maps from raw sensory inputs to control outputs. Imitation learning has shown some success in many settings like table-top manipulation [8]–[14]

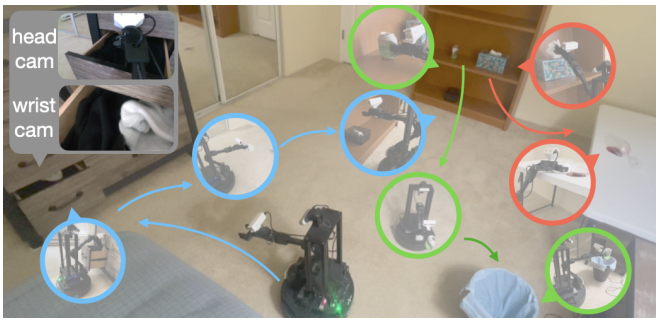

Fig. 1: Our method can solve mobile manipulation tasks with limited expert demonstrations. By using topological planning conditioning on both wrist and head RGB cameras, our method can effectively learn to solve some bedroom tidying tasks.

and bimanual manipulation [15]–[17]. Despite some success in mobile manipulation settings [13], [18], [19], the high-dimensional nature of mobile manipulation problems pose challenges for end-to-end imitation learning. As a result, such systems have required large amounts of data. For example, the RT-1 project [18] used a fleet of 13 robots over a span of 17 months to collect 130k episodes of data. Can we avoid both heavy task-specific model engineering and collecting a huge training set by developing a more data-efficient method for learning from mobile manipulation demonstrations?

In this work, we present a data-efficient method for learning from demonstrations for end-to-end mobile manipulation by incorporating topological planning. Our method has three main parts: a planning algorithm, a traversability function for planning, and a local goal-conditioned policy. We observe that planning plus a local goal-conditioned policy is more data-efficient than behavior cloning. The rationales are two-fold. First, by using planning to reach a next immediate goal state within the training set, the method mitigates the issue of compounding errors that imitation learning methods commonly face, where the robot can drift away from the training distribution due to small mistakes [20], [21]. Second, the local goal-conditioned policy at a given state is easier to model than an imitation policy at the same state, because conditioning on the immediate next goal state to reach reduces the multi-modality of the action distribution that arises from the stochasticity of human demonstrations [22]–[25].

To implement our method, we first train a traversability function to predict the number of time steps between two states, where states are images captured by the robot's head and wrist cameras. The traversability function is trained using temporal information from expert demonstrations, such that states that are close in time within one trajectory are predicted

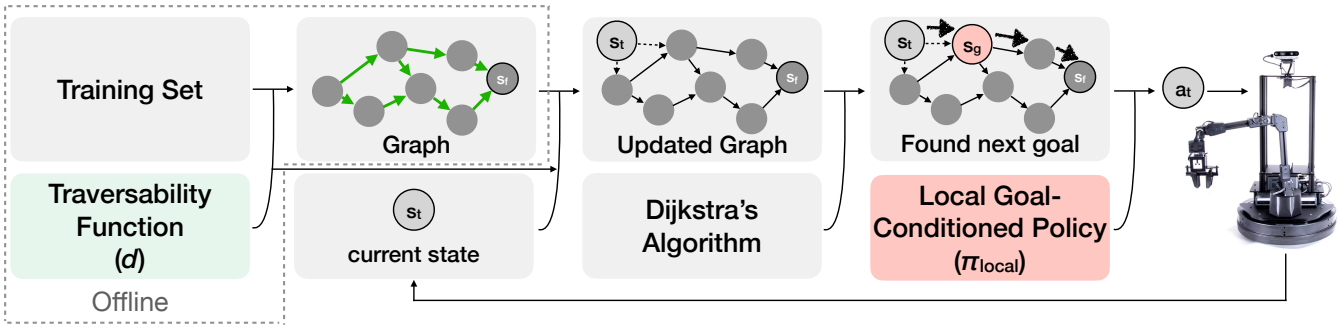

Fig. 2: Our pipeline. We first train a traversability function and a local goal-conditioned policy using the training set of expert demos. Then we construct a graph using the training set and the learned traversability function. During deployment, we use Dijkstra's algorithm to plan the next goal to reach, which is used by the local policy to act; the next state is added to the graph, and the process repeats.

with the difference between corresponding time steps, and states that are far apart or not from the same trajectories are predicted as far apart. This traversability function allows us to build a topological graph over the training data, where each state is a node, edges are weighted by the predicted traversability, and nodes that are predicted to be far apart are not connected. Given this graph, we can use any shortest path finding algorithm (e.g. Dijkstra's) for planning given the current state and a goal state. The planner provides the next state to reach. To reach the next state, a local goal-conditioned policy can be trained using only state pairs within a trajectory that are close in time steps. During deployment, the planning algorithm gives the best next state to reach at each time step, and the local goal-conditioned policy is used to reach that state.

The main contribution of this paper is a data-efficient method for learning from mobile manipulation demonstrations using topological planning, demonstrating its effectiveness and data efficiency both in real-world room tidying tasks and in simulation tasks. We introduce two simple, yet key components: a latent-flow network structure and using both wrist and head cameras in our method, and show that they are crucial for mobile manipulation performance under the data-limited regime. Our experiments show that with only 20 demonstrations, our method can perform real-world bedroom tidying tasks like opening a drawer, picking up a sock inside, and then relocating it with around 76% success rate, better than baseline methods. In addition, topological planning enables new capabilities like experience chaining, where the robot can solve two tasks sequentially by only learning from demonstrations of singular tasks.

II. Related Work

a) Learning for Mobile Manipulation.

Mobile manipulation has recently been tackled by both reinforcement learning and imitation learning. By either decomposing the action space [26]–[29], or instead imposing whole-body control objectives during training [30], [31], reinforcement learning can be effective for learning mobile manipulation tasks. Unlike these prior approaches, our method does not rely on state estimators or object bounding boxes; by learning directly from RGB pixels, our method can leverage all relevant information from the camera inputs. Prior works have also learned mobile manipulation tasks by incorporating a fixed-pose grasping primitive [32] or pre-trained primitives and skills (e.g. picking, tossing and in-context learning by large language models) [33] to avoid prohibitively expensive exploration. In contrast, we present a method that does not restrict the action space of the robot. Unlike reinforcement learning, end-to-end imitation learning allows a mobile manipulator to directly learn reactive vision-based policies from expert data [13], [18], [34]–[36]. RT-1 [18], BC-Z [13], and Palm-E [36] show the potential of leveraging language instructions to control mobile manipulators to complete tasks reactively. However, huge amounts of training data have been needed to train a policy with diverse visual inputs and a large action space. For example, RT-1 [18] trains a policy on 130k episodes of data collected with a fleet of 13 robots over 17 months. Our work tries to tackle this data hunger problem in mobile manipulation by using topological planning for learning from demonstrations.

b) Topological Planning for Robotics.

Early works used topological planning in model-based control for navigation [37]–[40]. These methods rely on sonar measurements and do not directly use visual inputs. More recently, topological planning with learned parametric models has been widely applied in robot navigation using end-to-end learned visual representations [41]–[50]. Focusing on mobile manipulation instead of pure navigation, we take inspirations from these methods and add two key components for data-efficient learning from mobile manipulation demonstrations: using both wrist and head cameras, and a latent-flow network structure.

III. Learning from Mobile Manipulation Demos by Topological Planning

Our goal is to design a method for learning mobile manipulation tasks from demonstrations. While direct imitation learning provides a viable solution when a large number of demonstrations are available [13], [18], [36], existing methods struggle with learning from fewer demonstrations due to compounding errors [20], [21]. We focus on this data-limited regime, and present a method that leverages topological planning to stay close to the data distribution and recover from mistakes due to compounding errors. In this section, we first formalize the problem setting and overview the method before presenting each method component in detail. We also include an detailed algorithm section in the project website for both the training pipeline and the deployment pipeline.

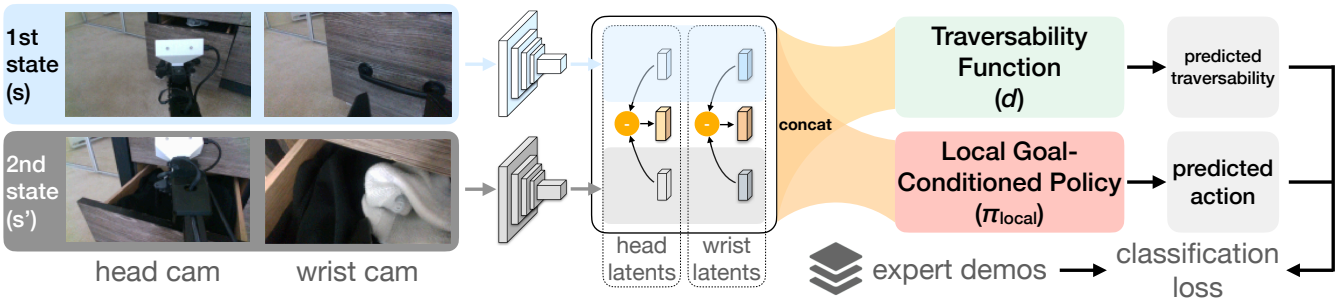

Fig. 3: Latent-flow network structure for traversability function and local goal-conditioned policy given wrist and head camera RGB images. We train the networks end-to-end using classification loss with expert demonstrations.

A. Problem Formulation

We consider the problem of learning mobile manipulation tasks from demonstrations. Given a training set of expert demonstrations $D = \{\tau_i\}_{i=1,...}$, where each trajectory consists of state-action pairs $\tau_i = \{s_1^i, a_1^i, s_2^i, a_2^i, ...\}$, the robot aims to solve the task demonstrated in the expert behavior. Each state contains both a head camera image I_{head} and a wrist camera image I_{wrist}. Actions are either discrete motion primitives or discretized continuous controls of the arm end-effector position and orientation, the gripper state, and the base linear velocity and angular velocity.

B. Method Overview

In contrast to behavior cloning, where a policy is supervised trained by using the expert demonstrations, our method has three components: a traversability function d, a local goal-conditioned policy π_{local} and a planning algorithm. Shown in Figure 2, we first construct a graph offline using the training set and a learned traversability function d. The traversability between each node represents the number of time steps that the robot takes to reach one state from another. Since the traversability from one state to another is not necessarily symmetric, the graph is directional. During deployment, the current state s_t is added to the graph at each time step after querying the traversability function d to get the predicted traversability between this state to all states in the training set $\{d(s_t, s_{t'}) \mid \forall s_{t'} \in D\}$. The planning algorithm (Dijkstra's algorithm) then provides the best immediate goal state s_g so that the local goal-conditioned policy π_{local} can then generate the action a_t.

The general pipeline of our method is similar to prior work on topological planning for navigation such as ViNG [43] and SPTM [42]. However, we introduce two key components to topological planning for mobile manipulation: latent-flow network structure, and using both wrist camera and head camera. Empirical ablation experiments and comparison studies with baseline methods show that these added components to topological planning enable better performance on mobile manipulation tasks both in simulation and in real-world room tidying tasks.

C. Traversability Function

To train a traversability function $d(\cdot, \cdot)$ that measures the weight of each directional edge connecting states in the graph, we frame the problem as predicting the number of time steps between states. Because time steps are discrete, we formulate the training of the traversability function as a classification problem, where the traversability function outputs a probability distribution over a finite set of $d_{\max}$ discrete classes, representing the possible traversability from 1 to $d_{\max}$. If two states are more than or equal to $d_{\max}$ steps apart or from different trajectories, they are considered disconnected and assigned the same traversability class. If two states are identical, they are considered to be 1 step apart instead of zero, so no zero weight edge exists in the graph, and any off-the-shelf shortest-distance algorithms can be used for planning. Formally, given two states s_t^i, the state at t of i-th trajectory, and $s_{t'}^j$, the state at t' of j-th trajectory, the classification loss is

$$L_{\text{traversability}}\left(s_t^i, s_{t'}^j\right) = \begin{cases} \text{CE}\left(\max(t'-t, 1), d(s_t^i, s_{t'}^j)\right), \\ \qquad \text{if } i = j \text{ and } 0 <= t'-t < d_{\max} \text{ (Case 1)} \\ \beta * \text{CE}\left(d_{\max}, d(s_t^i, s_{t'}^j)\right), \text{otherwise (Case 2)} \end{cases}$$

where $\text{CE}(\cdot, \cdot)$ is the cross-entropy loss, θ are the parameters of the traversability function, β is a loss scaling factor the Case 2. We set β to 5 to ensure that the traversability function can distinguish states that are far apart in time but visually similar, which is common in mobile manipulation tasks that require more granular actions than navigation. We use a sampling strategy Δ to get mini-batches from the demonstration dataset D to train the traversability function. For 50% of the mini-batch, we use a balanced sampling strategy to sample states in Case 1 that are from the same trajectory and close in time, ensuring that each traversability class is equally represented. For the other 50%, we sample states in Case 2 that are from different trajectories or far apart in time uniformly at random. The objective is then given by $J_{\text{traversability}} = \mathbb{E}_{(s_t^i, s_{t'}^j)\sim\Delta}[L_{\text{traversability}}(s_t^i, s_{t'}^j)]$.

D. Planning via Dijkstra's Algorithm

Given the learned traversability function, a graph can be constructed offline using the training set. During deployment, we iteratively add the current state s_t to the graph by querying the traversability function $d(\cdot, \cdot)$ between s_t and all other states in the training set D. For each node in the graph, we update the edge weights from s_t to this node using the traversability function $d(s_t, s_{t'})$ and vice versa. Then, given a final state s_f, which is the last state of a expert trajectory, we use Dijkstra's shortest path algorithm on this updated graph to select the immediate next goal state s_g on the shortest path from the current state s_t to the final state s_f.

Fig. 4: Real-world experiment tasks: 'sock', 'trash', and 'wine', each of which involve a sequences of manipulation and navigation actions.

E. Local Goal-Conditioned Policy

After planning and an immediate goal state s_g to reach is found, we use a local goal-conditioned policy to generate the current action $a_t = \pi_{\text{local}}(s_t, s_g)$. This goal-conditioned policy is local, as the planning algorithm will only provide a goal state that is close to the current state. Hence, π_{local} only learns for states that are close (i.e. Case 1). Similar to the RT-1 policy training objective [18], we discretize each action dimension into bins for continuous controls, and use the cross-entropy loss for supervised training. Given two states s_t^i, the state at t of i-th trajectory, and $s_{t'}^j$, the state at t' of j-th trajectory, the classification loss is

$$L_{\text{policy}}\left(s_t^i, s_{t'}^j\right) = \begin{cases} \text{CE}\left(a_t^i, \pi_{\text{local}}(s_t^i, s_{t'}^i)\right), \\ \qquad \text{if } i = j \text{ and } 1 <= t' - t < d_{\max} \text{ (Case 1)} \\ 0, \qquad \text{otherwise (Case 2)} \end{cases}$$

In contrast to a global policy learned in behavior cloning, the local goal-conditioned policy at a given state is easier to model because conditioning on the immediate next goal state to reach reduces the multi-modality of the action distribution that arises from the stochasticity of human demonstrations. We later show empirical evidences in Section IV. The local goal-conditioned policy is trained by using the same sampling strategy Δ as the traversability function, so the objective is $J_{\text{policy}} = \mathbb{E}_{(s_t^i, s_{t'}^j)\sim\Delta}[L_{\text{policy}}(s_t^i, s_{t'}^j)]$.

F. Latent-Flow Network Structure

Prior work uses frame stacking in the channel dimension [43], [51], [52] or a Siamese architecture [42] to model a image-based traversability function or a goal-conditioned policy that takes a pair of images as inputs. Inspired by the use of latent flow in reinforcement learning [53], we introduce a latent-flow network structure to model both the traversability function and the local goal-conditioned policy by explicitly incorporating the change of states in a latent space into the network structure (Figure 3). Concretely, consider the first state $s = [I_{\text{head}}, I_{\text{wrist}}]$ and the second state $s' = [I'_{\text{head}}, I'_{\text{wrist}}]$. Each image is separately encoded by a small ResNet [54] resulting in four latent vectors $z_{\text{head}}, z_{\text{wrist}}, z'_{\text{head}}, z'_{\text{wrist}}$. We can get the change of states in the latent space for the head camera views by $z'_{\text{head}} - z_{\text{head}}$ and for the wrist camera views by $z'_{\text{wrist}} - z_{\text{wrist}}$. Then, all of these six latents are concatenated and fed into two multi-layer perceptions (MLPs), one for the traversability function d and the other for the local goal-conditioned policy π_{local}. The networks are trained end-to-end by using a joint objective function $J_{\text{traversability}} + \alpha * J_{\text{policy}}$. We set α to 5 to balance the scale of the two objectives. Due to the small amount of training data, we impose strong network regularization, including Stochastic Depth [55] in the ResNets with a 0.3 probability and Dropout [56] in the MLPs with a 0.3 rate.

IV. Experimental Results

We evaluate our method in three experiments: one experiment in the real world (a LoCoBot doing room tidying tasks) and two experiments in simulation (a object rearrangement task using a LoCoBot, and a table-top grasping task using a static robot arm). All of the experiments are only given small amounts of training data. The control space of the robots range from continuous control using a discretized action space to discrete motion primitives. We aim to address the following research questions:

- Does our topological planning method enable data-efficient learning from mobile manipulation data?
- How does our method compare with other prior imitation learning methods?
- How do various design choices in our method affect performance?

A. Baselines

We evaluate our method against four baselines. All of them are trained with both the wrist camera and the head camera, same as ours. In addition, a goal image that shows the end of a task is supplied to the baselines.

- **Behavior Cloning (BC)** is a simple but commonplace baseline for imitation learning from expert demonstrations. We follow [57] and implement BC using two separate ResNets producing two latent vectors, one for the wrist camera image and one for the head camera image, which are concatenated and processed by a MLP to produce an action.
- **BC-RNN** [58] complements BC with a sequential encoding of past visual information by using a GRU [59], and the latent embedding of the GRU is fed to a MLP to produce an action.
- **RT-1** [18] is a transformer-based imitation learning method that predicts the current action given a fixed-length history of image observations, where each image is separately processed by a CNN and produce discrete tokens. The discrete tokens are concatenated and fed into a transformer to produce an action. We remove the network stream for language encoding.
- **MVP** [60], [61] provides visual representations that are pre-trained on various real-world image datasets. We fix and use these visual representations for training a downstream MLP that outputs actions via behavior cloning.

B. Real-World Experiments: Room Tidying Tasks

Robot Setup. The mobile manipulation platform that we use is a LoCoBot, which is equipped with two RealSense

	10 demos			20 demos			30 demos		
	sock	trash	wine	sock	trash	wine	sock	trash	wine
Ours	30	**60**	**80**	**60**	**80**	**90**	**70**	**90**	**100**
BC	10	40	50	30	50	60	50	70	90
BC-RNN	0	20	10	0	40	50	30	60	**100**
RT-1	0	0	0	0	20	30	10	50	30
MVP	**40**	**60**	70	50	60	**90**	50	60	90

TABLE I: Success rate of real-world room tidying tasks. Our method perform consistently better than the baselines in data-limited regimes. Each measurement is averaged across 10 trials.

D435 cameras, one on the wrist and one on the head, to capture RGB images. Both RGB images are downscaled to a dimension of 90 x 160 before being fed into the networks. We collect expert demonstrations through the PyRobot software interface [62] mapping controls to a keyboard. Due to lack of continuous teleoperation interface, the action space is discrete, consisting of 15 motion primitives, including "end_effector_left", "end_effector_right", "end_effector_forward", "end_effector_backward", "end_effector_up", "end_effector_down", "base_forward", "base_backward", "base_turn_left", "base_turn_right", "wrist_up", "wrist_down", "gripper_close", "gripper_open", and "terminate". Videos can be found on the project website.

Tasks and Expert Demonstrations. We create three bedroom tidying tasks for the robot. Shown in Figure 4 and Figure 1, the first task ("sock") is retrieving a sock from a drawer. It requires the robot to locate the drawer storing the sock, navigate to a nearby location of the drawer that is suitable for subsequent arm manipulation, open the drawer using the whole arm motions, locate the sock and pick up using the gripper, navigate to a container box on the other side of the bedroom, and release the gripper to put the sock into that container box. The second task ("trash") is trash collection, requiring the robot to raise the gripper high above its head, fetch a empty soda can, navigate to a waste bin on the other side of the bedroom, and then put the soda can inside the waste bin. The last task ("wine") is cleaning a wine spill on a high surface. It requires the robot to fetch a tissue, navigate to the high surface, reach the arm towards the wine spill, and then release the tissue to cover the spill. We collect 30 expert demonstrations per task for a total of 90 trajectories. Each trajectory lasts around one minute and has around 40 to 100 timesteps. We set the maximum traversability class $d_{\max}$ to be 5, based on the length of the expert trajectories. We stack last two frames of history to form the current state.

Comparisons with Baselines. We compare our method with the four imitation learning baselines introduced in Section IV-A across the three indoor mobile manipulation tasks that require both manipulation and navigation, and across three different training set sizes of 10, 20 and 30 expert demonstrations. We early stop model training for every baseline once the validation loss starts to rise. In Table I, we report the average success rate across 10 trials. Our method achieves the highest success rate across all tasks and all training sizes. While MVP, BC and BC-RNN can achieve good performance when training data increases to 30 expert demonstrations, our method has non-trivial success rates when only 10 or 20 demonstrations are provided during training. We notice that MVP's performance does not improve with more demos from 20 to 30 given its fixed representations. We also notice that the complexity of retrieving a sock from a half-opened drawer is higher than other two tasks due to the control complexity of opening the drawer using a low-cost and small robot arm, and the perception complexity of locating sock inside the drawer, which is occluded from the head camera.

Ours (10 demos)	Ours (30 demos)	BC	BC-RNN	RT-1	MVP
0.023s	0.032s	0.008s	0.010s	0.13s	0.23s

TABLE II: Execution time per step running on an Intel NUC.

In addition, shown in Table II, we compare the execution time of our algorithm to the baselines, showing that our method has longer execution time than BC and BC-RNN, but much better execution time than RT-1 and MVP that have large networks.

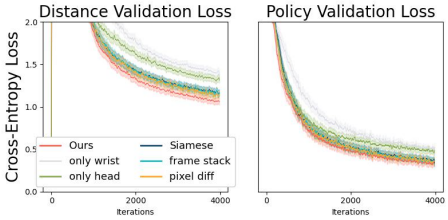

Fig. 5: Validation loss of traversability function and policy training. Our method achieve lower validation loss in both than ablations.

Ablation Studies. Our method applies the latent-flow network structure and both wrist and head cameras to topological planning, aiming to achieve better modeling of the traversability function and the local goal-conditioned policy. We ablate each of the three components to study the usefulness of them. For a fair comparison, we ensure that the network size of the ablations are roughly the same size as ours. In Figure 6, we report the validation training loss of our method against the five ablations ("Siamese [42]", "frame stacking", "pixel-level difference [41]", "only head", "only wrist") across three random seeds and three real-world tasks trained on 20 expert demonstrations. With the help of the latent-flow network structure, wrist camera and head camera, our method achieve lower validation loss for both the traversability function and the local goal-conditioned policy. In Table III, we further test the performance of ablations in the real world. Without latent flow, the robot has lower success in all tasks, especially in the 'sock' task where precise actions for opening the drawer and reaching to grasp the sock are crucial for the final success. With only head camera, the robot cannot complete the 'trash' and 'wine' task, both of which require the wrist camera to observe the objects to be grasped that are not observable to the head camera. With

	sock	trash	wine
Siamese	30	60	80
pixel-level diff	40	50	60
frame stacking	40	50	50
only head cam	10	0	0
only wrist cam	0	60	80
Ours	60	80	90

TABLE III: Success rates of real-world tasks of our method and ablations given 20 demonstrations, averaged over 10 trials.

only wrist camera, the robot cannot complete the 'sock' task which has a relatively long path of navigation to a container box, requiring the head camera.

Experience Chaining. Another benefit of using teleological planning other than mitigating distribution shift is experience chaining. When collecting the expert demonstrations for the 'sock' task and the 'trash' task, we observe that the starting states of the 'sock' task and the end states of the 'trash' task are close to each other. As a result, when we train our method using both the trajectories from the 'sock' task and the end states of the 'trash' task, even though not in the same trajectories, the starting states of the 'sock' task and the end states of the 'trash' task can be classified as being close in traversability. Hence, before the robot starts to solve the 'sock' task, we can set the final state s_f to be one of the end states of the 'trash' demonstrations where the robot releases the trash into the trash can, and expect the robot to first complete the 'sock' task and then the 'trash' task sequentially. Effectively, we can chain two separate experiences into one long-horizon sequential execution of two tasks by using topological planning. Due to the limited data, we can achieve a 20% success rate of chaining the 'sock' task and the 'trash' task, evaluated on 10 trials. One of the frequent mistakes that the robot makes is learning to "shortcut" by directly approaching the trash can right after completing fetching the sock from the drawer without approaching the container box or the trash. In contrast, BC does not complete the task and its trajectories diverge during navigation and trying to fetch the trash.

Higher Action Certainty. During the collection of expert demonstrations, the human operator can generate multi-modal sequences of commands to the robot as long as the tasks are solved. As a result, when behavior cloning such multi-modal expert demonstrations, the resulting learned policy will have a higher entropy on the output action distribution. In contrast, our method supplies an immediate next goal to the local goal-conditioned policy, so it can be more deterministic about the action to take to reach the goal. We show one example in Figure 6. The expert demonstrations for the 'sock' task contain two ways for the robot to navigate to the drawer, namely following the grey dotted line to move towards the cabinet first and then turning leftwards to face the drawer, or first turning left, and then approaching the drawer directly. As the a result, the BC policy has a high entropy at the start of the 'sock' task, the action probability for selecting `base_forward` and `base_turn_left` are 0.68 and 0.31 respectively when averaged around 10 trials. In contrast, our method prefers selecting goal states that can lead the robot to the final state faster, so the immediate goal state supplied by our method at the starting state is always on the grey dotted line. Hence, our method has a constant 0.99 probability of selecting `base_forward` at the start of the 'sock' task, which is much more deterministic than behavior cloning.

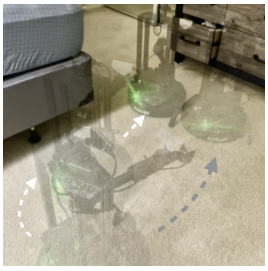

Fig. 6: Stochasticity in expert demonstrations at the start of the 'sock' task.

C. Mobile Manipulation Simulation: Object Rearrangement

We additionally create a simulated object rearrangement task in Pybullet [63]. Shown in Figure 7, the task requires a LoCoBot to grasp a cube from one table, navigate to another table, and then put the cube onto of another table. The action space is 6-dimensional, containing a binary command of base moving forward with a unit speed or still, a three-class base angular velocity command (rotate left with a unit speed, still, rotate right with a unit speed), a 3-dim end-effector command where each dimension is discretized into 100 bins, and a gripper command. The control frequency is 10Hz. The end-effector command is convert into target joint positions by using the default IK of Pybullet, and the base commands are converted to target wheel velocities using the differential drive model. Both the arm and the base are converted using PD controllers.

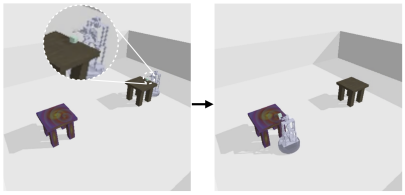

Fig. 7: Object rearrangement in simulation.

We generate 30 expert demonstrations using a scripted policy with randomly sampled textures of the tables and color of the cube. As a result, the demonstrations are much less stochastic than human demonstrations. Each demonstration trajectory is around 250 timesteps. In Table IV, we show the success rates of our method and baselines averaged across 3 random seeds and 100 trials. Due to the determinism of the demonstrations, all methods can complete the tasks with decent success rates given 10, 20 or 30 demonstrations for training, but our method still performs better in data-limited regime and is competitive given all 30 demonstrations.

	BC	BC-RNN	RT-1	Ours
10 demos	52	40	63	**72**
20 demos	80	74	79	**82**
30 demos	91	**92**	**92**	89

TABLE IV: Success rates of object rearrangement in simulation.

V. Conclusion and Limitations

In this paper, we present a method for learning from demonstrations for mobile manipulation tasks. We introduce two key components to topological planning: a latent-flow network structure, and using both wrist and head cameras to our method, and show that they enable robust mobile manipulation performance under data-limited regimes. However, the planning step to generate the next goal to reach adds additional computational burdens during deployment, compared to parametric methods like behavior cloning. Though negligible in our real-world experiments, computation complexity can further increase when more training data is collected and included in the topological graph. This poses a fundamental limit to our method. In future work, we hope to explore means to mitigate this problem by using parametric models possibly trained on large offline datasets, instead of non-parametric ones, to plan and generate next goals to guide the policy from drifting away from confident regions.

REFERENCES

[1] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, A. Casal, and A. Baader, "Force strategies for cooperative tasks in multiple mobile manipulation systems," in *Robotics Research: The Seventh International Symposium*, 1996.

[2] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *ICRA*, 2005.

[3] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the atlas robot," in *International Conference on Humanoid Robots*, 2014.

[4] M. Bajracharya, J. Borders, D. Helmick, T. Kollar, M. Laskey, J. Leichty, J. Ma, U. Nagarajan, A. Ochiai, J. Petersen *et al.*, "A mobile manipulation system for one-shot teaching of complex tasks in homes," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[5] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski, "The darpa robotics challenge finals: Results and perspectives," *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*, 2018.

[6] C. G. Atkeson, P. B. Benzun, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin *et al.*, "What happened at the darpa robotics challenge finals," *The DARPA robotics challenge finals: Humanoid robots to the rescue*.

[7] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman *et al.*, "Team ihmc's lessons learned from the darpa robotics challenge trials," *Journal of Field Robotics*, 2015.

[8] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*, 2023.

[9] S. James, K. Wada, T. Laidlow, and A. J. Davison, "Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[10] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, 2016.

[11] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "Viola: Imitation learning for vision-based manipulation with object proposal priors," *arXiv preprint arXiv:2210.11339*, 2022.

[12] 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.

[13] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*, 2022.

[14] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Conference on robot learning*. PMLR, 2017, pp. 357–368.

[15] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[16] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *2015 IEEE international conference on robotics and automation (ICRA)*, 2015.

[17] S. Stepputtis, M. Bandari, S. Schaal, and H. B. Amor, "A system for imitation learning of contact-rich bimanual manipulation policies," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

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

[19] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *CoRL*, 2022.

[20] 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*, 2011.

[21] S. Tu, A. Robey, T. Zhang, and N. Matni, "On the sample complexity of stability constrained imitation learning," in *Learning for Dynamics and Control Conference*, 2022.

[22] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[23] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," *Advances in neural information processing systems*, 2022.

[24] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conference on robot learning*, 2020.

[25] Z. J. Cui, Y. Wang, N. Muhammad, L. Pinto *et al.*, "From play to policy: Conditional behavior generation from uncurated robot data," *arXiv preprint arXiv:2210.10047*, 2022.

[26] S. Jauhri, J. Peters, and G. Chalvatzaki, "Robot learning of mobile manipulation with reachability behavior priors," *IEEE Robotics and Automation Letters*, 2022.

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

[28] N. Yokoyama, A. W. Clegg, E. Undersander, S. Ha, D. Batra, and A. Rai, "Adaptive skill coordination for robotic mobile manipulation," *arXiv preprint arXiv:2304.00410*, 2023.

[29] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese, "Relmogen: Integrating motion generation in reinforcement learning for mobile manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[30] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: learning a unified policy for manipulation and locomotion," in *Conference on Robot Learning*, 2022.

[31] J. Hu, P. Stone, and R. Martín-Martín, "Causal policy gradient for whole-body mobile manipulation," *arXiv preprint arXiv:2305.04866*, 2023.

[32] C. Sun, J. Orbik, C. M. Devin, B. H. Yang, A. Gupta, G. Berseth, and S. Levine, "Fully autonomous real-world reinforcement learning with applications to mobile manipulation," in *Conference on Robot Learning*, 2021.

[33] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," *arXiv preprint arXiv:2305.05658*, 2023.

[34] H. Ito, K. Yamamoto, H. Mori, and T. Ogata, "Efficient multitask learning with an embodied predictive model for door opening and entry with whole-body control," *Science Robotics*, vol. 7, no. 65, p. eaax8177, 2022.

[35] Y. Du, D. Ho, A. Alemi, E. Jang, and M. Khansari, "Bayesian imitation learning for end-to-end mobile manipulation," in *International Conference on Machine Learning*. PMLR, 2022, pp. 5531–5546.

[36] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[37] S. Thrun and A. Bücken, "Integrating grid-based and topological maps for mobile robot navigation," in *Proceedings of the national conference on artificial intelligence*, 1996.

[38] D. Kortenkamp and T. Weymouth, "Topological mapping for mobile robots using a combination of sonar and vision sensing," in *AAAI*, 1994.

[39] S. P. Engelson and D. V. McDermott, "Error correction in mobile robot map learning," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, 1992.

[40] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and autonomous systems*, 1991.

[41] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Scaling local control to large-scale topological navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[42] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," *arXiv preprint arXiv:1803.00653*, 2018.

[43] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine, "Ving: Learning open-world navigation with visual goals," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[44] Y. He, I. Fang, Y. Li, R. B. Shah, and C. Feng, "Metric-free exploration for topological mapping by task and motion imitation in feature space," *arXiv preprint arXiv:2303.09192*, 2023.

[45] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," *Advances in Neural Information Processing Systems*, 2019.

[46] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[47] M. Hahn, D. S. Chaplot, S. Tulsiani, M. Mukadam, J. M. Rehg, and A. Gupta, "No rl, no simulation: Learning to navigate without navigating," *Advances in Neural Information Processing Systems*, 2021.

[48] D. Shah and S. Levine, "Viking: Vision-based kilometer-scale navigation with geographic hints," *RSS*, 2022.

[49] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine, "Rapid exploration for open-world navigation with latent goal models," *CoRL*, 2021.

[50] E. Beeching, J. Dibangoye, O. Simonin, and C. Wolf, "Learning to plan with uncertain topological maps," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 2020.

[51] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. C. Julian, C. Finn, and S. Levine, "Actionable models: Unsupervised offline reinforcement learning of robotic skills," in *International Conference on Machine Learning*, 2021.

[52] R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak, "Discovering and achieving goals via world models," *Advances in Neural Information Processing Systems*, 2021.

[53] W. Shang, X. Wang, A. Srinivas, A. Rajeswaran, Y. Gao, P. Abbeel, and M. Laskin, "Reinforcement learning with latent flow," *Advances in Neural Information Processing Systems*, 2021.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[55] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, 2016.

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, 2014.

[57] K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn, "Vision-based manipulators need to also see from their hands," *arXiv preprint arXiv:2203.12677*, 2022.

[58] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[59] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[60] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell, "Real-world robot learning with masked visual pre-training," *CoRL*, 2022.

[61] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik, "Masked visual pre-training for motor control," *arXiv preprint arXiv:2203.06173*, 2022.

[62] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta, "Pyrobot: An open-source robotics framework for research and benchmarking," *arXiv preprint arXiv:1906.08236*, 2019.

[63] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning.(2016)," *URL http://pybullet. org*, 2016.