

# RAPIDLY ADAPTING POLICIES TO THE REAL WORLD VIA SIMULATION-GUIDED FINE-TUNING

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

Robot learning requires a considerable amount of high-quality data to realize the promise of generalization. This amount is challenging to collect entirely in the real world. Simulation can help greatly, wherein techniques such as reinforcement learning can provide broad coverage over states and actions. However, high-fidelity physics simulators are fundamentally misspecified approximations to reality, making direct zero-shot transfer challenging, especially in tasks where precise and force-sensitive manipulation is necessary. This makes real-world fine-tuning of policies pretrained in simulation an attractive approach to robot learning in principle, but for many practical applications exploring the real-world dynamics with standard RL fine-tuning techniques is too inefficient. This paper introduces Simulation-Guided Fine-Tuning, a general framework that leverages the structure of a value function learned in simulation to guide and accelerate real-world exploration. We demonstrate our approach across several real-world manipulation tasks for which learning successful policies using purely real-world data is very hard. Last but not least, we provide theoretical justification for this new paradigm.

## 1 INTRODUCTION

Robot learning offers a pathway to building robust, general-purpose robotic agents that can rapidly adapt their behavior to new environments and tasks. This shifts the burden from designing accurate environment models and task-specific controllers by hand to the problem of collecting large behavioral datasets with sufficient coverage. This raises a fundamental question: *how do we cheaply obtain and leverage such datasets at scale?* Real-world data collection via teleoperation (Walke et al., 2023; team, 2024) can generate high-quality trajectories but is tedious and scales linearly with human effort. Even with community-driven teleoperation (Mandlekar et al., 2018; Collaboration, 2024), current robotics datasets are still orders of magnitude smaller than those powering vision and language applications.

Massively parallelized physics simulation (Todorov et al., 2012; Makoviychuk et al., 2021) is a potential solution for cheaply generating vast quantities of synthetic robotics data. Moreover, applying techniques such as reinforcement learning (RL) in simulation can cheaply generate datasets with extensive *coverage*, when coupled with techniques such as automatic scene generation (Chen et al., 2024; Deitke et al., 2022) and extensive randomization of initial conditions and dynamics parameters (Peng et al., 2018; Andrychowicz et al., 2020).

Unfortunately, simulation-generated data is not a silver bullet, as it provides cheap but ultimately *off-domain data*. Namely, simulators are fundamentally *misspecified* approximations to reality. This is highlighted in tasks like hammering in a nail, where the modeling of high-impact, deformable contact remains an open problem (Acosta et al., 2022; Levy et al., 2024). In these regimes, *no choice of*

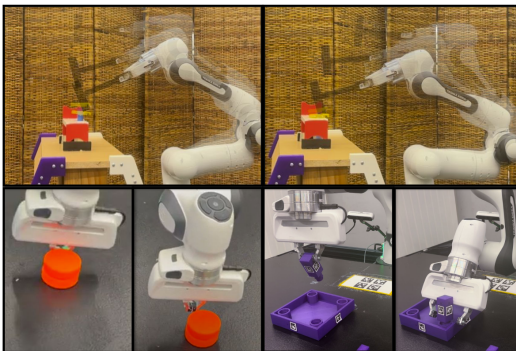


Figure 1: Three dynamic, contact-rich manipulation tasks – hammering (**top**), pushing (**bottom left**), and inserting (**bottom right**) – solved in the real world using our method *SGFT*.

054 *parameters for the physics simulator accurately capture the real-world dynamics.* This gap persists  
 055 despite efforts towards improving existing physics simulators with system identification (Memmel  
 056 et al., 2024; Huang et al., 2023) identification and learning generative world models directly from  
 057 large, but ultimately finite, real-world data sets Yang et al. (2023); Bruce et al.. Thus, despite  
 058 impressive performance for many tasks, methods that transfer policies from simulation to reality  
 059 zero-shot (Kumar et al., 2021; Lee et al., 2020; Peng et al., 2018; Andrychowicz et al., 2020) still  
 060 run into failure modes when they encounter situations outside simulated training distribution (Smith  
 061 et al., 2022b).

062 The question becomes: *can inaccurate simulation models be useful in the face of fundamental*  
 063 *misspecifications?* A natural technique is to fine-tune policies pre-trained in a simulator using real-  
 064 world experience (Smith et al., 2022b; Zhang et al., 2023). This approach can overcome misspeci-  
 065 fication by training directly on data from the target domain. However, existing approaches employ  
 066 the unstructured exploration strategies used by standard, general reinforcement learning algorithms  
 067 Haarnoja et al. (2018), which were designed not for fine-tuning but for *tabula rasa* learning. As a  
 068 result, current RL fine-tuning frameworks remain to sample inefficient for real-world deployment.

069 We argue the following: despite inherently getting the finer details wrong, generative physics  
 070 simulators capture the rough structure of real-world dynamics well enough to distill targeted ex-  
 071 ploration strategies which can substantially accelerate real-world learning. Leveraging this insight,  
 072 we propose *Simulation-Guided Fine-Tuning (SGFT)*, a general framework for rapid adaptation of  
 073 behaviors learned in simulation to the nuances of real-world dynamics. The key ingredient of *SGFT*  
 074 is to use a value function  $V_{sim}$  learned in the approximate simulator to guide exploration in the real-  
 075 world. We demonstrate this general framework can be built on top of generic reinforcement learning  
 algorithms, substantially accelerating real-world learning.

076 In more detail, unlike most fine-tuning strategies which aim to solve an intractable infinite-  
 077 horizon objective, *SGFT* solves a short-horizon  $H$ -step policy optimization problem with a re-  
 078 shaped objective. Specifically, we use  $V_{sim}$  to provide guidance to base policy optimization strate-  
 079 gies by leveraging the Potential-based Reward-Shaping Formalism Ng et al. (1999). This horizon-  
 080 shortening approach effectively uses  $V_{sim}$  to boot-strap long-horizon returns, leading to a tractable  
 081 search problem Westenbroek et al. (2022); Cheng et al. (2021). This paradigm is particularly power-  
 082 ful when built on top of a base model-based reinforcement learning (MBRL) algorithm Hafner et al.  
 083 (2019); Janner et al. (2019); Hansen et al. (2024). The central challenge for these methods (when  
 084 solving standard long-horizon objectives) is that small errors in the model rapidly compound when  
 085 unrolling predictions over multiple steps Janner et al. (2019). The short-horizon *SGFT* effectively  
 086 leverages experience gained in simulation to circumvent this challenge, unlocking the potential of  
 generative environment modeling.

087 We outline our contributions as follows:

- 088
- 089 1. We introduce the *SGFT* framework, and illustrate through extensive experiments how it is  
 090 a light-weight modification to existing sim-to-real RL finetuning frameworks Smith et al.  
 091 (2022b); Zhang et al. (2019); indeed, because these approaches generally employ algo-  
 092 rithms which already learn a value function  $V_{sim}$  in simulation, implementing the reshaped  
 093 *SGFT* objective is only a few lines of code.
- 094
- 095 2. We implement two versions of *SGFT* on top of different base MBRL algorithms, and  
 096 demonstrate that *SGFT* can learn highly effective policies in regimes where existing zero-  
 097 shot transfer techniques fail. Moreover, *SGFT* provides substantial sample complexity  
 098 gains over existing fine-tuning method. In particular, *SGFT* provides a powerful framework  
 099 for learning policy with near 100% success rates, which we observe to be a significant  
 100 challenge for existing fine-tuning approaches.
- 101
- 102 3. We demonstrate theoretically how *a) SGFT* can learn highly performant policies, despite  
 103 the bias introduced by the short-horizon objective and *b) SGFT* renders standard MBRL  
 104 approaches robust to large errors in generative dynamics models, which are bound to arise  
 105 in the fine-tuning setting where only a small amount of on-task data is available. Together,  
 106 these insights underscore how *SGFT* effectively leverages plentiful off-domain data from  
 107 a simulator alongside small real-world data sets to substantially accelerate real-world fine-  
 tuning.

## 2 RELATED WORK

**Simulation-to-Reality Transfer.** In this work, we assume that perception in simulation and reality is approximately matched and focus primarily on the dynamics gap. To bridge this dynamics gap, two classes of methods have been popular: 1) adapting simulation parameters to real-world data and 2) learning adaptive or robust policies to account for changing real-world dynamics. While going back from the real world to simulation can help target the simulation parameters more accurately (Chebotar et al., 2019; Ramos et al., 2019; Memmel et al., 2024), it cannot overcome inherent model misspecification, as we show in our experimental evaluation. Learning adaptive policies to account for changing real-world dynamics (Qi et al., 2022; Kumar et al., 2021; Yu et al., 2017) can help to some extent but is unable to guide exploration and adapt beyond the training dynamics range. Another appealing approach is to search for distributions of domain randomization parameters that will lead to maximally robust transfer in a principled fashion. For example, Tiboni et al. (2023a) uses a small pre-collected data set to estimate distributions over dynamics parameters, rather than collapsing to a point estimate as in pure system identification. This is taken further by Tiboni et al. (2023b), which uses an entropy maximization to strategy to automatically evolve training distributions purely in simulation to generate a curricula of environments which leads to policies which are both performant and robust.

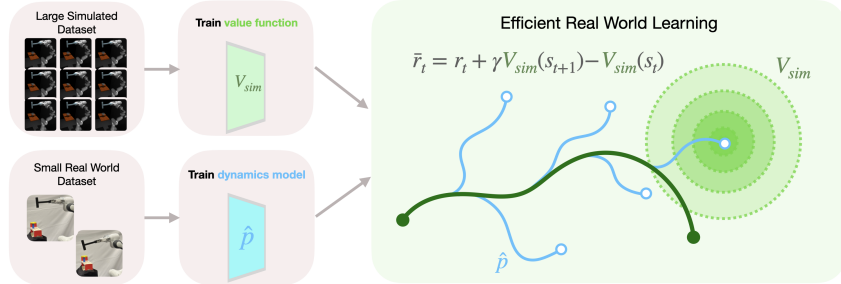
Perhaps most related is Smith et al. (2022b); Zhang et al. (2023), where policies learned in simulation are fine-tuned with off-policy RL. However, besides initialization, simulation is not used to guide exploration throughout fine-tuning. Prior work has additionally considered mixing simulated and real data during policy optimization – either through co-training (Torne et al., 2024), simply initializing the replay-buffer with simulation data (Smith et al., 2022a; Ball et al., 2023), or adaptively sampling the simulated dataset and up-weighting transitions that approximately match the true dynamics (Eysenbach et al., 2020; Liu et al., 2022; Xu et al., 2023; Niu et al., 2022). In contrast, our approach focuses on distilling useful *exploration strategies from simulation*.

**Fine-tuning in Reinforcement Learning.** Our work is related to algorithms for RL-based fine-tuning with online data collection, primarily starting from offline RL-aided or imitation learning-aided initializations. These algorithms typically aim to provide an initialization that can continue improving with standard off-policy RL (Rajeswaran et al., 2018; Nair et al., 2020; Kostrikov et al., 2021; Hu et al., 2023; Nakamoto et al., 2024). They initialize policies and Q-functions from offline data and continue training them with standard RL methods, but do not use the pre-training data beyond initialization and populating the replay buffer. We utilize the pretraining in simulation not just for initialization but also to provide guidance throughout the policy improvement process.

**Reward Design in Reinforcement Learning.** A significant component of our methodology is learning dense shaped reward in simulation to guide real-world fine-tuning. This is closely tied to the problem of reward design and reward inference in RL (Gupta et al., 2022). This is a challenging problem if attempted tabula rasa but prior techniques have tried to infer rewards from expert demos (Ziebart et al., 2008; Ho & Ermon, 2016), success examples (Fu et al., 2018; Li et al., 2021a), LLMs (Ma et al., 2023; Yu et al., 2023), and heuristics (Margolis et al., 2024; Berner et al., 2019). We rely on simulation to provide reward supervision using the PBRs formalism (Ng et al., 1999), and shorten the horizon of the learning task to improve the sample complexity of real-world learning (Cheng et al., 2021; Westenbroek et al., 2022). Another line of work complementary to our comes from Eysenbach et al. (2020); Liu et al. (2022); Xu et al. (2023); Niu et al. (2022), which relabel rewards from off-task (simulated) data, effectively up-weighting transitions that approximately match the dynamics observed in the target domain. These works focus on the *retrieval* of useful samples from prior datasets with shifted dynamics. In contrast, our approach uses prior data to guide the discovery of novel sequences of states and actions in the target domain. In principle, these techniques could be used in conjunction; we leave this to future work.

**Model-Based Reinforcement Learning.** A significant body of work on model-based RL learns dynamics models to perform data augmentation (Sutton, 1991; Wang & Ba, 2019; Janner et al., 2019; Yu et al., 2020; Kidambi et al., 2020) for downstream policy optimization algorithms, plan online using the model (Ebert et al., 2018; Zhang et al., 2019), or to use the model as a control variate to reduce variance of policy gradient methods (Chebotar et al., 2017; Cheng et al., 2019; 2020; Che et al., 2018). The central challenge for each of these model-based methods is that small inaccuracies in predictive models can quickly compound over time, leading to large *model-bias*. An effective critic can be used to shorten search horizons (Hansen et al., 2024; Bhardwaj et al., 2020; Hafner et al., 2019) yielding easier decision-making problems, but learning such a critic from scratch can

162 still require large amounts of on-task data. We demonstrate that, for many real-world continuous  
 163 control problems, critics learned entirely in simulation can be robustly transferred to the real-world  
 164 and substantially accelerate model-based learning.  
 165



175 Figure 2: Depiction of *SGFT*: Rather than simply fine-tuning from a policy initialization, *SGFT* uses short  
 176 horizon rollouts from the learned model along with potential-based reward shaping using the learned value  
 177 function to inform efficient sim-to-real fine-tuning.  
 178

### 180 3 PRELIMINARIES

181 Let  $\mathcal{S}$  and  $\mathcal{A}$  be state and action spaces. Our goal is to control a real-world system defined by  
 182 an unknown Markovian dynamics  $s' \sim p_{real}(\cdot|s, a)$ , where  $s, s' \in \mathcal{S}$  are states and  $a \in \mathcal{A}$  is an  
 183 action. The usual formalism for solving tasks with RL is to define a Markov Decision Process of the  
 184 form  $\mathcal{M}_r = (\mathcal{S}, \mathcal{A}, p_{real}, \rho_{real}^0, r, \gamma)$  with initial real-world state distribution  $\rho_{real}^0$ , reward function  
 185  $r$ , and discount factor  $\gamma \in [0, 1)$ . Given a policy  $\pi$ , we let  $d_{real}^\pi(s)$  denote the distribution over  
 186 trajectories  $\tau = (s_0, a_0, s_1, a_1, \dots)$  generated by applying  $\pi$  starting at initial state  $s_0$ . Defining the  
 187 value function under  $\pi$  as  $V_{real}^\pi(s) = \mathbb{E}_{s_t \sim d_{real}^\pi(s)}[\sum_t \gamma^t r_t(s_t)]$ , our objective is to find  $\pi_{real}^* \leftarrow$   
 188  $\sup_\pi \mathbb{E}_{s \sim \rho_{real}^0}[V_{real}^\pi(s)]$ . We define the optimal value function as  $V_{real}^*(s) := \sup_\pi V_{real}^\pi(s)$ .  
 189

190 Unfortunately, as discussed above, obtaining a good approximation to  $\pi_{real}^*$  using only real-  
 191 world data is often impractical. Thus, many approaches leverage an approximate simulation environ-  
 192 ment  $s' \sim p_{sim}(s, a)$  and solve an approximate MDP of the form  $\mathcal{M}_{sim} := (\mathcal{S}, \mathcal{A}, p_{sim}, \rho_{sim}^0, r, \gamma)$   
 193 to train a policy  $\pi_{sim}$  meant to approximate  $\pi_{real}^*$ . We let  $V_{sim}$  denote  $\pi_{sim}$ 's value function with  
 194 respect to  $\mathcal{M}_{sim}$ . We do not require  $\pi_{sim}$  to be optimal in simulation, but merely an approximately  
 195 optimal one that can reliably solve the given task under  $p_{sim}$ . Here,  $\rho_{sim}^0$  is the distribution over  
 196 initial conditions in the simulator.  
 197

### 198 4 SIMULATION-GUIDED FINE-TUNING

199 Simulation-Guided Fine-Tuning (*SGFT*) takes the perspective that  $\pi_{sim}$  can be viewed as an ap-  
 200 proximate expert for controlling  $\mathcal{M}_{real}$ . That is,  $\pi_{sim}$  can capture the rough structure of behaviors  
 201 that solve the task in reality, even if it fails to transfer to  $\mathcal{M}_{real}$  zero-shot. Specifically, we contend  
 202 that these behaviors are encoded into  $V_{sim}$  in a way that is robust to dynamics shifts. Indeed,  $V_{sim}$   
 203 defines an ordering over states that are desirable to reach in the future, *but says nothing about the*  
 204 *actions needed to reach those states*. *SGFT* uses small amounts of real-world data to learn actions  
 205 that will increase  $V_{sim}$  over short horizons, and thus leverages  $V_{sim}$  to guide real-world exploration  
 206 towards behaviors that can solve the entire (longer-horizon) task.  
 207

#### 208 4.1 REWARD SHAPING AND HORIZON SHORTENING

209 We implement our horizon-shortening approach using the Potential-Based Reward Shaping (PRBS)  
 210 formalism (Ng et al., 1999), which replaces the reward  $r(s)$  with  $\bar{r}(s, s') = r(s) + \gamma\Phi(s') - \Phi(s)$   
 211 for some function  $\Phi$  and is usually applied to infinite horizon MDPs  $\mathcal{M}$  to form a new MDP  $\bar{\mathcal{M}}$ .  
 212 Intuitively, the addition of the potential term encourages policy search algorithms to follow  $\Phi$  by  
 213 increasing its value during each transition. A well-designed  $\Phi$  can guide exploration and decrease  
 214 the variance of policy search by effectively acting as a *baseline*, but the reshaped reward does not  
 215 change the optimal policy. Indeed, by telescoping out the rewards one may observe that  $\sum_{t=0}^\infty \bar{r}_t =$   
 $\sum_{t=0}^\infty r_t - \Phi(s_0)$ . Because  $\Phi(s_0)$  does not depend on the choice of the policy, the reshaped reward  
 does not affect the ordering over policies. However, even with an informative reshaped reward,

solving the infinite horizon reshaped MDP is still costly. This motivates us to shorten the horizon of the returns that we optimize in the real-world, and thus we will investigate optimizing  $H$ -step returns of the form  $\sum_{t=0}^{H-1} \bar{r}_t = \sum_{t=0}^{H-1} r_t + \gamma^H \Phi(s_H) - \Phi(s_0)$ . Here, the  $\gamma^H \Phi(s_H)$  term can be interpreted as a fixed approximation to the true long-horizon returns  $\gamma^H V_{real}^*(s_H)$ , while  $-\Phi(s_0)$  again acts as a baseline to reduce variance. While this introduces bias into the returns, optimizing the  $H$ -step return from a given initial condition presents a significantly more tractable problem.

#### 4.2 $H$ -STEP SIMULATION-GUIDED EXPERT POLICIES

We propose to set  $\Phi(s) = V_{sim}(s)$  (i.e. the value of the policy  $\pi_{sim}$  with respect to  $\mathcal{M}_{sim}$ ) and optimize the reshaped reward  $\bar{r}(s, s') = r(s) + \gamma V_{sim}(s') - V_{sim}(s)$  over  $H$ -steps from every initial condition  $s \in \mathcal{S}$ , as a mean to adapt  $\pi_{sim}$  from  $\mathcal{M}_{sim}$  to  $\mathcal{M}_{real}$ . We will use ‘tilde’ notation  $\tilde{\pi}_H = \{\pi_{H,0}, \pi_{H,1}, \dots, \pi_{H,H-1}\}$  to denote non-stationary policies of horizon  $H$ , where  $\pi_{H,t}$  is the policy applied at time  $t$ . Consider the following  $H$ -step returns under the real-world dynamics:

$$V_H^{\tilde{\pi}_H}(s) = \mathbb{E} \left[ \gamma^H V_{sim}(s_H) + \sum_{t=0}^{H-1} \gamma^t r(s_t) - V_{sim}(s_0) \mid s_0 = s, a_t \sim \pi_{H,t}(\cdot | s_t) \right]. \quad (1)$$

$$V_H^*(s) := \sup_{\tilde{\pi}_H} V_H^{\tilde{\pi}_H}(s) \quad Q_H^*(s, a) = \mathbb{E}_{s' \sim p_{real}(s, a)} [\gamma V_{H-1}^*(s') + \bar{r}(s, s')] \quad (2)$$

Note that the  $-V_{sim}(s_0)$  term in Equation (3) is not affected by the choice of  $\tilde{\pi}_H$ , and does not affect the ordering of policies. Thus,  $V_H^{\tilde{\pi}_H}$  is equivalent to the planning objective used by model predictive control (MPC) methods (Jadbabaie et al., 2001; Hansen et al., 2024; Sun et al., 2018; Bhardwaj et al., 2020) with  $H$ -step look-ahead and a terminal reward of  $V_{sim}$  (when the ground truth dynamics is known). Thus, we define the  $H$ -step simulation-guided MPC expert via  $\pi_H^*(\cdot | s) = \pi_{H,0}^*(\cdot | s)$ , where  $\pi_0^*$  is taken from the first step of the solution to  $\tilde{\pi}_H^* \leftarrow \max_{\tilde{\pi}_H} V_H^{\tilde{\pi}_H}(s)$ . In other words, this policy simply applies the first action generated by the optimal  $H$ -step policy at each state. Abusing notation, note that this policy can also be calculated as  $\pi_H^*(\cdot | s) \leftarrow \max_{\pi} Q_H^*(s, \pi(s))$ .

Intuitively,  $\pi_H^*$  will greedily follow  $V_{sim}$  at every state when  $H = 1$ , and as we take  $H \rightarrow \infty$  the behavior of  $\pi_H$  will recover the behavior of  $\pi_{real}^*$ . Thus,  $\pi_H^*$  can be viewed as a policy that has adapted the behavior of  $\pi_{sim}$  to follow  $V_{sim}$  along the real-world dynamics, and for smaller values of  $H$  we should expect  $\pi_H^*$  to retain more of the behavior of  $\pi_{sim}$ . As we discuss further in Section 5, these idealized MPC policies have a number of desirable properties, when compared to directly transferring  $\pi_{sim}$  to  $\mathcal{M}_{real}$ . Of course, because these expert policies depend on the real-world dynamics, we do not have direct access to their actions. Next, We introduce a general framework that implicitly attempts to mimic these experts through interactions with the real-world dynamics.

#### 4.3 THE GENERAL SGFT FRAMEWORK

Even though we do not have direct access to  $\pi_H^*$ , we can implicitly learn its actions by optimizing policies to maximize the  $H$ -step return (Equation (3)) starting from every initial condition  $s \in \mathcal{S}$ . Specifically, note that the expert actions are given by  $\pi_H^* \leftarrow \max_{\pi} Q_H^*(s, \pi)$ , and that many standard (model-based and model-free) RL methods can be leveraged to solve this short-horizon policy optimization problem. This motivates our conceptual *Simulation-Guided Fine-Tuning (SGFT)* framework, which is defined via the pseudo-code in Algorithm 1. *SGFT* fine-tunes  $\pi_{sim}$  to succeed under the real-world dynamics by iteratively 1) unrolling the current policy to correct transitions from  $p_{real}$  and 2) using the current dataset  $\mathcal{D}$  of transitions to approximately optimize

---

##### Algorithm 1 Simulation-Guided Fine-tuning (SGFT)

---

**Require:** Pretrained policy  $\pi_{sim}$  and value function  $V_{sim}$

- 1:  $\pi \leftarrow \pi_{sim}, \mathcal{D} \leftarrow \emptyset$
  - 2: **for** each iteration  $k$  **do**
  - 3:   **for** time step  $t = 1, \dots, T$  **do**
  - 4:      $a_t \sim \pi(\cdot | s_t)$
  - 5:     Observe the state  $s_{t+1}$  and the reward  $r_t$ .
  - 6:      $\bar{r}_t \leftarrow r_t + \gamma V_{sim}(s_{t+1}) - V_{sim}(s_t)$
  - 7:      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, \bar{r}_t, s_{t+1})\}$
  - 8:   **end for**
  - 9:   Approx. optimize  $\pi \leftarrow \max_{\pi} Q_H^*(s, \pi(s_j))$  using transitions in  $\mathcal{D} \forall s_j$  in  $\mathcal{D}$ ’s transitions.
  - 10: **end for**
-

$\pi \leftarrow \max_{\pi} Q_H^*(s, \pi(s_j))$  at each state  $s_j$  the agent has visited. In Sun et al. (2018) a model-free method for approximately achieving the optimization in step 2) is proposed<sup>1</sup>, and we discuss how this step can be performed with model-based methods below. By approximately optimizing  $\pi \leftarrow \max_{\pi} Q_H^*(s, \pi(s_j))$ , *SGFT* is implicitly attempting to learn and approximate the actions taken  $\pi_H^*$  at every state the agent has visited (which is reminiscent of the learning loop used by DAgger (Ross et al., 2011), with the obvious caveat that in our case the expert is not directly available).

**What are we taking away from simulation?** We are explicitly interested in applying *SGFT* with small values of  $H$ , as this will yield real-world learning problems that are significantly more tractable than the original infinite-horizon problem (Laidlaw et al., 2023; Li et al., 2021b). In this regime, *SGFT* provides an ideal separation between what is learned using the simulated dynamics  $p_{sim}$  and what is learned from interacting with real dynamics  $p_{real}$ . For small values of  $H$ , optimizing (3) guides policy optimization algorithms towards policies that increase the value of  $V_{sim}$  over  $H$ -step windows. In the extreme case where  $H = 1$ , *SGFT* greedily attempts to increase the value of  $V_{sim}$  each step. Conceptually, in this special case we have reduced the policy search problem to a contextual bandits problem, which is much easier to solve than the original infinite horizon objective. Intuitively, this is because the critic is frozen and no bootstrapping in the real-world is required. More generally, as  $H$  becomes larger the dependence on  $V_{sim}$  is lessened while the dependence on the real returns increase. In simulation, we can easily generate enough data to explore many paths through the state space and discover which motions lead to higher returns (e.g., reach towards an object to be manipulated). This information is distilled into  $V_{sim}$  during the learning process, which provides information about *where* the agent should go  $H$  steps into the future. On the other hand, by optimizing Equation (3) over short horizons, we only need to learn short sequences of actions to move the system to states where  $V_{sim}$  is higher. That is, the short-horizon problem makes it easy to learn *how to get to these desirable states*, even if the required sequences of intervening states and actions differ substantially under  $p_{sim}$  and  $p_{real}$ . Thus, *SGFT* approximately decomposes into a) learning *where to go* in the simulator and b) learning the finer details of *how to get there* with small amounts of real-world data.

#### 4.4 LEVERAGING SHORT MODEL ROLL-OUTS

Model-based reinforcement learning (MBRL) holds the promise of learning a generative model  $\hat{p}$  to rapidly identify effective policies with significantly less data than model-free methods Janner et al. (2019). In particular, generating rollouts with the model enables an agent to reason about trajectories not contained in the dataset of transitions the agent has observed. However, as discussed in Section 2, the central challenge for MBRL is that small errors in  $\hat{p}$  can quickly compound over multiple steps, degrading the quality of the predictions. As a consequence, learning a model accurate enough to solve long-horizon problems can often take as much data as solving the task with modern model-free methods (Chen et al., 2021; Hiraoka et al., 2021). By bootstrapping  $V_{sim}$  in simulation, where data is plentiful, the *SGFT* framework enables agents to act effectively over long horizons using only short, local predictions about the real-world dynamics. As our experiments demonstrate, this substantially improves performance compared to model-free approaches.

In what follows, we denote the following returns under the model for  $\tilde{\pi}_H = \{\pi_{H,t}\}_{t=0}^{H-1}$ :

$$\hat{V}_H^{\tilde{\pi}_H}(s) = \mathbb{E} \left[ \gamma^H V_{sim}(s_H) + \sum_{t=0}^{H-1} \gamma^t r(s_t) - V_{sim}(s_0) \mid s_0 = s, a_t \sim \pi_{H,t}(\cdot | s_t), s_{t+1} \sim \hat{p}(s_t, a_t) \right]. \quad (3)$$

$$\hat{V}_H^*(s) := \sup_{\tilde{\pi}_H} \hat{V}_H^{\tilde{\pi}_H}(s) \quad \hat{Q}_H^*(s, a) = \mathbb{E}_{s' \sim \hat{p}(s, a)} \left[ \gamma \hat{V}_{H-1}^*(s') + \bar{r}(s, s') \right] \quad (4)$$

The corresponding MPC that uses  $\hat{p}$  is given by:

$$\hat{\pi}_H^*(\cdot | s) \leftarrow \arg \max_{\pi} \hat{Q}_H^*(s, \pi). \quad (5)$$

Next, we discuss two broad approaches that use  $\hat{p}$  to approximately learn  $\hat{\pi}_H^*$  in each iteration.

<sup>1</sup>In particular, in Sun et al. (2018) (where the authors consider general potentials  $\Phi$  in the  $H$ -step returns), instead of optimizing the non-stationary objective Equation (3) directly, the  $H$ -step return  $\hat{V}_H^{\pi}(s) = \mathbb{E}_{d^{\pi}(s)} \left[ \gamma^H \Phi(s_H) + \sum_{t=0}^{H-1} \gamma^t r(s_t) - \Phi(s_0) \right]$  for the current (stationary) policy is used as a stand-in. This approach uses Monte-Carlo returns to estimate the advantage  $\bar{A}_H^{\pi}(s, a) = \bar{Q}_H^{\pi}(s, a) - V(s, a)$  for  $\pi$  and update  $\pi \leftarrow \max_{\pi} \bar{A}_H^{\pi}(s, \pi)$ . Repeating this optimization fits policies which optimize  $\pi \leftarrow \max_{\pi} Q_H^*(s, \pi(s))$ .

### Improved Sample Efficiency with Data Augmentation (Algorithm 2).

The generative model  $\hat{p}$  can be used for *data augmentation* by generating a dataset of synthetic rollouts  $\hat{\mathcal{D}}$  to supplement the real-world dataset  $\mathcal{D}$  (Janner et al., 2019; Sutton, 1990; Gu et al., 2016). The combined dataset can then be fed to any policy optimization strategy, such as generic model-free algorithms. We are specifically interested in state-of-the-art Dyna-style algorithms (Janner et al., 2019), which, in our context, branch  $H$ -step rollouts from states the agent has visited previously. As Algorithm 2 shows, after each data-collection phase, this approach updates the generative model  $\hat{p}$  and then repeatedly *a)* generates a dataset  $\hat{\mathcal{D}}$  of synthetic  $H$ -step rollouts under the current policy  $\pi$  starting from states in  $\mathcal{D}$ , and *b)* approximately solves  $\pi \leftarrow \max_{\pi} Q_H^*(s, \pi(s))$  at the ob-

served real-world states using the augmented dataset  $\hat{\mathcal{D}} \cup \mathcal{D}$  and a base model-free method. For example, the method from Sun et al. (2018) can be used for this purpose. In Section 6, we describe how to implement this approach with 1-step hallucinated trajectories and SAC (Haarnoja et al., 2018) as a base model-free algorithm.

**Online Planning (Algorithm 3).** The most straightforward way to approximate the behavior of  $\pi_H^*$  is simply to apply the MPC controller  $\hat{\pi}_H^*$  generated using the current best guess for the dynamics  $\hat{p}$ , as in Equation (5). Algorithm 3 provides general pseudocode for this approach, which iteratively 1) rolls out  $\hat{\pi}_H^*$  (which is calculated using online optimization and  $\hat{p}$  (Williams et al., 2017)) then 2) updates the model on the current dataset of transitions  $\mathcal{D}$ . This high-level approach encompasses a wide array of methods proposed in the literature, e.g., Ebert et al. (2018) and Zhang et al. (2019). In Section 6, we implement this approach using the TDMPC-2 (Hansen et al., 2024).

## 5 THEORETICAL ANALYSIS

In this section we analyze the effectiveness of the  $H$ -step simulation-guided expert  $\pi_H^*$ . Specifically, we seek suboptimality bounds for this agent and a characterization of situations when the behavior of this idealized policy will be robust to the errors made by MBRL techniques, which leverage an approximate model  $\hat{p}$  for  $p_{real}$  that has been learned from real-world data. We are particularly interested in understanding how *SGFT* with short prediction horizons  $H$  can mitigate errors in  $\hat{p}$  with a small number of samples. To set the stage for this analysis, we first recall several relevant theoretical results from prior RL work.

**Proposition 1.** *Suppose that  $\max_{s \in \mathcal{S}} |V_{sim}(s) - V_{real}^*(s)| \leq \epsilon$  and that  $\|\hat{p}(s, a) - p_{real}(s, a)\| \leq \alpha$ . Then for  $H$  sufficiently small and for each  $s \in \mathcal{S}$  we have:*

$$V_{real}^*(s) - V_{real}^{\hat{\pi}_H^*}(s) \leq O\left(\frac{\gamma}{1-\gamma}\alpha H + \frac{\gamma^H}{1-\gamma^H}\epsilon\right), \quad (6)$$

where  $\hat{\pi}_H^*$  is the MPC policy under the approximate mode as in Equation (5)

This result is a direct translation of (Bhardwaj et al., 2020, Theorem 3.1) to our setting, where the big- $O$  notation suppresses problem-dependent constants and lower-order terms for small values of  $H$  which are not important for our discussion. To understand the bound, first set  $\alpha = 0$  so that there is no modeling error (and we recover the behavior of  $\pi_H^*$ ). In this case, we are incentivized to make  $H$  larger. Intuitively, this follows from the fact that  $V_{sim}(s_H)$  can be viewed as an approximation to  $V_{real}^*(s_H)$  in the  $H$ -step look ahead objective Equation (3). Because this approximation is scaled by

---

### Algorithm 2 Dyna-SGFT

---

**Require:** Pretrained policy  $\pi_{sim}$  and value function  $V_{sim}$

- 1:  $\pi \leftarrow \pi_{sim}$
- 2: **for** each iteration  $k$  **do**
- 3:   Generate rollout  $\{(s_t, a_t, r_t, s_{t+1})\}_{t=0}^T$  under  $\pi$ .
- 4:    $\bar{r}_t \leftarrow r_t + \gamma V_{sim}(s_{t+1}) - V_{sim}(s_t)$
- 5:    $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, \bar{r}_t, s_{t+1})$
- 6:   Fit generative model  $\hat{p}$  with  $\mathcal{D}$ .
- 7:   **for**  $G$  policy updates **do**
- 8:     Generate synthetic branched rollouts  $\hat{\mathcal{D}}$  under  $\pi$ .
- 9:     Approx. optimize  $\pi \leftarrow \max_{\pi} Q_H^*(s, \pi(s_j))$   
 $\quad \forall s_j \in \mathcal{D}$  using augmented dataset  $\hat{\mathcal{D}} \cup \mathcal{D}$
- 10:   **end for**
- 11: **end for**

---



---

### Algorithm 3 MPC-SGFT

---

**Require:** Pretrained value  $V_{sim}$  and initialized model  $\hat{p}$ .

- 1: **for** each iteration  $k$  **do**
- 2:   Generate rollout  $\{(s_t, a_t, r_t, s_{t+1})\}_{t=0}^T$  under  $\hat{\pi}_H^*$ .
- 3:    $\bar{r}_t \leftarrow r_t + \gamma V_{sim}(s_{t+1}) - V_{sim}(s_t)$
- 4:    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, \bar{r}_t, s_{t+1})\}$ .
- 5:   Fit generative model  $\hat{p}$  with  $\mathcal{D}$ .
- 6: **end for**

---

$\gamma^H$ , the effect of errors in  $V_{sim}$  is diminished for larger values of  $H$ . Note, however, that this result scales poorly for small values of  $H$ . This is especially true for long-horizon problems where  $\gamma \approx 1$ . On the other hand, the term involving  $\alpha$  captures how errors in the model accumulate for different horizons  $H$ , leading to mistakes in decision making. This term incentivizes us to make  $H$  as small as possible. In particular, we are interested in understanding how we can mitigate large values of  $\alpha$ , which will arise in low-data regimes.

Since results like Proposition 1 simply assume uniform worst-cases bounds on the difference between between the magnitudes of  $V_{sim}$  and  $V_{real}^*$ , they do not capture the fact that  $V_{sim}$  may still preserve an ordering over states that is useful for guiding real-world decision making, i.e., when the *geometry* of  $V_{sim}$  enables *SGFT* to guide policy search algorithms towards effective policies under the real dynamics. We use the following definition from Cheng et al. (2021), which is similar to properties from other works (Westenbroek et al., 2022; Grune & Rantzer, 2008):

**Definition 1.** We say that  $V_{sim}$  is improvable with respect to  $\mathcal{M}_{real}$  if for each  $s \in \mathcal{S}$  we have:

$$\max_a \mathbb{E}_{s' \sim p_{real}(s,a)} [\gamma V_{sim}(s')] - V_{sim}(s) \geq -r(s). \quad (7)$$

Namely,  $V_{sim}$  is improvable with respect to  $\mathcal{M}_{real}$  if there exists a policy that can increase  $V_{sim}$  enough over time for each state (with respect to the reward function). A quick intuition we make precise later is the following: as long as  $V_{sim}$  reaches a maximum at desirable states in the real world (e.g., at desired positions for an object being manipulated), then if  $V_{sim}$  is improvable with respect to  $\mathcal{M}_{real}$  we can greedily follow  $V_{sim}$  over a short horizon to reach these desirable states. As defined in Section 3,  $V_{sim}$  is learned under the simulation dynamics and policy  $\pi_{sim}$  such that  $V_{sim}(s) = \mathbb{E}[\gamma V_{sim}(s') + r(s) | s' \sim p_{sim}(s, a), a \sim \pi_{sim}(\cdot | s)]$ , and thus is constructed to be improvable with respect to  $\mathcal{M}_{sim}$ . We provide a pedagogical example in Section F highlighting why this is a plausible property to assume for real-world transfer.

We now present our main theoretical result:

**Theorem 1.** Let the Assumptions of Proposition 1 hold. Further, suppose that  $V_{sim}$  is improvable with respect to  $\mathcal{M}_{real}$ . Then for  $H$  sufficiently small and each  $s \in \mathcal{S}$  we have:

$$V_{real}^*(s) - V_{real}^{\hat{\pi}_H^*}(s) \leq O\left(\frac{\gamma}{1-\gamma} \alpha H + \gamma^H \epsilon\right), \quad (8)$$

where  $\hat{\pi}_H^*$  is the MPC policy under the approximate model, as in Equation (5).

The proof can be found in Appendix A. At a high-level, the proof uses arguments similar to Grune & Rantzer (2008) to bound the suboptimality of the expert policy  $\pi_H^*$  and then combines this bound with the perturbation bounds from Bhardwaj et al. (2020) to bound how errors in the dynamics lead to additional suboptimality. Note that the dependence on  $H$  and  $\alpha$  is identical to the bound from Proposition 1 above. However, the scaling for the term involving  $\epsilon$  is improved substantially for small values of  $H$ , especially for long-horizon problems where  $\gamma \approx 1$ . Thus, we can more readily use small values of  $H$  to combat large model bias when  $V_{sim}$  is improvable with respect to  $\mathcal{M}_{real}$ . This provides insight into how *SGFT* can rapidly learn effective policy in the real world by using short model rollouts with a coarse model to approximate the behaviors of  $\hat{\pi}_H^*$ .

## 6 EXPERIMENTS

We aim to answer the following questions: **(1)** Can *SGFT* facilitate online fine-tuning of policies for dynamic, real-world robotic manipulation tasks using only small amounts of real-world data? **(2)** Does *SGFT* improve the sample efficiency of online fine-tuning compared benchmarks? Can *SGFT* learn policies that outperform direct transfer techniques which leverage extensive domain randomization and/or system identification?

### 6.1 METHODS EVALUATED

Our experiments compare several approaches from three families:

**SGFT Instantiations.** We implement concrete instantiations of the general *Dyna-SGFT* and *MPC-SGFT* frameworks sketched in Algorithms 2 and 3. **SGFT-SAC** fits a model to real world transitions to perform data augmentation and uses SAC as a base model-free policy optimization algorithm. Specifically, after collecting each real-world trajectory for  $G$  gradient steps we 1) generate



a dataset  $\hat{\mathcal{D}}$  of synthetic rollouts branched from the real dataset  $\mathcal{D}$ , and then 2) sample from the joint replay buffer  $\hat{\mathcal{D}} \cup \mathcal{D}$  to perform one policy updates with SAC using the reshaped reward. We use  $H = 1$  in all our hardware experiments. To implement our approach, we set the ‘done’ flag to true at the end of each roll-out. In the case where  $H = 1$ , this means applying ‘dones’ each transition. As discussed above, this is equivalent to reducing the problem to a contextual bandits problem, which is a much easier problem to solve than the original infinite horizon objective. In effect, we freeze the critic learned in simulation, only updating the policy to rapidly adapt to dynamics shifts.

**SGFT-TDMPC-2** uses TDMPC-2 Hansen et al. (2024) as a backbone. The base method learns a critic, a policy, and an approximate dynamics model through interactions with the environment. When acting in the world and MPC controller solves online planning problems using the approximate model, the critic as a terminal reward, and uses the policy prior to seed an MPPI planner Williams et al. (2017). To integrate this method with SGFT, when transferring to the real world we simply freeze the critic learned in simulation and use the reshaped objective in Equation (3) as the online planning objective. For our experiments, we use  $H = 4$  and the default hyperparameters reported in Hansen et al. (2024).

**Baseline Fine-tuning Methods.** The SAC baseline fine-tunes the pre-trained policy to solve the original MDP  $\mathcal{M}_{real}$  using SAC Haarnoja et al. (2018) – it does not use shaping or horizon shortening. **PBRS** fine-tunes the policy under a reshaped infinite-horizon MDP using the reshaped reward  $\bar{r}$  and SAC (Haarnoja et al., 2018) as the policy optimizer, but does not leverage horizon shortening. **TDMPC-2** fine-tunes the entire TDMPC-2 architecture Hansen et al. (2024) in the real world, but does not leverage reward shaping. *This serves as a state-of-the-art baseline for MBRL.* **IQL** fine-tunes the pre-trained policy to solve the original MDP  $\mathcal{M}_{real}$  using IQL (Kostrikov et al., 2021). *This serves as a state-of-the-art baseline for fine-tuning methods.*

**Baseline Sim-to-Real Methods.** Our **Domain Randomization** baseline refers to policies trained with extensive domain randomization in simulation and transferred directly to the real world. These policies rely only on the previous observation. **Recurrent Policy + Domain Randomization** uses policies conditioned on histories of observations, similar to methods such as Kumar et al. (2021). The history enables the agent to infer information about the dynamics parameters of the environment it is operating in. **ASID** (Mommel et al., 2024) is a system identification method that performs targeted exploration in the real world to identify the dynamics parameters of the simulator that best match the real-world scene. Once the parameters are identified, a policy is trained under the parameters in simulation and then deployed zero-shot to the real world.

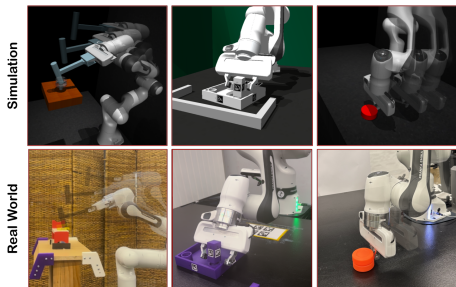


Figure 3: **Sim-to-Real Setup** Simulation setup for pretraining (top) and execution of real-world fine-tuning (bottom) of real-world hammering (left), insertion (middle), and pushing (right).

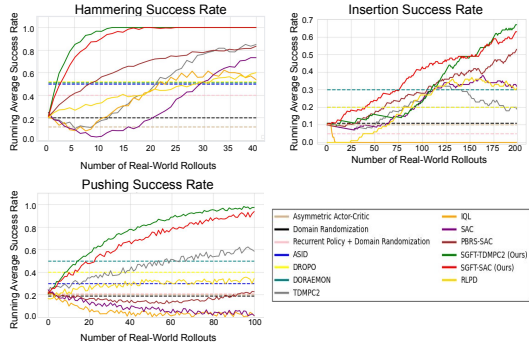


Figure 4: **Real-world success rates during the course of online fine-tuning.** We plot task success rates over number of fine-tuning rollouts for the tasks described in Sec. 6. We see that *SGFT* yields significant improvements in success and efficiency.

## 6.2 SIM-TO-REAL EVALUATIONS

We test each methods on three real-world manipulation tasks illustrated in Figure 3, demonstrating that both the *SGFT-SAC* and *SGFT-TDMPC-2* instantiations of *SGFT* excel at learning policies with minimal real-world data.

**Hammering** is a highly dynamic task involving force and contact dynamics that are impractical to precisely model in simulation. In our setting, the robot is tasked with hammering a nail in a board. The nail has high, variable dry friction along its shaft. In order to hammer the nail into the board, the robot must hit the nail with high force repeatedly. The dynamics are inherently misspecified between simulation and reality here due to the infeasibility of accurately modeling the properties of the nail and its contact interaction with the hammer and board.

**Insertion** (Heo et al., 2023) involves the robot grasping a table leg and accurately inserting it into a table hole. The contact dynamics between the leg and the table differ between simulation and real-world conditions. In the simulation, the robot successfully completes the task by wiggling the leg into the hole, but in the real world this precise motion becomes challenging due to inherent noise in the real-world observations as well as contact discrepancies between the leg and the table hole.

**Pushing** requires pushing a puck of unknown mass and friction forward to the edge of the table without it falling off the edge. Here, the underlying feedback controller of the real world robot inherently behaves differently from simulation. Additionally, retrieving and processing sensor information from cameras incurs variable amounts of latency. As a result, the controller executes each commanded action for variable amounts of time. These factors all contribute to the sim-to-real dynamics shift, requiring real-world fine-tuning to reconcile.

Each of these tasks is evaluated on a physical setup using a Franka FR3 robot operating with either Cartesian position control or joint position control at 5Hz. We compute object positions by color-thresholding pointclouds or by Aruco marker tracking, although this approach could easily be upgraded. Further details of all tasks, reward functions, robot setups, environments and implementation details can be found in the Appendix.

### 6.3 ANALYSIS

The results of the real-world evaluation during fine-tuning on these three tasks are presented in Figure 4. For all three tasks, zero-shot performance seen at the start of the plot is quite poor due to the dynamics sim-to-real gap. Moreover, the poor performance of system identification methods such as ASID highlights the fact that these gaps are due to more than parameter misidentification, but rather stem from fundamental misspecification.

The second class of comparison methods includes offline pretraining with online fine-tuning techniques like IQL Kostrikov et al. (2021) and SAC Haarnoja et al. (2018). Both flavors of *SGFT*, the model-free *SGFT-SAC* and model-based *SGFT-TDMPC-2*, substantially outperform these prior techniques in terms of efficiency and asymptotic performance. This suggests that simulation can offer more guidance during real-world policy search than just an initialization for subsequent fine-tuning. Our full system consistently leads to significant improvement from fine-tuning, achieving 100% success for hammering and pushing within an hour of fine-tuning and 70% success for inserting within two hours of fine-tuning. The fact that *SGFT* outperforms both TD-MPC2 Hansen et al. (2024) and PBRs-SAC, suggests that efficient fine-tuning requires a *combination* of both short model rollouts and value-driven reward shaping.

Last but not least, note that *SGFT* offers improvements on top of both SAC and TDMPC2, showing the generality of the proposed paradigm. Additional evaluations and visualizations are in the Appendix, namely a set of sim-to-sim transfer results following standard benchmarks (Appendix E), and visualizations of transferred value functions (Appendix D).

## 7 LIMITATIONS AND FUTURE WORK

In this work, we present *SGFT*, a technique for efficient sim-to-real fine-tuning using off-policy RL. The key idea in *SGFT* is to leverage learned value functions and models from simulation to provide guidance for exploration even when simulation does not perfectly match reality through a combination of short-horizon model hallucinations and potential-based reward shaping. There are several limitations of *SGFT* that open avenues for improvement. Firstly, scaling *SGFT* to work from raw perceptual inputs such as camera images rather than low-dimensional states would make this paradigm more broadly applicable. Secondly, it is important to scale *SGFT* to higher-dimensional action spaces and longer-horizon tasks. Thirdly, our choice of off-policy RL method can display a degree of instability and a more efficient and stable base algorithm should be considered.

## REFERENCES

- 540  
541  
542 Brian Acosta, William Yang, and Michael Posa. Validating robotics simulators on real-world im-  
543 pacts. *IEEE Robotics and Automation Letters*, 7(3):6471–6478, 2022.
- 544 OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew,  
545 Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning  
546 dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20,  
547 2020.
- 548 Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learn-  
549 ing with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR,  
550 2023.
- 551  
552 Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy  
553 Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz,  
554 Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto,  
555 Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya  
556 Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement  
557 learning. *CoRR*, abs/1912.06680, 2019. URL <http://arxiv.org/abs/1912.06680>.
- 558 Mohak Bhardwaj, Sanjiban Choudhury, and Byron Boots. Blending mpc & value function approxi-  
559 mation for efficient reinforcement learning. *arXiv preprint arXiv:2012.05909*, 2020.
- 560  
561 Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes,  
562 Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative inter-  
563 active environments. In *Forty-first International Conference on Machine Learning*.
- 564 Tong Che, Yuchen Lu, George Tucker, Surya Bhupatiraju, Shane Gu, Sergey Levine, and Yoshua  
565 Bengio. Combining model-based and model-free rl via multi-step control variates. 2018.
- 566  
567 Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey  
568 Levine. Combining model-based and model-free updates for trajectory-centric reinforcement  
569 learning. In *International conference on machine learning*, pp. 703–711. PMLR, 2017.
- 570 Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan D. Ratliff,  
571 and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world  
572 experience. In *ICRA*, 2019.
- 573  
574 Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning:  
575 Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.
- 576  
577 Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan  
578 Wu, Dieter Fox, and Abhishek Gupta. Urdformer: A pipeline for constructing articulated simula-  
579 tion environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.
- 580  
581 Ching-An Cheng, Xinyan Yan, Nathan Ratliff, and Byron Boots. Predictor-corrector policy opti-  
582 mization. In *International Conference on Machine Learning*, pp. 1151–1161. PMLR, 2019.
- 583  
584 Ching-An Cheng, Xinyan Yan, and Byron Boots. Trajectory-wise control variates for variance  
585 reduction in policy gradient methods. In *Conference on Robot Learning*, pp. 1379–1394. PMLR,  
586 2020.
- 587  
588 Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Heuristic-guided reinforcement learn-  
589 ing. *Advances in Neural Information Processing Systems*, 34:13550–13563, 2021.
- 590  
591 Open-X-Embodiment Collaboration. Open x-embodiment: Robotic learning datasets and rt-x mod-  
592 els. In *ICRA*, 2024.
- 593  
594 Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador,  
595 Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR:  
596 Large-scale embodied AI using procedural generation. In *Advances in Neural Infor-*  
597 *mation Processing Systems 35: Annual Conference on Neural Information Processing*  
598 *Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9,*

- 594 2022, 2022. URL [http://papers.nips.cc/paper\\_files/paper/2022/hash/](http://papers.nips.cc/paper_files/paper/2022/hash/27c546able4f1d7d638e6a8dfbad9a07-Abstract-Conference.html)  
595 [27c546able4f1d7d638e6a8dfbad9a07-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2022/hash/27c546able4f1d7d638e6a8dfbad9a07-Abstract-Conference.html).  
596
- 597 Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-  
598 real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp.  
599 1290–1296. IEEE, 2021.
- 600 Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual fore-  
601 sight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint*  
602 *arXiv:1812.00568*, 2018.  
603
- 604 Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdi-  
605 nov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv*  
606 *preprint arXiv:2006.13916*, 2020.
- 607 Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with  
608 events: A general framework for data-driven reward definition. *NeurIPS*, 2018.  
609
- 610 Lars Grune and Anders Rantzer. On the infinite horizon performance of receding horizon controllers.  
611 *IEEE Transactions on Automatic Control*, 53(9):2100–2111, 2008.  
612
- 613 Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning  
614 with model-based acceleration. In *International conference on machine learning*, pp. 2829–2838.  
615 PMLR, 2016.
- 616 Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham M. Kakade, and Sergey Levine. Unpacking  
617 reward shaping: Understanding the benefits of reward engineering on sample complexity. In  
618 *NeurIPS*, 2022. URL [http://papers.nips.cc/paper\\_files/paper/2022/hash/](http://papers.nips.cc/paper_files/paper/2022/hash/6255f22349da5f2126dfc0b007075450-Abstract-Conference.html)  
619 [6255f22349da5f2126dfc0b007075450-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2022/hash/6255f22349da5f2126dfc0b007075450-Abstract-Conference.html).
- 620 Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy  
621 maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018. URL  
622 <http://proceedings.mlr.press/v80/haarnoja18b.html>.  
623
- 624 Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James  
625 Davidson. Learning latent dynamics for planning from pixels. In *International conference on*  
626 *machine learning*, pp. 2555–2565. PMLR, 2019.
- 627 Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for contin-  
628 uous control. In *The Twelfth International Conference on Learning Representations*, 2024. URL  
629 <https://openreview.net/forum?id=Oxh5CstDJU>.  
630
- 631 Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible  
632 real-world benchmark for long-horizon complex manipulation, 2023. URL [https://arxiv.](https://arxiv.org/abs/2305.12821)  
633 [org/abs/2305.12821](https://arxiv.org/abs/2305.12821).
- 634 Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsu-  
635 ruoka. Dropout q-functions for doubly efficient reinforcement learning. *arXiv preprint*  
636 *arXiv:2110.02034*, 2021.  
637
- 638 Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural*  
639 *information processing systems*, 29, 2016.  
640
- 641 Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learn-  
642 ing. *arXiv preprint arXiv:2311.02198*, 2023.
- 643 Peide Huang, Xilun Zhang, Ziang Cao, Shiqi Liu, Mengdi Xu, Wenhao Ding, Jonathan Francis,  
644 Bingqing Chen, and Ding Zhao. What went wrong? closing the sim-to-real gap via differentiable  
645 causal discovery. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Conference on Robot*  
646 *Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of*  
647 *Machine Learning Research*, pp. 734–760. PMLR, 2023. URL [https://proceedings.](https://proceedings.mlr.press/v229/huang23c.html)  
[mlr.press/v229/huang23c.html](https://proceedings.mlr.press/v229/huang23c.html).

- 648 Ali Jadbabaie, Jie Yu, , and John Hauser. Unconstrained receding-horizon control of nonlinear  
649 systems. *IEEE Transactions on Automatic Control*, 46(5):776–783, 2001.
- 650
- 651 Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-  
652 based policy optimization. *Advances in neural information processing systems*, 32, 2019.
- 653
- 654 Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-  
655 based offline reinforcement learning. *Advances in neural information processing systems*, 33:  
656 21810–21823, 2020.
- 657 Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-  
658 learning. In *International Conference on Learning Representations*, 2021.
- 659
- 660 Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: rapid motor adaptation for  
661 legged robots. In *RSS*, 2021.
- 662
- 663 Cassidy Laidlaw, Stuart J. Russell, and Anca Dragan. Bridging rl theory and practice with the  
664 effective horizon. In *Advances in Neural Information Processing Systems*, volume 36, pp. 58953–  
665 59007, 2023.
- 666
- 667 Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning  
668 quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
- 669
- 670 Jacob Levy, Tyler Westenbroek, and David Fridovich-Keil. Learning to walk from three minutes of  
671 data with semi-structured dynamics models. In *8th Annual Conference on Robot Learning*, 2024.  
672 URL <https://openreview.net/forum?id=evCXw1CMIi>.
- 673
- 674 Kevin Li, Abhishek Gupta, Ashwin Reddy, Vitchyr H Pong, Aurick Zhou, Justin Yu, and Sergey  
675 Levine. Mural: Meta-learning uncertainty-aware rewards for outcome-driven reinforcement  
676 learning. In *ICML*, 2021a.
- 677
- 678 Yuanzhi Li, Ruosong Wang, and Lin F. Yang. Settling the horizon-dependence of sample com-  
679 plexity in reinforcement learning. In *62nd IEEE Annual Symposium on Foundations of Com-  
680 puter Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pp. 965–976. IEEE, 2021b.  
681 doi: 10.1109/FOCS52979.2021.00097. URL [https://doi.org/10.1109/FOCS52979.  
682 2021.00097](https://doi.org/10.1109/FOCS52979.2021.00097).
- 683
- 684 Jinxin Liu, Hongyin Zhang, and Donglin Wang. Dara: Dynamics-aware reward augmentation in  
685 offline reinforcement learning. *arXiv preprint arXiv:2203.06662*, 2022.
- 686
- 687 Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayara-  
688 man, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via  
689 coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.
- 690
- 691 Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin,  
692 David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High  
693 performance gpu-based physics simulation for robot learning. In *NeurIPS-2021 Datasets and  
694 Benchmarks Track*, 2021.
- 695
- 696 Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao,  
697 John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic  
698 skill learning through imitation. In *Conference on Robot Learning*, 2018.
- 699
- 700 Gabriel B. Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion  
701 via reinforcement learning. *Int. J. Robotics Res.*, 43(4):572–587, 2024. doi: 10.1177/  
02783649231224053. URL <https://doi.org/10.1177/02783649231224053>.
- 702
- 703 Marius Memmel, Andrew Wagenmaker, Chuning Zhu, Patrick Yin, Dieter Fox, and Abhishek  
704 Gupta. ASID: active exploration for system identification in robotic manipulation. *CoRR*,  
705 abs/2404.12308, 2024.
- 706
- 707 Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online rein-  
708 forcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

- 702 Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral  
703 Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-  
704 tuning. *Advances in Neural Information Processing Systems*, 36, 2024.  
705
- 706 Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations:  
707 Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999.  
708
- 709 Haoyi Niu, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming Hu, Xianyuan Zhan, et al. When to trust  
710 your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. *Advances in  
711 Neural Information Processing Systems*, 35:36599–36612, 2022.
- 712 Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of  
713 robotic control with dynamics randomization. In *2018 IEEE international conference on robotics  
714 and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.
- 715 Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation  
716 via rapid motor adaptation. In *CoRL*, 2022. URL [https://proceedings.mlr.press/  
717 v205/qi23a.html](https://proceedings.mlr.press/v205/qi23a.html).  
718
- 719 Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel  
720 Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforce-  
721 ment Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*,  
722 2018.  
723
- 724 Fabio Ramos, Rafael Possas, and Dieter Fox. Bayessim: Adaptive domain randomization via prob-  
725 abilistic inference for robotics simulators. In *RSS*, 2019.
- 726 Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and struc-  
727 tured prediction to no-regret online learning. In *Proceedings of the fourteenth international con-  
728 ference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference  
729 Proceedings, 2011.  
730
- 731 Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes  
732 with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022a.
- 733 Laura M. Smith, J. Chase Kew, Xue Bin Peng, Sehoon Ha, Jie Tan, and Sergey Levine. Legged  
734 robots that keep on learning: Fine-tuning locomotion policies in the real world. In *ICRA*, 2022b.  
735
- 736 Wen Sun, J Andrew Bagnell, and Byron Boots. Truncated horizon policy search: Combining rein-  
737 forcement learning & imitation learning. *arXiv preprint arXiv:1805.11240*, 2018.  
738
- 739 Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approxi-  
740 mating dynamic programming. In *Machine learning proceedings 1990*, pp. 216–224. Elsevier,  
741 1990.
- 742 Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART  
743 Bull.*, 2(4):160–163, jul 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL [https:  
744 //doi.org/10.1145/122344.122377](https://doi.org/10.1145/122344.122377).  
745
- 746 DROID Collaboration team. Droid: A large-scale in-the-wild robot manipulation dataset. In  
747 *Robotics Science and Systems (RSS)*, 2024.
- 748 Gabriele Tiboni, Karol Arndt, and Ville Kyrki. Dropo: Sim-to-real transfer with offline domain  
749 randomization. *Robotics and Autonomous Systems*, 166:104432, 2023a.
- 750 Gabriele Tiboni, Pascal Klink, Jan Peters, Tatiana Tommasi, Carlo D’Eramo, and Georgia Chal-  
751 vatzaki. Domain randomization via entropy maximization. *arXiv preprint arXiv:2311.01885*,  
752 2023b.  
753
- 754 Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control.  
755 In *IROS*, 2012.

- 756 Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit  
757 Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust ma-  
758 nipulation. *CoRR*, abs/2403.03949, 2024. doi: 10.48550/ARXIV.2403.03949. URL <https://doi.org/10.48550/arXiv.2403.03949>.  
759
- 760 Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao,  
761 Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and  
762 Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot*  
763 *Learning (CoRL)*, 2023.  
764
- 765 Jin Wang, Qilong Xue, Lixin Li, Baolin Liu, Leilei Huang, and Yang Chen. Dynamic analysis  
766 of simple pendulum model under variable damping. *Alexandria Engineering Journal*, 61(12):  
767 10563–10575, 2022.
- 768 Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *arXiv preprint*  
769 *arXiv:1906.08649*, 2019.  
770
- 771 Tyler Westenbroek, Fernando Castaneda, Ayush Agrawal, Shankar Sastry, and Koushil Sreenath.  
772 Lyapunov design for robust and efficient robotic reinforcement learning. *arXiv preprint*  
773 *arXiv:2208.06721*, 2022.
- 774 Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral  
775 control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40  
776 (2):344–357, 2017.  
777
- 778 Kang Xu, Chenjia Bai, Xiaoteng Ma, Dong Wang, Bin Zhao, Zhen Wang, Xuelong Li, and Wei Li.  
779 Cross-domain policy adaptation via value-guided data filtering. *Advances in Neural Information*  
780 *Processing Systems*, 36:73395–73421, 2023.
- 781 Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter  
782 Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.  
783
- 784 Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn,  
785 and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information*  
786 *Processing Systems*, 33:14129–14142, 2020.
- 787 Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning  
788 a universal policy with online system identification. In *RSS*, 2017. URL [http://www.  
789 roboticsproceedings.org/rss13/p48.html](http://www.roboticsproceedings.org/rss13/p48.html).
- 790 Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Are-  
791 nas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to  
792 rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.  
793
- 794 Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine.  
795 Solar: Deep structured representations for model-based reinforcement learning. In *International*  
796 *conference on machine learning*, pp. 7444–7453. PMLR, 2019.
- 797 Yunchu Zhang, Liyiming Ke, Abhay Deshpande, Abhishek Gupta, and Siddhartha S. Srinivasa.  
798 Cherry-picking with reinforcement learning. In *RSS*, 2023.  
799
- 800 Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse  
801 reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence*  
802 *- Volume 3, AAAI’08*, pp. 1433–1438. AAAI Press, 2008. ISBN 9781577353683.  
803  
804  
805  
806  
807  
808  
809

## 810 A PROOFS

811 We first present several Lemma’s used in the proof of 1.

812 **Lemma 1.** (Bhardwaj et al., 2020, Lemma A.1.) Suppose that  $\|\hat{p}(s, a) - p_{real}(s, a)\|_1 \leq \alpha$ . Further  
813 suppose  $\Delta r = \max_{sim} r(s) - \min_{sim} r(s)$  and  $\Delta V = \max_{sim} V_{sim}(s) - \min_{sim} V_{sim}(s)$  are finite.  
814 Then, for each policy  $\tilde{\pi}$  we may bound the  $H$ -step returns under the model and true dynamics by:

$$815 \|\hat{V}_H^{\tilde{\pi}}(s) - V_H^{\tilde{\pi}}(s)\|_\infty \leq \gamma \left( \frac{1 - \gamma^{H-1}}{1 - \gamma} \frac{\Delta r}{2} + \gamma^H \frac{\Delta V}{2} \right) \cdot \alpha H. \quad (9)$$

816 *Proof.* This result follows imediatly from the proof of (Bhardwaj et al., 2020, Lemma A.1.), with  
817 changes to notation and noting that we assume access to the true reward.  $\square$

818 **Lemma 2.** Suppose that  $\|\hat{p}(s, a) - p_{real}(s, a)\|_1 \leq \alpha$ . Further suppose  $\Delta r = \max_{sim} r(s) -$   
819  $\min_{sim} r(s)$  and  $\Delta V = \max_{sim} V_{sim}(s) - \min_{sim} V_{sim}(s)$  are finite. Then for each state  $s \in \mathcal{S}$   
820 we have:

$$821 V_H^{\hat{\pi}_H^*}(s) - V_H^*(s) \leq \left( \frac{1 - \gamma^{H-1}}{1 - \gamma} \Delta r + \gamma^H \Delta V \right) \quad (10)$$

822 where  $\hat{\pi}_H^* \leftarrow \max_{\tilde{\pi}_H} \hat{V}_H^{\tilde{\pi}_H}(s)$ .

823 *Proof.* Let  $\tilde{\pi}_H^* \leftarrow \max_{\tilde{\pi}_H} V_H^{\tilde{\pi}_H}(s)$  be the optimal policy under the true dynamics. By Lemma 1 we  
824 have both that

$$825 V_H^*(s) \leq \hat{V}_H^{\tilde{\pi}_H^*}(s) + \gamma \left( \frac{1 - \gamma^{H-1}}{1 - \gamma} \frac{\Delta r}{2} + \gamma^H \frac{\Delta V}{2} \right) \cdot \alpha H. \quad (11)$$

$$826 \hat{V}_H^{\hat{\pi}_H^*}(s) \leq V_H^{\hat{\pi}_H^*}(s) + \gamma \left( \frac{1 - \gamma^{H-1}}{1 - \gamma} \frac{\Delta r}{2} + \gamma^H \frac{\Delta V}{2} \right) \cdot \alpha H. \quad (12)$$

827 Combining these two bounds with the fact that  $\hat{V}_H^{\hat{\pi}_H^*}(s) \leq V_H^{\hat{\pi}_H^*}(s)$  yields the desired result.  $\square$

828 **Lemma 3.** Suppose that  $\sup_a \mathbb{E}_{s \sim p_{real}(s, a)} [\gamma V_{sim}(s')] - V_{sim}(s) > -r(s)$ . Then we have  $V_H^*(s) \geq$   
829  $V_{H-1}^*(s)$  for each  $s \in \mathcal{S}$ . Then for each  $s \in \mathcal{S}$  we have:

$$830 V_H^*(s) \geq V_{H-1}^*(s) \quad (13)$$

831 *Proof.* Fix an initial condition  $s_0 \in \mathcal{S}$ . Let  $\pi$  be arbitrary, and fix the shorthand  $\pi^* =$   
832  $\{\pi_0^*, \dots, \pi_{H-1}^*\}$  for the time-varying policy  $\pi^* \leftarrow \max_{\tilde{\pi}} V_H^{\tilde{\pi}}(s_0)$ . Then, concatenate these poli-  
833 cies to define:  $\tilde{\pi} = \{\pi_1^*, \dots, \pi_{H-2}^*, \pi\}$ , which is simply the result of applying the optimal policy for  
834 the  $(H - 1)$ -step look ahead objective Equation (3) starting from  $s_0$ , followed by applying  $\pi$  for a  
835 single step. Letting the following distributions over trajectories be generated by  $\pi^*$ , by the definition  
836 of  $V_H^*$ :

$$837 \begin{aligned} & V_H^*(s_0) \\ & \geq \mathbb{E} \left[ \gamma^H V_{sim}(s_H) + \sum_{t=1}^{H-1} \gamma^t r(s_t) - V_{sim}(s_0) \right] \\ & = \mathbb{E} \left[ \gamma^H V_{sim}(s_H) - \gamma^{H-1} V_{sim}(s_{H-1}) + \gamma^H r(s_{H-1}) \right] \\ & \quad + \mathbb{E} \left[ \gamma^{H-1} V_{sim}(s_{H-1}) + \sum_{t=1}^{H-2} \gamma^t r(s_t) - V_{sim}(s_0) \right] \\ & = \mathbb{E} \left[ \gamma^H V_{sim}(s_H) - \gamma^{H-1} V_{sim}(s_{H-1}) + \gamma^H r(s_{H-1}) + V_{H-1}^*(s_0) \right] \end{aligned}$$

838 Now, since our choice of  $\pi$  used to define  $\tilde{\pi}$  was arbitrary, we choose  $\pi$  to be deterministic and  
839 such that  $\mathbb{E}_{s' \sim p_{real}(s, a)} [\gamma V_{sim}(s')] - V_{sim}(s) > -r(s)$  at each state  $s \in \mathcal{S}$ , as guaranteed by the  
840 assumption made for the result. This choice of policy gantees that:

$$841 \mathbb{E} \left[ \gamma^H V_{sim}(s_H) - \gamma^{H-1} V_{sim}(s_{H-1}) + \gamma^H r(s_{H-1}) \right] \geq 0. \quad (14)$$

842 The desired result follows immediately by combining the two preceding bounds, and noting that  
843 our choice of initial condition was arbitrary, meaning the preceding analysis holds for all initial  
844 conditions.  $\square$



**Lemma 4.** Suppose that  $V_{sim}$  is improvable and further suppose that  $\max_{s \in \mathcal{S}} |V_{sim}(s) - V_{real}^*(s)| < \epsilon$ . Then any policy  $\pi$  which satisfies  $A_H^*(s, \pi) = Q_H^*(s, \pi) - V_H^*(s) \geq -\delta$  will satisfy:

$$V_{real}^*(s) - V_{real}^\pi(s) \leq \gamma^H \epsilon + \frac{\delta}{1 - \gamma}. \quad (15)$$

*Proof.* Our goal is first to bound how  $Q_H^*(s, \pi)$  changes on expectation when applying the given policy for a single step. We have that:

$$Q_H^*(s, \pi) + \delta \geq V_H^*(s) \quad (16)$$

$$V_H^*(s) \geq V_{H-1}^*(s) \quad (17)$$

$$Q_H^*(s, \pi) = \mathbb{E}[\gamma V_{H-1}^*(s') + \bar{r}(s, s')] \quad (18)$$

where the first inequality follows from the Assumption of the theorem, the second inequality follows from Lemma 3 and is simply the definition of  $Q_H^*$ . Letting  $s' \sim p_{real}(s, a)$  with  $a \sim \pi(\cdot|s)$ , we can take expectations can combine the previous relation to obtain:

$$\gamma \mathbb{E}[Q_H^*(s', \pi) + \bar{r}(s, s')] + \gamma \delta \geq \gamma \mathbb{E}[V_H^*(s') + \bar{r}(s, s')] + \geq \mathbb{E}[\gamma V_{H-1}^*(s') + \bar{r}(s, s')] = Q_H^*(s, \pi). \quad (19)$$

That is:

$$\gamma \mathbb{E}[Q_H^*(s', \pi)] + \bar{r}(s) + \gamma \delta \geq Q_H^*(s, \pi). \quad (20)$$

Alternatively:

$$\bar{r}(s) \geq Q_H^*(s, \pi) - \gamma \mathbb{E}[Q_H^*(s', \pi)] - \gamma \delta. \quad (21)$$

Next, we use this bound to provide a lower bound for  $V_{real}^\pi(s)$ . Because the previous analysis holds at all states when we apply  $\pi$ , the following holds over the distribution of trajectories generated by applying  $\pi$  starting from the initial condition  $s_0$ :

$$\begin{aligned} \mathbb{E}_{\rho_{\pi}^{\pi}(s)} \left[ \sum_{t=0}^{\infty} \gamma^t \bar{r}(s_t) \right] &= V_{real}^\pi(s) - V_s(s_0) \\ &\geq \mathbb{E}_{\rho_{\pi}^{\pi}(s)} \left[ \sum_{t=0}^{\infty} \gamma^t \left( Q_H^*(s_t, \pi) - \gamma Q_H^*(s_{t+1}, \pi) \right) \right] - \gamma \delta \sum_{t=0}^{\infty} \gamma^t \\ &= Q_H^*(s_0, \pi) - \frac{\gamma \delta}{1 - \gamma}, \end{aligned}$$

where we have repeatedly telescoped out sums to cancel out terms.

Thus, we have the lower-bound:

$$V_{real}^\pi(s) \geq Q_H^*(s, \pi) + V_{sim}(s_0) - \frac{\gamma \delta}{1 - \gamma} \quad (22)$$

Next, we may bound:

$$\begin{aligned} V_H^*(s_0) + V_{sim}(s_0) &\geq \mathbb{E}_{\rho_{\pi}^{\pi}(s_0)} \left[ \gamma^H V_{sim}(s_H) + \sum_{t=0}^{H-1} \gamma^t r(s_t) \right] \\ &= \mathbb{E}_{\rho_{\pi}^{\pi}(s_0)} \left[ \gamma^H V_{sim}(s_H) - \gamma^H V_{real}^*(s_H) + \gamma^H V_{real}^*(s_H) + \sum_{t=0}^{H-1} \gamma^t r(s_t) \right] \\ &= \mathbb{E}_{\rho_{\pi}^{\pi}(s_0)} [\gamma^H V_{sim}(s_H) - \gamma^H V_{real}^*(s_H)] + V_{real}^*(s_0). \end{aligned} \quad (23)$$

Invoking the assumption that  $\max_s |V_{sim}(s) - V_{real}^*(s)| < \epsilon$ , we can combined this with the preceding bound to yield:

$$V_H^*(s_0) + V_{sim}(s_0) \geq V_{real}^*(s) - \gamma^H \epsilon. \quad (24)$$

Finally, once more invoking the fact that  $Q_H^*(s, \pi) + \delta \geq V_H^*(s)$  for each  $s \in S$  and combining this with Equation (22) and Equation (23), we obtain that:

$$\begin{aligned} V_{real}^\pi(s) &\geq Q_H^*(s, \pi) + V_{sim}(s_0) - \frac{\gamma\delta}{1-\gamma} \\ &\geq V_H^*(s_0) + V_{sim}(s_0) - \frac{\gamma\delta}{1-\gamma} - \delta \\ &\geq V_{real}^*(s) - \frac{\gamma\delta}{1-\gamma} - \delta - \gamma^H \epsilon \\ &= V_{real}^*(s) - \frac{\delta}{1-\gamma} - \gamma^H \epsilon \end{aligned}$$

from which the state result follows immediately.  $\square$

### Proof of Theorem 1:

*Proof.* The result follows directly from a combination of Lemma 4 and Lemma 2 by suppressing problem-dependent constants and lower order terms in the discount factor  $\gamma$ .  $\square$

## B ENVIRONMENT DETAILS

**Sim2Real Environment.** We use a 7-DoF Franka FR3 robot with a 1-DoF parallel-jaw gripper. Two calibrated Intel Realsense D455 cameras are mounted across from the robot to capture position of the object by color-thresholding pointcloud readings or retrieving pose estimation from aruco tags. Commands are sent to the controller at 5Hz. We restrict the end-effector workspace of the robot in a rectangle for safety so the robot arm doesn't collide dangerously with the table and objects outside the workspace. We conduct extensive domain randomization and randomize the initial gripper pose during simulation training. The reward is computed from measured proprioception of the robot and estimated pose of the object. Details for each task are listed below.

**Hammering.** For hammering, the action is 3-dimensional and sets delta joint targets for 3 joints of the robot using joint position control. The observation space is 12-dimensional and includes end-effector cartesian xyz, joint angles of the 3 movable joints, joint velocities of the 3 movable joints, the z position of the nail, and the xz position of the goal. Each trajectory is 50 timesteps. In simulation, we randomize over the position, damping, height, radius, mass, and thickness of the nail. Details are listed in Tab. 1.

The reward function is parameterized as  $r(t) = -10 \cdot r_{\text{nail-goal}}(t)$  where  $r_{\text{nail-goal}} = (\mathbf{r}_{\text{nail}})_z - (\mathbf{r}_{\text{goal}})_z$  represents the distance in the z dimension of the nail head to the goal, which we set to be the height of the board the nail is on.

**Puck Pushing.** For puck pushing, the action is 2-dimensional and sets delta cartesian xy position targets using end-effector position control. The observation space is 4-dimensional and includes end-effector cartesian xy and the xy position of the puck object. Each trajectory is 40 timesteps. In simulation, we randomize over the position of the puck. Details are listed listed in Tab. 3.

Let  $\mathbf{r}_{\text{ee}}$  be the cartesian position of the end effector and  $\mathbf{r}_{\text{obj}}$  be the cartesian position of the puck object. The reward function is parameterized as  $r(t) = -r_{\text{ee-goal}}(t) - r_{\text{obj-goal}}(t) + r_{\text{threshold}}(t) - r_{\text{table}}(t)$  where  $r_{\text{ee-goal}}(t) = \|\mathbf{r}_{\text{ee}}(t) - \mathbf{r}_{\text{obj}}(t) + [3.5\text{cm}, 0.0\text{cm}, 0.0\text{cm}]\|$  represents the distance of the end effector to the back of the puck,  $r_{\text{obj-goal}}(t) = \|(\mathbf{r}_{\text{obj}}(t))_x - 55\text{cm}\|$  represents the distance of the puck to the goal (which is the edge of the table along the x dimension),  $r_{\text{threshold}}(t) = \mathbb{I}[r_{\text{obj-goal}}(t) \geq 2.5\text{cm}]$  represents a goal reaching binary signal, and  $r_{\text{table}}(t) = \mathbb{I}[(r_{\text{obj}}(t))_z \leq 0.0]$  represents a binary signal for when the object falls of the table.

**Inserting.** For inserting, the action is 3-dimensional and sets delta cartesian xyz position targets using end effector position control. The observation space is 9-dimensional and includes end-effector cartesian xyz, the xyz of the leg, and the xyz of the table hole. Each trajectory is 40 timesteps. In simulation, we randomize the initial gripper position, position of the table, and friction of both the table and the leg.

Let  $\mathbf{r}_{\text{pos1}}(t)$  and  $\mathbf{r}_{\text{pos2}}(t)$  represent the Cartesian positions of the leg and table hole. Let:

$$x_{\text{distance}}(t) = \text{clip}(|\mathbf{r}_{\text{pos1},x}(t) - \mathbf{r}_{\text{pos2},x}(t)|, 0.0, 0.1)$$

$$y_{\text{distance}}(t) = \text{clip}(|\mathbf{r}_{\text{pos1},z}(t) - \mathbf{r}_{\text{pos2},z}(t)|, 0.0, 0.1)$$

$$z_{\text{distance}}(t) = \text{clip}(|\mathbf{r}_{\text{pos1},y}(t) - \mathbf{r}_{\text{pos2},y}(t)|, 0.0, 0.1)$$

Let the success condition be defined as:

$$r_{\text{success}}(t) = \mathbb{I}[x_{\text{distance}}(t) < 0.01 \text{ and } y_{\text{distance}}(t) < 0.01 \text{ and } z_{\text{distance}}(t) < 0.01]$$

The reward function is now:

$$r(t) = r_{\text{success}}(t) - 100 * (x_{\text{distance}}(t)^2 + y_{\text{distance}}(t)^2 + z_{\text{distance}}(t)^2)$$

**Sim2Sim Environment.** We additionally attempt to model a sim2real dynamics gap in simulation by taking the hammering environment and create a proxy for the real environment by fixing the domain randomization parameters, fixing the initial gripper pose, and rescaling the action magnitudes before rolling out in the environment.

## C IMPLEMENTATION DETAILS

**Algorithm Details.** We use SAC as our base off-policy RL algorithm for training in simulation and fine-tuning in the real world. For our method, we additionally add in two networks: a dynamics model that predicts next state given current state and action, and a state-conditioned value network which regresses towards the Q-value estimates for actions taken by the current policy. These networks are training jointly with the actor and critic during SAC training in simulation.

**Network Architectures.** The Q-network, value network, and dynamics model are all parameterized by a two-layer MLP of size 512. The dynamics model is implemented as a delta dynamics model where model predictions are added to the input state to generate next states. The policy network produces the mean  $\mu_a$  and a state-dependent log standard deviation  $\log \sigma_a$  which is jointly learned from the action distribution. The policy network is parameterized by a two-layer MLP of size 512, with a mean head and log standard deviation head on top parameterized by a FC layer.

**Pretraining in Simulation.** For hammering and puck pushing, we collect 25,000,000 transitions of random actions and pre-compute the mean and standard deviation of each observation across this dataset. We train SAC in simulation on the desired task by sampling 50-50 from the random action dataset and the replay buffer. We normalize our observations by the pre-computed mean and standard deviation before passing them into the networks. We additionally add Gaussian noise centered at 0 with standard deviation 0.004 to our observations with 30% probability during training. For inserting, we train SAC in simulation with no normalization. We train SAC with autotuned temperature set initially to 1 and a UTD of 1. We use Adam optimizer with a learning rate of  $3 \times 10^{-4}$ , batch size of 256, and discount factor  $\gamma = .99$ .

**Fine-tuning in Real World.** We pre-collect 20 real-world trajectories with the policy learned in simulation to fill the empty replay buffer. We then reset the critic with random weights and continue training SAC with a fixed temperature of  $\alpha = 0.01$  and with a UTD of  $2d$  with the pretrained actor and dynamics model. We freeze the value network learned from simulation and use it to relabel PBRS rewards during fine-tuning. During fine-tuning, for each state sampled from the replay buffer, we additionally hallucinate 5 branches off and add it to the training batch. As a result, our batch size effectively becomes 1536. The policy, Q-network, and dynamics model are all trained jointly on the real data during SAC fine-tuning. We don't train on any simulation data during real-world fine-tuning because we empirically found it didn't help fine-tuning performance in our settings.

1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079

Table 1: Domain randomization of hammering task in simulation

Name	Range
Nail x position (m)	[0.3, 0.4]
Nail z position (m)	[0.55, 0.65]
Nail damping	[250.0, 2500.0]
Nail half height (m)	[0.02, 0.06]
Nail radius (m)	[0.005, 0.015]
Nail head radius (m)	[0.03, 0.04]
Nail head thickness (m)	[0.001, 0.01]
Hammer mass (kg)	[0.015, 0.15]

Table 2: Domain randomization of puck pushing task in simulation

Name	Range
Puck x position (m)	[0.0, 0.3]
Puck y position (m)	[-0.25, 0.25]

Table 3: Domain randomization of inserting task in simulation

Name	Range
Parts x/y position (m)	[-0.05, 0.05]
Parts rotation (degrees)	[0, 15]
Parts friction	[-0.01, 0.01]

### D QUALITATIVE RESULTS

We analyze the characteristics of hallucinated states and value functions in Fig. 5. We visualize a trajectory of executing puck pushing in simulation using the learned policy in this plot. The red dots indicate states along a real rollout in simulation. The blue dots indicate hallucinated states branching off real states along a real rollout in simulation. The blue dots indicate hallucinated states generated by the learned dynamics model. The green heatmap indicates the value function estimates at different states. A corresponding image of the state is shown for two states. The trajectory shown in the figure shows the learned policy moving closer to the puck before pushing it. The value function heatmap shows higher values when the end effector is closer to the puck and lower values when further. Hallucinated states branching off each state show generated states for fine-tuning the learned policy.

Note that it is hard to directly visualize states and values due to the high-dimensionality of the state space. To get around this for puck pushing, we only show a part of the trajectory where the puck does not move. This allows us to visualize states and values along changes in only end effector xy.

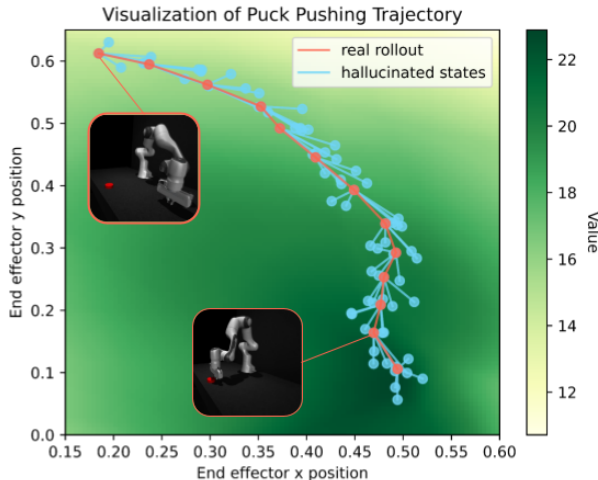


Figure 5: **Visualization of real rollout, hallucinated states, and value function.** The red dots indicate states along a real rollout in simulation. The blue dots indicate hallucinated states branching off real states generated by the learned dynamics model. The green heatmap indicates the value function estimates at different states. A corresponding image of the state is shown for two states. Since it is hard to directly visualize states and values due to the high-dimensionality of the state space, we only show a part of the trajectory where the puck does not move. This allows us to visualize states and values along changes in only end effector xy.

## E SIM-TO-SIM EXPERIMENTS

Here we additionally test each of the proposed methods on the sim-to-sim set-up from Du et al. (2021), which is meant to mock sim-to-real gaps but for familiar RL benchmark tasks. The results are depicted in Figure for the Walker Walk and Cheetah run environments. For both tasks, we use the precise settings from Du et al. (2021). Note that the general trend of these results matches our real world experiments – *SGFT* substantially accelerates learning and overcoming the dynamics gap between the ‘simulation’ and ‘real environments’.

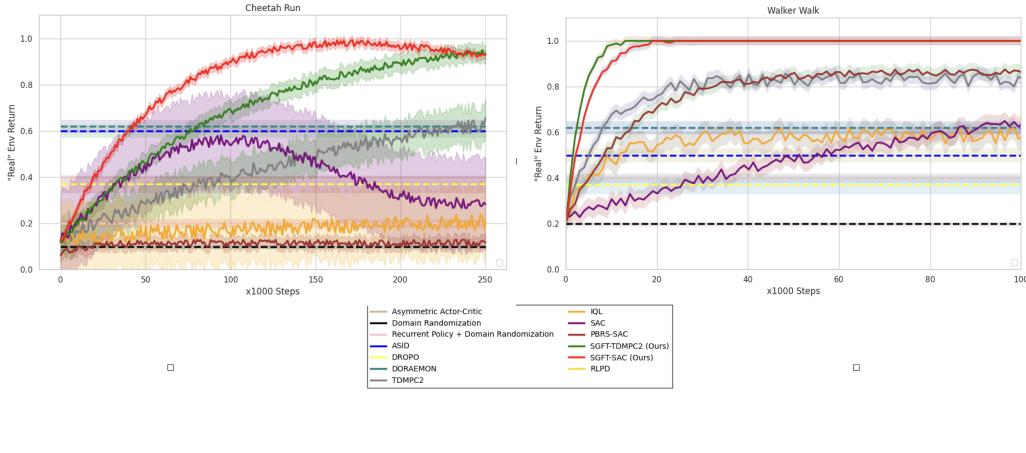


Figure 6: **Normalized Rewards for Sim-to-Sim Transfer.** We plot the normalized rewards for two sim-to-sim transfer tasks, where the rewards are normalized by the maximum reward achieved by any method.

## F PEDIGOGICAL EXAMPLE:

We use the following pedagogical example to begin building an intuition for why we might expect  $V_{sim}$  to also be improvable with respect to  $\mathcal{M}_{real}$ .

**Pedagogical Example.** Consider the following case where the real and simulated dynamics are both deterministic, namely,  $s' = f_{real}(s, a)$  and  $s' = f_{sim}(s, a)$  for some real and simulated transition maps  $f_{real}$  and  $f_{sim}$ . Further, assume for simplicity that  $\pi_{sim}$  is deterministic. Specifically, consider the case where  $s = (s_1, s_2) \in \mathcal{S} \subset \mathbb{R}^2$ ,  $a \in \mathcal{A} = \mathbb{R}$ , and the dynamics are given by:

$$f_{sim}(s, a) = \begin{bmatrix} s'_1 \\ s'_2 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \Delta t \left[ \begin{matrix} x_2 \\ \frac{g}{l} \sin(x_1) + a \end{matrix} \right]$$

$$f_{real}(s, a) = \begin{bmatrix} s'_1 \\ s'_2 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \Delta t \left[ \begin{matrix} x_2 \\ \frac{g}{l} \sin(x_1) + a + e(s_1, s_2) \end{matrix} \right]$$

These are the equations of motion for a simple pendulum (Wang et al., 2022) under an Euler discretization with time step  $\Delta t$ , where  $s_1$  is the angle of the arm,  $s_2$  is the angular velocity,  $a$  is the torque applied by the motor,  $g$  is the gravitational constant, and  $l$  is the length of the arm. The real-world dynamics contains unmodeled terms  $e(s_1, s_2)$ , which might correspond to complex frictional or damping effects. Consider the policy for the real world given by  $\pi_{real}(s) = \pi_{sim}(s) - e(s_1, s_2)$  and observe that  $f_{sim}(s, \pi_{sim}(s)) = f_{real}(s, \pi_{real}(s))$ . This implies that  $\gamma V_{sim}(f_{real}(s, \pi_{real}(s))) - V_{sim}(s) = \gamma V_{sim}(f_{sim}(s, \pi_{sim}(s))) - V_{sim}(s) = -r(s)$ , and thus  $V_{sim}$  is improvable with respect to  $\mathcal{M}_{real}$ , because it is improvable with respect to  $\mathcal{M}_{sim}$  by definition. Note that  $\pi_{sim}$  and  $\pi_{real}$  can differ substantially for a large gap  $e(s_1, s_2)$ .

**Main Insight.** To make this property precise, we observe the following fact: if we assume that for any state  $s$ , there is some  $a$  such that  $p_{real}(\cdot|s, a) = p_{sim}(\cdot|s, \pi_{sim}(s))$ , then  $V_{sim}$  is improvable with respect to  $\mathcal{M}_{real}$ . More generally, for many continuous control tasks, it is reasonable to expect that  $p_{sim}$  approximately captures the geometry of what motions are possible under  $p_{real}$ , even if the actions required to realize those motions in the two MDPs differ substantially, and thus it is

1134 reasonable to assume  $V_{sim}$  is improvable. This intuition is highlighted by our real-world learning  
1135 examples in cases where we use *SGFT* with a prediction horizon of  $H = 1$ . In these cases the  
1136 learned policy is able to greedily follow  $V_{sim}$  at each state and reach the goal, even in the face of  
1137 large dynamics gaps.  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187