Mobi-π: Mobilizing Your Robot Learning Policy

Jingyun Yang[1], **Isabella Huang**[*2], **Brandon Vu**[*1]
Max Bajracharya[2], **Rika Antonova**[3], **Jeannette Bohg**[1]
[1]Stanford University [2]Toyota Research Institute [3]University of Cambridge
https://mobipi.github.io[†]

Abstract: Learned visuomotor policies are capable of performing increasingly complex manipulation tasks. However, most of these policies are trained on data collected from limited robot positions and camera viewpoints. This leads to poor generalization to novel robot positions, which limits the use of these policies on mobile platforms, especially for precise tasks like pressing buttons or turning faucets. In this work, we formulate the *policy mobilization* problem: find a mobile robot base pose in a novel environment that is in distribution with respect to a manipulation policy trained on a limited set of camera viewpoints. Compared to retraining the policy itself to be more robust to unseen robot base pose initializations, policy mobilization decouples navigation from manipulation and thus does not require additional demonstrations. Crucially, this problem formulation complements existing efforts to improve manipulation policy robustness to novel viewpoints and remains compatible with them. To study policy mobilization, we introduce the Mobi-π framework, which includes: (1) metrics that quantify the difficulty of mobilizing a given policy, (2) a suite of simulated mobile manipulation tasks based on RoboCasa to evaluate policy mobilization, (3) visualization tools for analysis, and (4) several baseline methods. We also propose a novel approach that bridges navigation and manipulation by optimizing the robot's base pose to align with an in-distribution base pose for a learned policy. Our approach utilizes 3D Gaussian Splatting for novel view synthesis, a score function to evaluate pose suitability, and sampling-based optimization to identify optimal robot poses. We show that our approach outperforms baselines in both simulation and real-world environments, demonstrating its effectiveness for policy mobilization.

Figure 1: **Introducing policy mobilization.** (a) Assume a visuomotor policy π trained from one or a set of limited camera poses. (b) We are interested in running π on a mobile platform, where the robot is initialized at a random pose and needs to navigate before running π. (c) **Naively navigating towards the object of interest** and executing the manipulation policy is likely to result in an out-of-distribution initialization for the policy, leading to failures. Left: The robot gets too close and pushes the object inward with the arm. Right: suboptimal heading makes the object unreachable for the left arm. (d) **Improving the robustness of the policy** requires extensive data collection to cover all possible robot base pose initializations. (e) We define the new problem of **policy mobilization** which aims to find the optimal robot pose that leads to an in-distribution viewpoint for executing π, achieving task success without the need to collect additional demonstrations.

1 Introduction

Recent advances in learning visuomotor manipulation policies have enabled robots to execute increasingly complex manipulation tasks using imitation and reinforcement learning [1–7]. However, most manipulation policies are trained on data collected from a limited set of robot base poses and camera viewpoints, leading to a tight coupling between the policy's input distribution and the configurations seen during training. When such a policy is deployed on a mobile robot, changes in the robot's base pose inherently alter its viewpoint. This can cause both the visual input and the reachable workspace to drift outside the policy's training distribution, severely degrading performance.

[*] These authors contributed equally.
[†] Contact: jingyuny@stanford.edu

Given the abundance of stationary robot data, it is highly desirable to reuse existing datasets and pre-trained policies for mobile manipulation. While retraining a policy to handle a wide range of base poses and viewpoints is a straightforward solution, it is also costly—requiring extensive data collection across diverse positions and scene variations [3, 8]. Moreover, as policies are deployed in mobile settings, their field of view expands, demanding stronger robustness to scene-level variations. An alternative strategy is to decouple navigation and manipulation by chaining a navigation module with the trained manipulation policy. However, most prior work in navigation specifies goals via language instructions or object locations [9–18] rather than selecting base poses that are *policy-compatible*. As a result, the robot often navigates to viewpoints that lie outside of the policy's training distribution, leading to unreliable performance.

In this work, we introduce the *policy mobilization problem*: determine a mobile robot base pose in a novel environment that aligns with the training distribution of a manipulation policy learned from a limited set of camera viewpoints. Addressing this problem decouples navigation from manipulation policy training while ensuring that navigation remains *policy-aware*—that is, it selects poses that are compatible with the policy's visual input distribution. This enables effective policy deployment in unseen environments without requiring extensive additional data collection. Consequently, policy mobilization methods enhance the viewpoint robustness of manipulation policies, independent of the policy's original tolerance to viewpoint variations.

To investigate the policy mobilization problem, we present the **Mobi-π framework**. Mobi-π provides tools for systematic study, including: (1) metrics to quantify the difficulty of mobilizing a given policy, (2) a suite of simulated mobile manipulation tasks built on RoboCasa [19], (3), visualization tools for qualitative analysis, and (4) several baseline methods for benchmarking. We first evaluate *non-policy aware* baselines that decouple navigation and manipulation without considering that policies perform better from specific base poses. These approaches frequently fail due to the poor alignment with the policy's training distribution. We also examine *policy-aware* baselines that learn to connect navigation and manipulation using large amounts of additional training data and show that they still struggle to generalize to unseen room layouts.

To address these limitations, we propose a method that searches for robot base poses likely to yield successful policy execution. Our approach (a) represents the scene using 3D Gaussian Splatting [20], (b) uses differentiable rendering to evaluate whether a candidate pose is in-distribution, provides visibility of task-relevant objects, and avoids collisions, and (c) optimizes over poses with a sampling-based procedure to identify the most suitable robot base location. We validate our approach on both the Mobi-π simulation suite and three real-world mobile manipulation tasks. Our method outperforms the several baselines in various simulation and real-world experiments, enabling effective deployment of manipulation policies trained solely on stationary robot data.

2 Related Work

Semantic navigation. In semantic navigation, robots navigate environments to find specific objects or regions. Prior work [9, 10, 21–25] use classical methods like SLAM [26] to build and explore semantic maps, while recent methods use learning-based approaches [11–14, 16, 17]. Other methods predict semantic top-down maps [18, 27, 28] and navigate using waypoint planners [29]. These approaches typically stop in an area where the object is visible and close, without ensuring that the viewpoint is suitable for manipulation. In contrast, our method selects navigation targets that maximize downstream manipulation policy success.

Imitation learning. Imitation learning teaches robots tasks from demonstrations [1, 30–35], with recent efforts scaling to generalist models [3–8, 36–38]. Training such policies for both navigation and manipulation requires massive data. Instead, we use imitation learning for the manipulation policy and leverage existing navigation methods to move the robot to an in-distribution robot pose ensuring policy success. Our insight is that by selecting suitable poses for manipulation, we reduce the need for collecting more data from many robot poses or viewpoints and improve deployment in novel scenes. We focus on single-task policies and leave multi-task extensions to future work.

Mobile manipulation. Prior work has studied various ways to bring learned policies to mobile manipulation [34, 39–45]. Some methods [3, 34] rely on end-to-end learning but require large-scale data. Others [39–41, 43] assume known initial base poses for a given task. ARPlace [46] predicts valid robot poses for running skills but focuses more on reachability rather than viewpoint restric-

tions for these skills. ASC [47] trains coordinated mobile pick-and-place skills with RL in simulation but requires highly accurate simulators, which is often difficult to obtain for complex real-world tasks with deformables or complex multi-object interactions. Some works use reachability maps or abstract planners [48, 49], but ignore that policies may be viewpoint sensitive. Our approach directly addresses this by finding robot base and camera poses that enable manipulation policy success. This "mobilizes" policies with fixed initial base poses, making them usable on mobile robots in complex environments, including those with deformable objects or multi-object interactions.

In-distribution detection. Policy mobilization involves finding an in-distribution initial condition for a given policy. Prior work has extensively studied in-distribution detection or anomaly detection, covering classification-based methods [50–54], reconstruction methods [55, 56], methods that utilize generative adversarial networks (GANs) [57, 58], and diffusion-based methods [59]. Nearly all these methods assume a passive setting where they detect whether a given input is in-distribution without solving the problem of how an agent can find a sample that is more in-distribution. Recent robotics works have started to address out-of-distribution (OOD) detection [60–62] but treat the perception module as a passive component without actively updating the robot's camera or base pose to mitigate OOD scenarios. Our work closes this gap, marrying in-distribution detection with active robot pose optimization in a mobile manipulation context.

Hierarchical methods for policy learning. Our setting resembles Hierarchical Reinforcement Learning (HRL), where an option [63] includes an initiation set, a policy, and termination function. HRL methods [64–68] often learn these jointly but have limited success in complex settings. Instead of learning all elements of an option, we focus on selecting optimal robot poses for pre-trained manipulation policies in realistic environments.

Training view-robust policies. Related to the problem setting of our work, prior work has studied methods of obtaining view-robust manipulation policies. One approach achieve this by collecting multi-view data in simulation [69–71], but suffers from the sim-to-real transfer challenge. Some works achieves view-robustness by using 3D representations, but training a policy to use 3D input requires data with well-calibrated camera extrinsics [72–74], which is not readily available in many robot learning datasets today. Some works achieve real-world view robustness by manually collecting multi-view data in the real world [8], but this requires extensive human effort. There are proposals for using single-view novel view synthesis to bypass the need for multi-view data [75], but methods for novel view synthesis from a single image struggle to effectively render views that are too different from the given viewpoint. From a view robustness perspective, our work takes advantage of the mobility of the robot so the robot can navigate to a base pose with an appropriate camera viewpoint for subsequent policy execution, thus requiring no extensive, additional data collection.

3 The Mobi-π Framework

Given a visuomotor manipulation policy π trained with imitation learning [1, 32] on data recorded from one or a limited set of robot poses or viewpoints, we are interested in deploying π on a mobile robot in an unseen scene S.

To study policy mobilization, we propose the Mobi-π framework, which includes metrics to quantify the difficulty of mobilizing a given policy (Section 3.1), a suite of simulated mobile manipulation tasks to benchmark methods and policies for mobilization (Section 3.2), as well as baseline methods (Section 3.3).

3.1 Mobilization Feasibility Metrics

Given a policy for a task, we want to know how hard it is to mobilize it. We say a policy for a task is "difficult to mobilize" if running this policy on a mobile platform results in a large performance drop *relative to* its in-distribution performance. To quantify mobilization feasibility, we propose two distinct and complementary metrics as part of the Mobi-π-framework.

Spatial mobilization feasibility metric. First, we quantify mobilization feasibility by measuring how robot base pose perturbations affect the task success of manipulation policies trained with fixed initial base poses. Specifically, we measure the task success $S_\pi(\sigma)$ of policy π given Gaussian noise $N \sim (\mu = 0, \sigma^2)$ applied to the base pose. We then fit an exponential curve using the following form: $S_\pi(\sigma) = C_0 \cdot e^{-\gamma\sigma}$, where C_0 and γ are parameters to be fitted. From this fitted curve, we

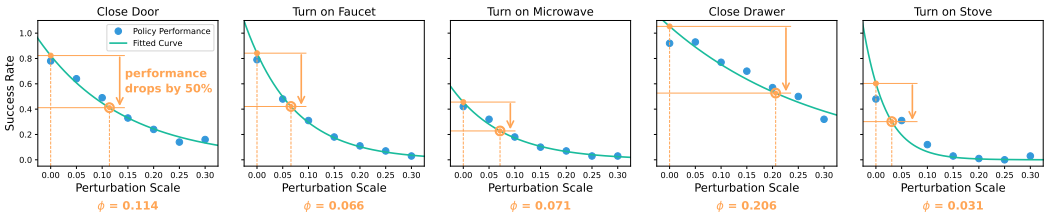

Figure 2: **Spatial mobilization feasibility.** We show the performance of running the given policy with Gaussian noise applied to the initial robot base pose at different standard deviations. The x-axis represents the standard deviation σ of the applied noise (which we also call "perturbation scale"). The y-axis represents the success rate of the policy evaluated for 150 episodes per blue dot. The green curves represent the fitted performance decay curve. Given the fitted curve, we can find the perturbation scale ϕ (in meters) that causes the policy success rate to drop by half. The smaller ϕ, the harder it is to mobilize this policy since policy performance quickly decays with an increasing deviation from the in-distribution poses.

Figure 3: **Visual mobilization feasibility.** In each task, we evaluate how feasible it is to mobilize a robot learning policy from a visual perspective by computing on average, how much of the view is occupied by objects of interest. We denote this metric as S_v. This is important because the visual appearance of objects of interest will help policy mobilization methods decide a proper robot placement pose for policy execution. If the object of interest only takes up a tiny part of the scene, it is more difficult for a method to precisely position the robot such that the visual viewpoint is proper for policy execution.

extract the *spatial mobilization feasibility metric*: $\phi = \frac{\ln 2}{\gamma}$. This metric ϕ can be interpreted as the deviation of the base pose in meters that causes the policy success rate to drop by half. The higher ϕ, the easier it is to mobilize this policy as the policy performance decays more slowly.

Visual mobilization feasibility metric. The visual appearance of the tasks can also affect mobilization difficulty. For example, if a policy manipulates an object while barely seeing the object in view, it will be harder to find a good location for policy execution. To quantify this, we evaluate how relevant objects of the task are represented in the scene. Formally, we sample N_v images of the scene at near-optimal poses and compute the average percentage S_v of the images occupied by the objects of interest in the task. The visual mobilization feasibility metric can be evaluated for methods that consider aligning the robot's viewpoint with the policy's capabilities during base placement. Naive baselines like LeLaN [17] and VLFM [18] are not policy-aware, so this metric does not apply.

3.2 Simulated Task Suite for Evaluating Policy Mobilization

Part of the Mobi-π framework is a suite of simulated mobile manipulation tasks based on RoboCasa [19] that will allow us to study the policy mobilization problem and benchmark different methods. Our task suite includes five challenging tasks illustrated in Figure 4. For each task, we train a standard image-based imitation learning policy [32] from 300 demonstrations. These demos are generated using MimicGen [76] with procedurally generated textures in 5 training room layouts. In each task, we use a Franka Panda robot mounted on an Omron mobile base initialized from the same base pose. Each task comes with a language description and a horizon of 500 steps. The goal of the benchmark is to deploy these policies into 5 held-out scenes with unseen room layouts and textures. Note that since these tasks come originally from RoboCasa, the tasks and policies are not designed for the problem of policy mobilization. Therefore, this benchmark can serve as an unbiased challenge task suite for testing methods for policy mobilization.

Quantifying mobilization feasibility of benchmark tasks. We plot the spatial mobilization feasibility metrics of the policies in the five simulated kitchen tasks in Fig. 2. We observe that the most challenging task to mobilize based on our metric is the *Turn on Stove* task ($\phi = 0.031$). This reflects the difficulty to precisely turn a small knob which has limited contact area. Figure 3 shows the visual mobilization feasibility metric evaluated in the five environments. Results show that visual mobilization feasibilities do not necessarily match spatial mobilization feasibilities. More specifically, the *Close Drawer* task, which appears to be easy to mobilize spatially, turns out to be the most

Figure 4: **A suite of simulated tasks for benchmarking performance of policy mobilization methods.** We pick five single-stage manipulation tasks from the RoboCasa benchmark [19]: (1) *Close Door:* push or close the microwave or cabinet door using the robot gripper. (2) *Turn on Faucet:* precisely move the faucet handle to turn on water. (3) *Turn on Microwave*: push a small button on the microwave. (4) *Close Drawer*: close an open drawer next to the robot. (5) *Turn on Stove*: turn a specific knob out of six knobs on the stove. Here, we visualize two successful rollouts for each task. Note that episodes are executed in environments with different layouts and textures. In the *Close Door* task, the robot might encounter either a cabinet door or a microwave door. The *Turn on Faucet*, *Turn on Microwave*, and *Turn on Stove* tasks require accurate robot placement to be executed correctly. In the *Close Drawer* task, the drawer can be either on the left or right side of the robot.

difficult in terms of visual mobilization feasibility metric. This is expected because even in spatially tolerant tasks, using proper visual features to find optimal policy initialization poses can be difficult. We show in experiments how the mobilization metrics are correlated with method performance in more detail.

3.3 Baseline Methods for Policy Mobilization

We include three baselines for policy mobilization in our Mobi-π framework. We divide those into two categories. The first type of baselines navigates to the object of interest *without considering the manipulation policy's capabilities*. These methods are *not policy-aware*. Representative methods include: (1) *LeLaN* [17], a language-conditioned navigation model trained from large-scale videos. Since the pre-trained LeLaN model is trained on only real-world data, we fine-tune the pre-trained LeLaN model with 7.5k episodes of in-domain, randomized navigation data in simulation. At test time, we run LeLaN for a fixed number of steps to approach the object while using the scene point cloud for collision checking, and then execute the target policy. (2) *VLFM* [18], a zero-shot navigation model that explores the scene, uses a ground-truth detector to locate the object, selects a point on its unprojected point cloud, and navigates to it using RRT and the scene point cloud for collision checking. As shown in our experiments, these types of methods often fail to find suitable base poses as they are not taking manipulation policy performance into account, resulting in failed policy executions.

The second type of baselines is *policy-aware* and leverages large-scale data to connect navigation with manipulation. A representative method is *BC w/ Nav*: a Behavior Transformer [32] trained to jointly perform navigation and manipulation using combined demonstrations. We provide 1,500 demonstrations per task, $5\times$ the number of demos used compared to our base manipulation policy. We show in experiments that these methods generalize poorly to unseen layouts despite extensive data collection. See supplementary materials for baseline details.

4 A Novel Method for Policy Mobilization

We introduce a novel method for policy mobilization, illustrated in Figure 5. We want our method to satisfy the following criteria: (1) be policy-aware, (2) do not require extensive data collection for policy learning, and (3) operate in arbitrary unseen room layouts.

Assumptions. We need a setting where our method is aware of the policy's capabilities without collecting large-scale navigation data. To make this possible, we assume access to the initial image frames of each demonstration in the training dataset: $\mathcal{D}_{\text{initial}} = \{[o_1^1, \ldots, o_1^{N_{DS}}]\}$ used for training the policy, where o_t^k is the t-th observation of the k-th training episode and N_{DS} denotes the number of trajectories in the dataset. To successfully operate in any unseen room layout, our method needs a mechanism for quickly learning about the test-time room layout. Therefore, we assume that before executing any policy evaluation, our method is allowed to build a map of the scene by looking around the scene *without* interacting with it. For simplicity, we assume that the initial robot arm pose is fixed and known for each task. We also assume known camera intrinsics and a known, fixed relative transform between the robot base and the camera. This implies that optimizing robot base poses is the same as optimizing camera viewpoints in our setting. At the start of a mobile manipulation episode, the mobile robot is placed at a pose with an unobstructed view of the target object $p_{\text{init}} \in SO(2)$. The goal is to navigate to a target pose p_π where the policy π can be successfully executed.

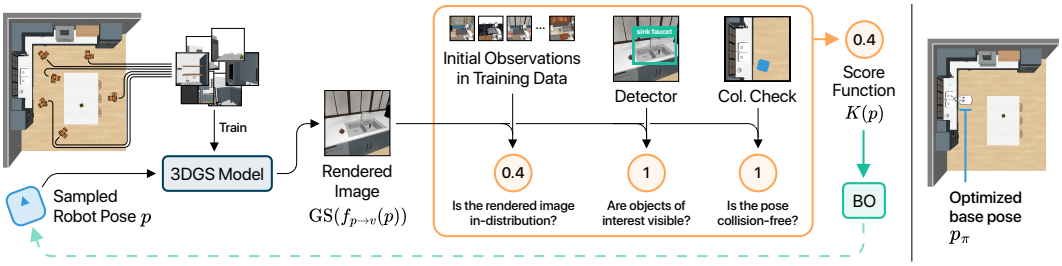

Figure 5: **Overview of our proposed proof-of-concept method.** The goal of our method is to find a proper robot pose p for policy initialization such that a given policy π can be successfully executed in a scene S. Our method has three components: (1) a proposed robot pose p is converted to its corresponding camera pose v via $v = f_{p\to v}(p)$ and then sent into a 3D Gaussian Splatting model to produce a rendered image; (2) the rendered image and the proposed robot pose are passed into a hybrid score function K that predicts whether the proposed robot pose is suitable for policy initialization; (3) the score function output is used to update a Bayesian Optimization algorithm, which repeatedly proposes new sampled robot poses to be evaluated and updates its internal belief on which robot poses are optimal for robot policy execution. The hybrid score function is composed of multiple parts powered by visual foundation models so that it can make judgments of whether the proposed pose (a) is in distribution with respect to the training dataset; (b) has the objects of interest in view; and (c) is collision-free.

4.1 Representing the Scene with 3D Gaussian Splatting

When the robot enters a previously unseen scene, it must first acquire a representation of the environment before identifying base poses from which the manipulation policy π can be executed effectively. To this end, we employ 3D Gaussian Splatting (3DGS) [20], a technique that enables high-quality, novel-view synthesis from arbitrary viewpoints. Given a set of RGB-D images captured from various viewpoints, 3DGS constructs a scene representation composed of Gaussians $\mathcal{G} = \{g_i\}_{i=1}^{N_g}$ where each Gaussian $g_i = (\mu_i, \Sigma_i, o_i, c_i)$ encodes the 3D position $\mu_i \in \mathbb{R}^3$, spatial uncertainty via covariance matrix $\Sigma_i \in \mathbb{R}^{3\times3}$, opacity $o_i \in \mathbb{R}$, and view-dependent color $c_i \in SH(4)$. To render an image from a novel camera pose $v \in SO(3)$, each Gaussian is projected to the image plane [77] and splatted using α-blending to generate the final image $I = \mathrm{GS}(v)$.

We construct the 3DGS model using 1,000 RGB-D images collected throughout the scene. This image acquisition process takes less than five minutes and involves driving the robot around the scene while varying camera height and viewing angle. We then reconstruct a point cloud from the depth images relative to the world frame to initialize the Gaussians. We then train the 3DGS models using `nerfstudio` [78] for 30,000 update steps.

4.2 Hybrid Score Function for Estimating Policy Execution Viability

To estimate how likely a policy will be successful from a certain base pose p, we define a hybrid score function $K(p)$ that incorporates three key factors: (1) the rendered view should be in-distribution with respect to the training data, (2) the object of interest should be visible; (3) the robot pose should be free of collisions. To evaluate whether a rendered view is in-distribution, we leverage the initial camera observations $\{o_1^k\}_{k=1}^{N_{DS}}$ in the demonstration dataset. We compare the candidate view against these initial camera observations in a learned feature space. Among the image embeddings we evaluated, dense descriptors from DINO [79] proved most effective in distinguishing in- from out-of-distribution views and are robust to variations in scene texture. We precompute DINO features for all initial image observation, yielding$\{\mathrm{DINO}(o_1^k)\}_{k=1}^{N_{DS}}$. For a queried robot pose p, we render the corresponding image using our 3DGS models and compute its dense descriptors as $\mathrm{DINO}(\mathrm{GS}(f_{p\to v}(p)))$, where $f_{p\to v}$ denotes the transformation from base pose p to camera pose v. The encoder $\mathrm{DINO}(I)$ maps an image I of size $H \times W$ to a feature tensor in $H' \times W' \times D$. To assess similarity, we apply K-nearest neighbors (KNN) in the feature space to compute the average L2 distance to the demonstration features, yielding an in-distribution score:

$$K_{\mathrm{id}}(p) = \mathrm{KNN}\Big(\mathrm{DINO}\big(\mathrm{GS}(f_{p\to v}(p))\big), \{\mathrm{DINO}(o_1^k)\}_{k=1}^{N_{DS}}\Big). \tag{1}$$

To increase robustness, we augment $K_{\mathrm{id}} \in \{0, 1\}$ with two additional binary checks. First, K_{id} can return high scores for views that are texturally similar to the training set, even if they fail to include the task relevant object. To mitigate this, we verify object visibility using a vision-language model MiniCPM-v2 [80] and define

$$K_{\mathrm{obj}}(p) = \text{MiniCPM-v2}(\mathrm{GS}(f_{p\to v}(p))), \tag{2}$$

where $K_{\text{obj}}(p) = 1$ if the object is detected and 0 otherwise. We found this approach to outperform conventional object detectors [81, 82]. Second, we verify that pose p is not in collision with the environment by consulting an occupancy map constructed from the depth data used the train the 3DGS model. We define:

$$K_{\text{col}}(p) = (1 \text{ if pose } p \text{ is vacant else } 0). \tag{3}$$

The final hybrid score combines all three components:

$$K(p) = \begin{cases} K_{\text{id}}(p) & \text{if } K_{\text{obj}}(p) = 1 \;\&\; K_{\text{col}}(p) = 1, \\ 0 & \text{otherwise.} \end{cases}. \tag{4}$$

4.3 Robot Pose Optimization with Bayesian Optimization

Given the score function $K(p)$, our objective is to find the robot base pose p_π that maximizes the likelihood of successful policy execution: $p_\pi = \max_p K(p)$. While some components of $K(p)$ are differentiable, we observe that the overall optimization landscape is non-smooth. In particular, small changes in the camera viewpoint can lead to abrupt variations in the score due to discontinuities introduced by occlusions, changes in object visibility or reachability. These factors make gradient-based optimization unreliable. To address this, we adopt a gradient-free optimization approach and use Bayesian Optimization (BO) to efficiently search for the optimal base pose p_π.

Bayesian Optimization is a sampling-based, gradient-free optimization algorithm designed to find parameters $\mathbf{w}^*$ that maximize a target function $f_{\text{BO}}(\mathbf{w})$. It achieves this by modeling f_{BO} using a surrogate model—typically a Gaussian Process (GP) or random forest—that approximates the underlying function based on observed samples. At each iteration, BO selects new samples from f_{BO} by maximizing an acquisition function, such as the Upper Confidence Bound (UCB) [83], which balances exploration and exploitation. The acquisition function prioritizes regions of the parameter space that are both uncertain and potentially high-scoring, enabling BO to identify promising regions efficiently with a limited number of queries. To optimize the policy optimization score $K(p)$, we initialize BO with $N_{\text{init}}^{(\text{BO})}$ randomly sampled base poses in the scene and evaluate their corresponding scores. We then run $N_{\text{iter}}^{(\text{BO})}$ optimization rounds where each round proposes $N_{\text{batch}}^{(\text{BO})}$ new poses based on the acquisition function. The highest-scoring pose found among all iterations is selected as the final pose for executing the manipulation policy. Please check out the implementation details of our method in the supplementary materials.

5 Experiments

In experiments, we aim to investigate the following key questions: *[Q1]* Do non-policy-aware baselines fail to navigate to base poses that enable successful policy execution? *[Q2]* Do methods trained with large-scale, in-domain navigation data exhibit limited generalization to out-of-distribution room layouts? *[Q3]* Does our proposed method outperform these baselines and consistently identify suitable base poses for successful policy execution? *[Q4]* Are our mobilization feasibility metrics predictive of the actual performance across different methods? *[Q5]* Can our method be effectively deployed in real-world settings?

5.1 Simulation Experiments

Metrics. In addition to reporting the success rate of the baselines (see Section 3.3) and our method, we also report *Manipulation Policy Performance*, which is the success rate of the original, non-mobile robot policy per task without mobilization. Note that the success rates of the base policies vary significantly across tasks. We report the success rate and standard deviation across all tasks.

Evaluation setup. In each task, we evaluate our method and competing baselines in 10 scenes with distinct layouts and textures. For each of the 3 random seeds, we run 50 evaluation episodes, 5 episodes in each scene. In every episode, the robot is randomly placed in the scene with the target object in view. The method needs to navigate and then successfully execute the manipulation policy. We report the mean and standard deviation over the policy execution success rates. Each result value includes a total of 150 evaluation episodes.

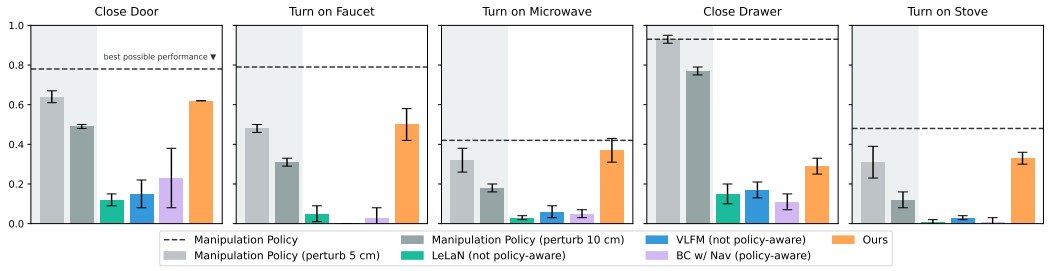

Figure 6: **Simuation experiment results.** We evaluate **LeLaN (non-policy-aware)**, **VLFM (non-policy-aware)**, and **BC w/ Nav (policy-aware, 5x training data)** as well as **our proposed method** in the 5 simulated tasks. For each task, we evaluate the success rate of task execution for 3 seeds and 50 episodes per seed. The plotted bars represent the mean and standard deviation over the results of the 3 seeds. We also report the **manipulation policy performances** with and without perturbation of the starting base pose. In four of five tasks, our method matches up with the performance of the manipulation policy when executed with a 5 cm perturbation at initialization.

Results. We present quantitative results for our method in Figure 6. *LeLaN (non-policy-aware)* and *VLFM (non-policy-aware)* perform poorly because they do not consider which robot poses enable a successful policy rollout (see Figure 7 for qualitative results). This validates *[Q1]*. *BC w/ Nav (policy-aware)* performs poorly in all tasks, showing that it is difficult to train a working mobile manipulation policy that generalizes to out-of-distribution room layouts, validating *[Q2]*. Our method outperforms all baselines in all tasks, achieving better performance in navigating to a pose suitable for executing the downstream policy, resulting in higher task success, answering *[Q3]*.

Relation to mobilization feasibility metrics. Baselines that are not policy-aware perform worst in *Turn on Faucet*, *Turn on Stove*, and *Turn on Microwave*. These are also tasks that have the lowest spatial mobilization feasibility scores (Section 3.1). This suggests that poor spatial tolerance of a manipulation policy makes it challenging for a non-policy-aware method to perform well. Our method's performance also correlates with the visual mobilization feasibility metric. It matches or exceeds *Manipulation Policy (Perturb 5 cm)* across tasks, except *Close Drawer*, which has low visual feasibility. As our method relies more on visual cues, this explains the performance drop, supporting *[Q4]*.

Ablations and analysis. We conduct ablation studies in two simulation environments, *Close Door* and *Turn on Microwave*. Specifically, we wish to understand which visual features are optimal when being used for selecting optimal robot base placement poses during policy mobilization. We consider the following ablated methods: (1) *Ours w/o Feature Descriptor*: this ablated method uses a DINO embedding vector instead of the DINO dense feature descriptor in K_{id} to represent information in an image. (2) *Ours w/ Policy Encoder*: this method uses the ResNet-18 [84] encoder of the given policy to represent an image in K_{id}.

	Close Door	Turn on Microwave
Ours	**0.62 ± 0.00**	**0.37 ± 0.06**
Ours w/o feature descriptor	0.41 ± 0.05	0.07 ± 0.08
Ours w/ policy encoder	0.11 ± 0.07	0.00 ± 0.00

Table 1: **Simulation ablation experiment results.**

We present the results of this ablation in Table 1. *Ours w/o Feature Descriptor* underperforms compared to our method, as the 1D DINO embedding lacks spatial cues necessary for distinguishing camera views. *Ours w/ Policy Encoder* also performs poorly, possibly because it focuses on task-relevant scene features, ignoring view-dependent cues. Since the policy is trained only with action-related losses, it may not capture the information needed to guide robot placement.

We also design visualization tools to show the qualitative performance of competing methods. In Figure 7, we show top-down maps of the simulated benchmark environment overlaid with arrows to illustrate the navigation end poses achieved by different methods. See more visualizations in the supplementary materials.

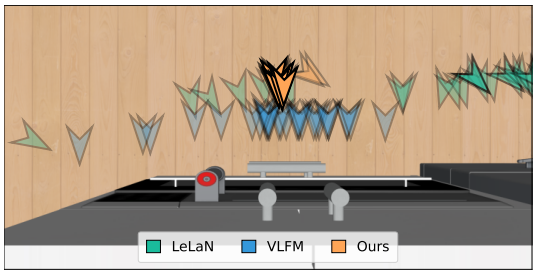

Figure 7: **Visualization of qualitative performances in simulated experiments.** This plot shows achieved navigation end poses of the robots produced by multiple competing methods in a sample scene in the *Turn on Microwave* task. The background plots the top-down map of the environment where the robot needs to run the designated policy. Each arrow represents a robot pose achieved by a method after navigation. The colors of the arrows represent method names. Semi-transparent arrows indicate failures and opaque arrows indicate successes. In the plot, our method consistently reaches the microwave door at an appropriate heading, while baselines fail to do so.

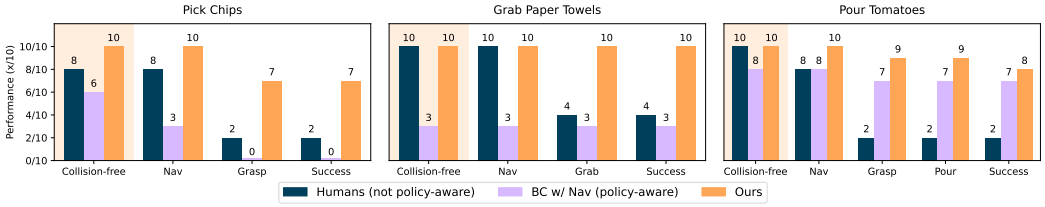

Figure 8: **Real-robot experiment results.** For each task and method evaluated, we track the incidence rates of certain performance milestones (i.e., navigation success, grasp success, full success). We also track the percentage of collision-free episodes. We consider a navigation attempt successful if the robot drives to a pose that not only faces the target object, but also is within a distance of 50 cm from the ideal target pose. A grasp success is defined to be when the robot stably grasps the object of interest (i.e. the chip bag, the paper towels, or the tray of tomatoes), and full success is counted only if the entire task is completed.

5.2 Real Robot Experiments

To validate *[Q5]*, we show a series of real robot experiments where a mobile robot performs various manipulation tasks in a mock grocery store. We pick three tasks that involve diverse objects and contact modes as pictured in Figure 9.

Data collection and training. We collect 30 human-teleoperated demonstrations for the *Pick Chips* and *Grab Paper Towels* task and 50 for the *Pour Tomatoes* task. We use a Basler camera mounted on the mobile robot to collect visual information and record both the images and the base and arm poses at a frequency of 10 Hz. We use the collected data to train a Diffusion Policy [1]. See supplementary materials for more robot setup and training details.

Baselines and rollouts. We initialize the robot at 10 random poses located 0.5 to 1.5 meters from the target policy execution pose, with the constraint that the target scene must remain within the robot's field of view. From each initialization, we execute both our proposed method and the baseline methods. For real-world experiments, we compare against one policy-aware and one non-policy-aware baseline. As the policy-aware baseline, we use *BC w/ Nav*. To construct a strong non-policy aware baseline, we introduce a *Human* baseline: participants are asked to manually drive the robot to the location they believe will maximize task success, without being given any information about how the manipulation policy was trained. Additional details about the human baseline are provided in the supplementary materials.

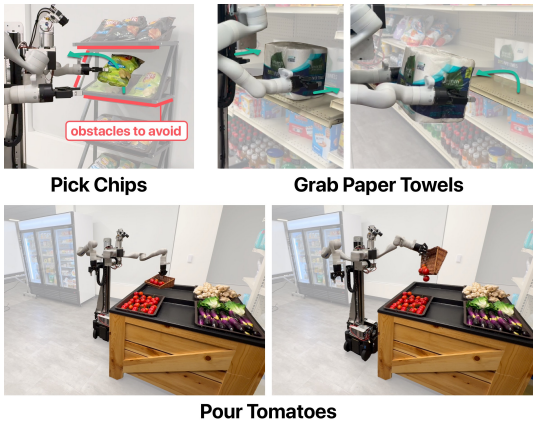

Figure 9: **Real-world tasks.** (1) *Pick Chips*: the robot retrieves a deformable bag of chips from a shelf. (2) *Bimanual Grab Paper Towels*: the robot squeezes a pack of paper towels on a shelf between the two arms using non-prehensile manipulation, lifts it up, and pulls it off the shelf. (3) *Pour Tomatoes*: the robot picks up a tray of tomatoes on the left side of a produce bin, moves rightward via base motions, and pours the tomatoes from the tray into a basket. See videos on our website.

Quantitative results. Quantitative results from the real-robot experiments are shown in Figure 8. The *Human* baseline displays an interesting pattern: although participants successfully navigate the robot near objects of interest, their lack of knowledge about the manipulation policy's capabilities often results in suboptimal base poses, leading to low success rates. The *BC w/ Nav* baseline achieves higher performance but frequently positions the robot inappropriately–too close or poorly oriented realtive to the target object–resulting in failed manipulation attempts. In contrast, our method consistently outperforms both baselines, reliably identifying base poses that enable successful policy execution.

Qualitative results. Aside from quantitative results, we also demonstrate real-world deployment of our method in a variety of interesting ways (see Figure 10).

First, in a real kitchen setting, we show that by querying our method multiple times, our method is capable of finding proper initial robot poses for multiple drawer closing skill executions in the scene, thus unlocking manipulation skill sequencing in the wild. During each round of base pose optimization, we avoid sampling poses that are close to the best poses found by previous optimization rounds to avoid running the drawer closing skill at the same location more than once. We assume that we know how many times the drawer closing skill needs to be executed.

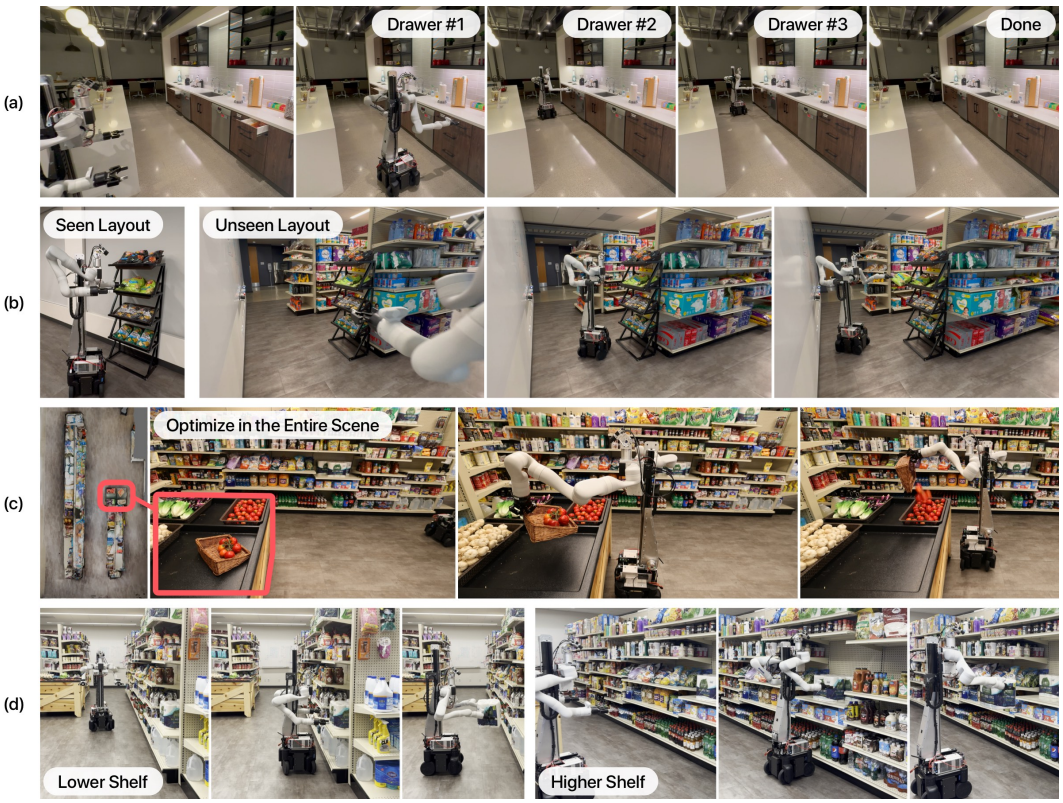

Figure 10: **Real-world qualitative results.** (a) In a real-world kitchen scene, we show that our method can chain a sequence of drawer-closing policy executions. (b) In the *Pick Chips* task, we show that our method achieves zero-shot generalization to unseen scene layouts. (c) In the *Pour Tomatoes* task, we show that our method is capable of operating in large spaces that the policies have not explored during training. The top-down image on the left shows the space in which our method performs its optimizations. This image is generated via a scan of the scene produced by the PolyCam iPhone app and is for visualization purposes only. (d) In the *Bimanual Grab Paper Towels* task, we show that our method can be easily extended to optimize for robot and camera height as well.

Second, we show that our method generalizes zero-shot to unseen scene layouts in the *Pick Chips* task. This is accomplished by simply running our method in the new scene, without any change in the policy or hyperparameters of our method.

Third, we demonstrate in the *Pour Tomatoes* task that our method is capable of operating in a large scene. We create a 3D Gaussian Splatting model of the entire grocery store scene, which spans 6×10 meters in size. We then query our method for the optimal starting robot base pose. Our method successfully completes navigation and manipulation policy execution in this large scene.

Finally, we show that our method can be easily extended to adapt to unseen object heights. To achieve this, we simply add one more delta robot height parameter to the optimization process, and command the robot to move to the target height instructed by our method before starting policy execution. We find that our method can flexibly determine the proper height to run the *Bimanual Grab Paper Towel* policy when the object is placed on shelves with different heights.

6 Conclusion

In this work, we presented Mobi-π, a framework for studying policy mobilization. We introduced a formal problem definition of policy mobilization, presented a spatial and visual metric for quantifying the feasibility of mobilization, and provided an evaluation platform consisting of simulation benchmarks, visualization tools, and baselines. We showcased a promising approach to solve this problem leveraging a 3D Gaussian Splatting model, a hybrid score function, and sampling-based pose optimization to identify suitable robot poses for policy execution. We demonstrated our method in simulated kitchen environments as well as a real-robot setup in a mock-grocery environment. This work provides a robust pathway for utilizing countless non-mobile robot policies on mobile platforms, enabling robots to perform intricate tasks in larger and more diverse environments.

7 Limitations and Discussions

While our method reliably identifies suitable initial robot base poses for executing non-mobile manipulation policies on a mobile platform, it does have several limitations.

Dynamic scene changes. Our approach assumes a static environment during policy execution and does not support dynamic scene updates. This limitation becomes apparent when chaining multiple policies in the same workspace: after executing one policy, changes in the environment are not reflected in the scene model, potentially leading to degraded performance in subsequent steps. Addressing this limitation would require integrating a dynamic neural rendering model that can be updated in real-time as the robot interacts with the scene [85, 86].

Access to training data. Another potential limitation is the requirement to access the manipulation policy's training data. However, we emphasize that policy mobilization—by design—requires some degree of policy awareness, which inherently necessitates access to policy-related information. For example, the *BC w/ Nav* baseline achieves this by training a navigation model on additional task-specific data, which can be expensive to collect. In contrast, our method relies only on the initial observation frames from the demonstration dataset, making it less data-intensive. Nevertheless, future work could explore reducing this requirement further—e.g., by sampling a smaller subset of initial frames or, in the absence of training data, using a few successful rollouts of the policy and extracting initial observations from those episodes.

Supporting multi-task policies. Our current study focuses on single-task manipulation policies. Extending policy mobilization to multi-task settings, where the policy is conditioned on goal images or language instructions, is a promising direction for future work.

Fixed base-to-camera transform. Although the policy mobilization problem does not impose strict assumptions on the base-to-camera configuration, our current implementation assumes a fixed transformation between the robot base and its onboard camera. A natural extension of our approach would be to jointly optimize over both the base pose and camera configuration. This could be achieved by making the robot visible from one of the camera views and running the same iterative optimization algorithm with the camera viewpoint as an additional optimization parameter within the existing framework. Exploring this direction is an exciting direction for future research.

Acknowledgments

This work was completed when Jingyun was an intern at Toyota Research Institute. This work was supported by the Toyota Research Institute.

We thank members of the Mobile Manipulation Team at Toyota Research Institute for building the robot used in this work and maintaining related infrastructures. We thank members of the Stanford IPRL lab, Leonidas J. Guibas, Sergey Levine, Shuran Song, Zi-ang Cao, Yufei Ding, Ray Song, Chen Wang, and Haoyu Xiong for their helpful discussions.

References

[1] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, Jul. 2023. DOI: 10.15607/RSS.2023.XIX.016.

[3] A. Brohan *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[4] A. Brohan *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[5] O. M. Team *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[6] M. J. Kim *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[7] K. Black *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[8] A. Khazatsky *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," *arXiv preprint arXiv:2403.12945*, 2024.

[9] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song, "Semantic slam based on object detection and improved octomap," *IEEE Access*, vol. 6, pp. 75 545–75 559, 2018.

[10] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 1689–1696.

[11] M. Chang, A. Gupta, and S. Gupta, "Semantic visual navigation by watching youtube videos," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4283–4294, 2020.

[12] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable objectnav," *arXiv preprint arXiv:2104.04112*, 2021.

[13] M. Deitke *et al.*, "Procthor: Large-scale embodied ai using procedural generation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5982–5994, 2022.

[14] N. Gireesh, D. S. Kiran, S. Banerjee, M. Sridharan, B. Bhowmick, and M. Krishna, "Object goal navigation using data regularized q-learning," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2022, pp. 1092–1097.

[15] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "Clip on wheels: Zero-shot object navigation as object localization and exploration," *arXiv preprint arXiv:2203.10421*, vol. 3, no. 4, p. 7, 2022.

[16] R. Ramrakhya, D. Batra, E. Wijmans, and A. Das, "Pirlnav: Pretraining with imitation and rl finetuning for objectnav," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 896–17 906.

[17] N. Hirose, C. Glossop, A. Sridhar, O. Mees, and S. Levine, "Lelan: Learning a language-conditioned navigation policy from in-the-wild video," in *8th Annual Conference on Robot Learning*.

[18] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, "Vlfm: Vision-language frontier maps for zero-shot semantic navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 42–48.

[19] S. Nasiriany *et al.*, "Robocasa: Large-scale simulation of everyday tasks for generalist robots," in *Robotics: Science and Systems (RSS)*, 2024.

[20] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, Jul. 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[21] B. Kuipers, "The spatial semantic hierarchy," *Artificial intelligence*, vol. 119, no. 1-2, pp. 191–233, 2000.

[22] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli, "Local metrical and global topological maps in the hybrid spatial semantic hierarchy," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, IEEE, vol. 5, 2004, pp. 4845–4851.

[23] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt, "Active visual object search in unknown environments using uncertain semantics," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 986–1002, 2013.

[24] K. Zheng and A. Pronobis, "From pixels to buildings: End-to-end probabilistic deep networks for large-scale semantic mapping," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 3511–3518.

[25] S. Jin, X. Dai, and Q. Meng, ""focusing on the right regions"—guided saliency prediction for visual slam," *Expert Systems with Applications*, vol. 213, p. 119 068, 2023.

[26] I. A. Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A survey of state-of-the-art on visual slam," *Expert Systems with Applications*, vol. 205, p. 117 734, 2022.

[27] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, "Poni: Potential functions for objectgoal navigation with interaction-free learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 890–18 900.

[28] J. Chen, G. Li, S. Kumar, B. Ghanem, and F. Yu, "How to not train your dragon: Training-free embodied object goal navigation with semantic frontiers," *arXiv preprint arXiv:2305.16925*, 2023.

[29] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3d scene graph construction and optimization," *arXiv preprint arXiv:2201.13360*, 2022.

[30] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[31] A. Zeng *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*, PMLR, 2021, pp. 726–747.

[32] 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*, vol. 35, pp. 22 955–22 968, 2022.

[33] J. Aldaco *et al.*, "Aloha 2: An enhanced low-cost hardware for bimanual teleoperation," *arXiv preprint arXiv:2405.02292*, 2024.

[34] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation using low-cost whole-body teleoperation," in *8th Annual Conference on Robot Learning*, 2024.

[35] C. Chi *et al.*, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," *arXiv preprint arXiv:2402.10329*, 2024.

[36] J. Gu *et al.*, "Rt-trajectory: Robotic task generalization via hindsight trajectory sketches," *arXiv preprint arXiv:2311.01977*, 2023.

[37] J. Yang, C. Deng, J. Wu, R. Antonova, L. Guibas, and J. Bohg, "Equivact: Sim (3)-equivariant visuomotor policies beyond rigid object manipulation," in *2024 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2024, pp. 9249–9255.

[38] J. Yang, Z.-a. Cao, C. Deng, R. Antonova, S. Song, and J. Bohg, "Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning," *arXiv preprint arXiv:2407.01479*, 2024.

[39] M. Ahn *et al.*, "Do as i can and not as i say: Grounding language in robotic affordances," in *arXiv preprint arXiv:2204.01691*, 2022.

[40] J. Wu *et al.*, "Tidybot: Personalized robot assistance with large language models," *Autonomous Robots*, vol. 47, no. 8, pp. 1087–1102, 2023.

[41] N. M. M. Shafiullah *et al.*, "On bringing robots home," *arXiv preprint arXiv:2311.16098*, 2023.

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

[43] R.-Z. Qiu *et al.*, "Wildlma: Long horizon loco-manipulation in the wild," *arXiv preprint arXiv:2411.15131*, 2024.

[44] J. Wu *et al.*, "Tidybot++: An open-source holonomic mobile manipulator for robot learning," *arXiv preprint arXiv:2412.10447*, 2024.

[45] Y. Huang, C. Agia, J. Wu, T. Hermans, and J. Bohg, "Points2plans: From point clouds to long-horizon plans with composable relational dynamics," *arXiv preprint arXiv:2408.14769*, 2024.

[46] F. Stulp, A. Fedrizzi, L. Mösenlechner, and M. Beetz, "Learning and reasoning with action-related places for robust mobile manipulation," *J. Artif. Int. Res.*, vol. 43, no. 1, pp. 1–42, Jan. 2012, ISSN: 1076-9757.

[47] N. Yokoyama *et al.*, "Asc: Adaptive skill coordination for robotic mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 779–786, 2023.

[48] S. Jauhri, J. Peters, and G. Chalvatzaki, "Robot learning of mobile manipulation with reachability behavior priors," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8399–8406, 2022.

[49] E. Rosen *et al.*, "Synthesizing navigation abstractions for planning with portable manipulation skills," in *Conference on Robot Learning*, PMLR, 2023, pp. 2278–2287.

[50] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.

[51] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[52] L. Ruff *et al.*, "Deep one-class classification," in *International conference on machine learning*, PMLR, 2018, pp. 4393–4402.

[53] I. Golan and R. El-Yaniv, "Deep anomaly detection using geometric transformations," *Advances in neural information processing systems*, vol. 31, 2018.

[54] S. Fort, J. Ren, and B. Lakshminarayanan, "Exploring the limits of out-of-distribution detection," *Advances in neural information processing systems*, vol. 34, pp. 7068–7081, 2021.

[55] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," *arXiv preprint arXiv:1812.04606*, 2018.

[56] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" *arXiv preprint arXiv:1810.09136*, 2018.

[57] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi, "A survey on gans for anomaly detection," *arXiv preprint arXiv:1906.11632*, 2019.

[58] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "F-anogan: Fast unsupervised anomaly detection with generative adversarial networks," *Medical image analysis*, vol. 54, pp. 30–44, 2019.

[59] J. Liu *et al.*, "A survey on diffusion models for anomaly detection," *arXiv preprint arXiv:2501.11430*, 2025.

[60] A. Farid, S. Veer, and A. Majumdar, "Task-driven out-of-distribution detection with statistical guarantees for robot learning," in *Conference on Robot Learning*, PMLR, 2022, pp. 970–980.

[61] R. Sinha *et al.*, "A system-level view on out-of-distribution data in robotics," *arXiv preprint arXiv:2212.14020*, 2022.

[62] C. Agia *et al.*, "Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress," *arXiv preprint arXiv:2410.04640*, 2024.

[63] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[64] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[65] M. C. Machado, M. G. Bellemare, and M. Bowling, "A laplacian framework for option discovery in reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2017, pp. 2295–2304.

[66] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, "Meta learning shared hierarchies," *arXiv preprint arXiv:1710.09767*, 2017.

[67] A. Bagaria, J. Senthil, M. Slivinski, and G. Konidaris, "Robustly learning composable options in deep reinforcement learning," in *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 2021.

[68] A. Bagaria, B. Abbatematteo, O. Gottesman, M. Corsaro, S. Rammohan, and G. Konidaris, "Effectively learning initiation sets in hierarchical reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 73 441–73 463, 2023.

[69] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2real viewpoint invariant visual servoing by recurrent control," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4691–4699.

[70] N. Hirose, D. Shah, A. Sridhar, and S. Levine, "Exaug: Robot-conditioned navigation policies via geometric experience augmentation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 4077–4084.

[71] Y. Seo, J. Kim, S. James, K. Lee, J. Shin, and P. Abbeel, "Multi-view masked world models for visual robotic manipulation," in *International Conference on Machine Learning*, PMLR, 2023, pp. 30 613–30 632.

[72] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3d: Infinite resolution action detection transformer for robotic manipulation," *arXiv preprint arXiv:2306.17817*, vol. 1, no. 3, 2023.

[73] Y. Zhu, Z. Jiang, P. Stone, and Y. Zhu, "Learning generalizable manipulation policies with object-centric 3d representations," *arXiv preprint arXiv:2310.14386*, 2023.

[74] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," *arXiv preprint arXiv:2403.03954*, 2024.

[75] S. Tian *et al.*, "View-invariant policy learning via zero-shot novel view synthesis," in *8th Annual Conference on Robot Learning*, 2024.

[76] A. Mandlekar *et al.*, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *7th Annual Conference on Robot Learning*, 2023.

[77] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "Ewa volume splatting," in *Proceedings Visualization, 2001. VIS'01.*, IEEE, 2001, pp. 29–538.

[78] M. Tancik *et al.*, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH '23, 2023.

[79] M. Caron *et al.*, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660.

[80] Y. Yao *et al.*, "Minicpm-v: A gpt-4v level mllm on your phone," *arXiv preprint arXiv:2408.01800*, 2024.

[81] M. Minderer *et al.*, "Simple open-vocabulary object detection," in *European conference on computer vision*, Springer, 2022, pp. 728–755.

[82] S. Liu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *European Conference on Computer Vision*, Springer, 2024, pp. 38–55.

[83] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[84] 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, pp. 770–778.

[85] G. Wu *et al.*, "4d gaussian splatting for real-time dynamic scene rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 310–20 320.

[86] C. Zheng, L. Xue, J. Zarate, and J. Song, "Gstar: Gaussian surface tracking and reconstruction," *arXiv preprint arXiv:2501.10283*, 2025.

[87] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[88] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel, "Deep vit features as dense visual descriptors," *arXiv preprint arXiv:2112.05814*, vol. 2, no. 3, p. 4, 2021.

[89] D. Hadjivelichkov, S. Zwane, L. Agapito, M. P. Deisenroth, and D. Kanoulas, "One-shot transfer of affordance regions? affcorrs!" In *Conference on Robot Learning*, PMLR, 2023, pp. 550–560.

[90] J. Yu *et al.*, "Language-embedded gaussian splats (legs): Incrementally building room-scale representations with a mobile robot," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024, pp. 13 326–13 332.

[91] M. Savva *et al.*, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9339–9347.

[92] C. Zhang *et al.*, "Faster segment anything: Towards lightweight sam for mobile applications," *arXiv preprint arXiv:2306.14289*, 2023.

Supplementary Materials

A Frequently Asked Questions

A.1 Motivation and Alternative Approaches

Why should I care about policy mobilization? Why not simply train a mobile manipulation policy from a large dataset? Indeed, one approach to learning mobile manipulation is to train a policy from a dataset that includes both navigation and manipulation data. However, training such a policy typically requires large amounts of training data, since the policy needs to not only generalize to different navigation and manipulation scenarios but also seamlessly coordinate navigation and manipulation. We showed in our simulation experiments that the *BC w/ Nav* baseline fails to perform well in unseen room layouts despite learning from $5\times$ more training data (see Section 5.1). Compared to training an end-to-end mobile manipulation policy, our policy mobilization framework provides a more data-efficient approach to learning mobile manipulation.

Why not train a mobile manipulation policy in simulation and transfer to the real world? Training a mobile manipulation policy in simulation from 3DGS requires creating an interactive and physically accurate simulation from the 3DGS model, which is an unsolved problem.

A.2 Method

How sensitive are DINO dense descriptors to viewpoint changes? In the early stage of the project, we compared several feature representations including DINO dense descriptors, DINO features, policy encoder features, ResNet-50 [84], and DeepLab v3 [87]. We empirically found that the DINO dense descriptor performs the best in finding an in-distribution image out of a set of similar-looking images rendered from various viewpoints. In the paper, we showed quantitative ablation results where our method outperforms variations of our method where the DINO dense descriptors are replaced with flat DINO features and policy encoder features. These observations coincide with findings in prior work [88, 89] that DINO dense descriptors carry rich, well-localized semantic and positional information. Also, note that our method is compatible with any improved visual feature representation that emerges in the future.

How does the approach handle distractors? Our sim setup includes unseen test objects. We find that our method, which utilizes DINO dense descriptors to score robot poses, is capable of ignoring these irrelevant objects.

Can the method handle imperfect scene reconstruction? Yes. Our learned 3DGS models have artifacts and surface color inconsistencies (see Figure 11), yet the method still performs well.

Figure 11: **Sample 3D Gaussian Splatting renders.** As seen in the samples, our 3DGS models synthesizes novel views that are imperfect. Nevertheless, our method maintains good performance.

The proposed method builds a 3D Gaussian Splatting model. Does it scale to larger, more complex scenes? In simulation, we showed that our method effectively finds optimal base positions in kitchen scenes as large as $36m^2$. This proves that our method is useful in room-scale settings. To make our method useful in a multi-room setting, one can consider designing a system that first navigates to the correct room and then executes our method. In the real world, another bottleneck is building a high-quality 3D Gaussian Splatting model. Prior work [90] studied building 3DGS models for complex scenes by incrementally registering images captured using a multicamera system to minimize drift during scene mapping. Integrating methods like this into our framework is a clear direction for future work.

Is the visual feasibility metric general to the problem of policy mobilization? Our visual metric applies to methods that consider aligning the robot's viewpoint with the policy's capabilities during base placement. Naive baselines (e.g. LeLaN, VLFM) for policy mobilization are not policy-aware, so this metric does not apply.

A.3 Experiments

The proposed method requires access to training data. Are there ways to relax this assumption? We explained in the paper that, to appropriately link navigation and the execution of a manipulation policy, a policy mobilization method needs to be *policy-aware*. This means that the policy mobilization method needs to have some information about the manipulation policy. Our proposed method assumes to have access to the initial observations of the training demonstrations. This is usually easy to obtain in open-source datasets and models. We mentioned in the Limitations section that there are ways to relax this assumption if the dataset is too large or not publically available. For example, you could sample a subset of demonstration episodes or rolling out the policy to collect some successful episodes. Note that the baselines also assume access to in-distribution data to different extents: *BC w/ Nav* requires collecting additional in-distribution data for policy learning and *LeLaN* is fine-tuned with in-distribution navigation data.

The proposed method has access to the 3D model of the test-time scene. How did you ensure a fair comparison to baselines that natively do not assume this information? To ensure a fair comparison, we provide the *LeLaN* and the *VLFM* baselines with ground-truth 3D models of the scene for collision detection. This detail is also mentioned in Section 3.3 of the main paper. The *BC w/ Nav* cannot take into account a 3D model of the test-time scene. Therefore, we do not provide it.

B Method Details

B.1 Hyperparameters

When training Gaussian Splatting models, we use an image size of 512×512 in all simulated experiments and 1280×1024 in all real robot experiments. In simulation, since we know the ground truth camera poses without noise, we turn off the camera pose optimizer in `Nerfstudio`. In real robot experiments, since robot odometry drifts overtime, we turn on the camera pose optimizer. We observe that turning the camera pose optimizer on improves the quality of the trained Gaussian Splatting model in the real world.

During hybrid score computation, we use an image size of 224×224 to match the input dimensionalities of the DINO encoder. When computing K_{id}, we use $k = 5$ for the KNN model. When computing K_{obj}, we query the MiniCPM-v2 model with the image and a prompt that says "Is [object name] in the image? Answer exactly 'yes' or 'no'." We then return positively if the answer includes the substring "yes".

In all simulated experiments, we use $N_{\text{init}}^{(\text{BO})} = 2500$, $N_{\text{iter}}^{(\text{BO})} = 100$, and $N_{\text{batch}}^{(\text{BO})} = 5$. In real robot experiments, we use $N_{\text{init}}^{(\text{BO})} = 1000$. We use the UCB acquisition function with $\kappa = 1.96$ for simulated experiments and $\kappa = 0.5$ for real robot experiments.

B.2 Implementation Details

Our method takes as input one image at a time but can be trivially extended to a set of input images. This is useful when the robot is equipped with multiple cameras for example at the wrist or from an additional third-person viewpoint. To compute a predicted score $K(p)$ in this case, we evaluate the K_{id} and K_{obj} scores for each view, take the max, and then compose with the K_{col} score to obtain the final score.

In simulation experiments, for all tasks but the *Close Drawer* task, we only use one camera view to compute $K(p)$ since the right camera view is always the best camera view. However, in the *Close Drawer* task, we found the need to use both camera views since the drawer might be either on the left or right side of the robot, resulting in different views being optimal for deciding the $K(p)$ score for different scenes and setups.

We also add minor modifications to our method to deal with language-conditioned policies. When computing the score function $K(p)$ in one episode, one can simply filter the training images $\{o_1^k\}_{k=1}^{N_{DS}}$ to include only images that have matching language descriptions of the current task description. In this way, the resulting score will be computed corresponding to the task the agent is currently performing.

We run our method on a Linux workstation equipped with an NVIDIA RTX 4090 GPU. Training a 3D Gaussian Splatting model on this machine takes approximately 15 minutes. Running a round of sampling-based robot pose optimization with Bayesian Optimization takes approximately 6 minutes.

C Baseline Details

LeLaN. In our experiments, we assume that the robot initializes at a base position where it can navigate in a straight line to the target object without collision. Therefore, we used a pre-trained LeLaN [17] checkpoint without the collision loss objective. We tried the pre-trained checkpoint in simulation and found it to perform poorly. To improve the robustness of the baseline, we fine-tune the pre-trained checkpoint with a randomized navigation dataset. For this dataset, we collected 300 demonstrations in each training layout and task, totaling $300 \times 5 \times 5 = 7,500$ navigation demonstrations. Each demo has a different, procedurally generated scene texture. In each demonstration, we retrieve the ground-truth target object position, add an offset to this position so we find the nearest robot pose in front of the object without collision, and add random noise to the position and rotation of this pose. As a result, we obtain a dataset of non-policy-aware navigation data where the robot initializes in training scenes and navigates to objects that the robot will encounter at test time. We fine-tune LeLaN on this generated dataset for 20 episodes before deploying it into each test-time environment. We only fine-tune a single LeLaN checkpoint to be used for all tasks.

VLFM. We integrated VLFM [18] into our framework with several key enhancements. Images and segmentation masks from RoboCasa [19] were sent to VLFM, which generated high-level navigation commands such as `TURN LEFT`, `TURN RIGHT`, and `STOP`, based on its algorithm using pretrained feature extractors and object detectors to build semantic value maps for its exploration phase. Following exploration, we replaced VLFM's pretrained navigation module (originally trained in Habitat [91]) with our own implementation of RRT, utilizing a point cloud representation of the scene for collision checking. Rather than navigating to the closest point on the target object, we improved accuracy by selecting the median point within the object's point cloud. Additionally, we enhanced the performance of the pretrained MobileSAM [92] detector by incorporating RoboCasa's ground truth masks, which significantly improved segmentation accuracy.

BC w/ Nav. In simulation, we collect 1,500 demonstrations per task, where each demonstration includes navigation and manipulation chained together. This is $3\times$ more demos than the number of demos used for training the pure manipulation policy. Note that each demo also has a longer horizon than demos used to train a manipulation policy because of the additional navigation steps. In real-robot experiments, we collect the same number of episodes to train *BC w/ Nav* as our method. We use the same algorithm, hyperparameter setting, and number of epochs to train this baseline as the manipulation policy used for mobilization.

D Real Robot Experiment Details

Robot setup. In all real robot experiments, we use a custom mobile robot with a wheeled mobile base, a prismatic lift that can be used to adjust the height of the robot arms, and two Kinova Gen3 7DoF arms mounted on the left and right side of the robot. The robot also has two Basler cameras mounted inbetween the two Kinova arms. In total, the robot has 22 degrees of freedom (3 for the base, 1 for the lift, 7 for each arm, 1 for each gripper, and 2 for the head joints used for adjusting viewing angles of the Basler cameras). During policy execution, we only control the base and arms of the robot and keep head and lift joints fixed. In the *Pick Chips* and *Pour Tomatoes* tasks, only the left arm will have non-zero actions, though we keep the action space the same by zero-padding right arm actions.

Policy learning. The base policies in our real robot experiments take as input an image of the camera view and the robot end-effector poses, and outputs desired robot arm position and rotation velocities. To make a lightweight policy, we downsample the RGB images to 256×320 before passing them into the policy. Similar to the original Diffusion Policy paper, we use an observation horizon of 2 steps, a prediction horizon of 16 steps. Since there are latencies between observation retrieval and execution of the predicted actions, we perform action skipping so that the robot executes the actions that the policy expects it to run. After action skipping, we run all remaining predicted actions before

performing another round of action inference. We use the DDPM noise scheduler with 100 diffusion steps during training. We train all policies for 1,000 epochs before deploying them onto the robot.

Rollout. We develop a multiprocessed pipeline for evaluating trained Diffusion Policies [1] in real robot setups. An inference process repeatedly receives observations from the Basler camera and performs iterative denoising to predict a sequence of 16 actions at a time. The predicted actions and the corresponding observation timestamps are sent to the rollout process. At every step of policy execution, the rollout process receives an action sequence from the inference process and then skips a few actions so that the predicted actions sent to the robot are aligned in time with what the policy expects the robot to execute. During evaluation, we use the same DDPM noise scheduler as the training procedure.

Details of the *Humans* baseline. For each task, we ask 10 unique users to drive the mobile robot to where they believe is the optimal starting base pose to execute the task. All users have technical backgrounds in robotics but are not aware of how the policies are trained and what robot positions are best for policy execution. We then initialize the robot at each of these human-designated navigation targets, and then evaluate the manipulation policy from there. This allows us to test whether our method can even outperform navigation targets generated via human intuition.

Measuring task success. For each task and method, we track the incidence rates of certain performance milestones (i.e., navigation success, grasp success, full success), and also of task ending-collisions (i.e., fatal failures). We consider a navigation attempt successful if the robot drives to a pose that not only faces the target object but also is within a distance of 50 cm from the ideal target pose. A grasp success is defined to be when the robot stably grasps the object of interest (i.e. the chip bag, the paper towels, or the tray of tomatoes), and full success is counted only if the entire task is completed.

Detailed analysis of results. We find that for all three manipulation tasks, our method significantly outperforms the two baselines. The end-to-end *BC w/ Nav* baseline performs decently well in the *Pour Tomatoes* task, where the robot can navigate to a reasonable position 80% of the time, ultimately achieving a full success rate of 70% with only 20% of the runs ending in collision. In the *Bimanual Grab Paper Towels* and *Pick Chips* tasks however, it sees an uptick in fatal failures at 70% and 40% respectively. In our experiments, *BC w/ Nav* only achieves a 30% navigation success rate for both of these two tasks, which even further degrades into a 0% full success rate for the *Pick Chips* task.

The human baseline achieves a very high navigation success of at least 80% for all tasks. However, the success rate of the entire task is very low, with a maximum score of 40% in *Bimanual Grab Paper Towels*. In the *Pick Chips* task, 20% of the rollouts even end in arm-collisions with the shelf. This result yields an interesting insight—although humans are adept at intuitively navigating the robot into very reasonable starting poses, they are ultimately unaware of what the policy training data looks like. Therefore, the human-provided navigation poses still generally fail to achieve task success. Our method is able to pick up on the subtleties of these distribution shifts. In real experiments, our method achieves successful final navigation poses 100% of the time for all three tasks, and achieves full success rates of 70%, 100%, and 80% for the *Pick Chips*, *Bimanual Grab Paper Towels*, and *Pour Tomatoes* tasks respectively. Furthermore, unlike any of the evaluated baselines, the rollouts using our method are completely collision-free.

E Additional Visualizations

In addition to the visualizations shown in the main paper, we also design tools to visualize the pose optimization process. In Figure 12, we show top-down maps of the simulated benchmark environment overlaid with arrows and markers illustrating the policy mobilization process of our method.

First, we plot the oracle pose for executing the policy in dark red and the initial pose in dark green. Then, we plot the position and heading of robot poses sampled by our method, along with the predicted scores represented by color codes. Areas with collision check failures are marked pink; areas where the target object is not in view are marked black, and areas where K_{id} predicts a score are marked with colors in a rainbow color map that represent the predicted scores. The figure shows an example of our method sampling throughout the scene and converging to the optimal robot

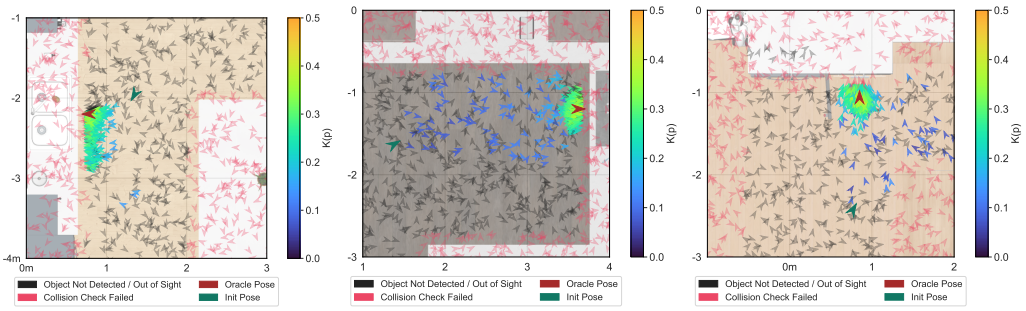

Figure 12: **Visualizations of policy mobilization process of our method.** Each plot shows one episode of robot pose optimization for policy mobilization. The background plots the top-down map of the environment where the robot needs to run the designated policy. Each arrow represents the position and heading of a robot pose sampled by our method. In the plot, our method converges toward the oracle policy initialization robot pose (in dark red).

pose for starting the execution of the manipulation policy. Such visualizations make debugging and developing policy mobilization methods easier with clear and precise visual feedback.