Rapidly Adapting Policies to the Real-World via Simulation-Guided Fine-Tuning

Patrick Yin* **Tyler Westenbroek*** **Simran Bagaria** **Kevin Huang** **Ching-An Cheng**

Andrey Kolobov **Abhishek Gupta**

Abstract: Robot learning requires a considerable amount of data to realize the promise of generalization. However, it can be challenging to actually collect the required magnitude of high-quality entirely in the real world. Simulation can serve as a source of plentiful data, wherein techniques such as reinforcement learning can obtain 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 forceful manipulation is necessary. This makes real-world fine-tuning of policies pretrained in simulation an attractive approach to robot learning. However, exploring the real-world dynamics with standard RL fine-tuning techniques is to inefficient for many real-world applications. This paper introduces Simulation-Guided Fine-Tuning, a general framework which leverages the structure of the simulator to guide exploration, substantially accelerating adaptation to the real-world. We demonstrate our approach across several manipulation tasks in the real world, learning successful policies for problems that are challenging to learn using purely real-world data. We further provide theoretical backing for the paradigm. Website: `weirdlabuw.github.io/sgft`.

1 Introduction

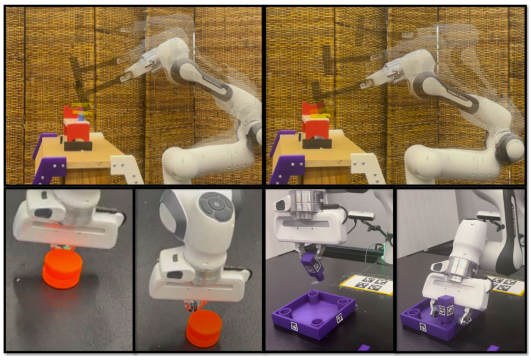

Fig. 1: Contact-rich tasks solved with *SGFT*– Hammering, pushing, and inserting.

Robot learning offers a pathway to building robust, general-purpose robotic agents which 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. Yet this raises a fundamental question: *How do we cheaply obtain and leverage such data sets at scale?* Real-world data collection via teleoperation [1, 2] can generate high-quality trajectories, but scales linearly with human effort. Community-driven teleoperation [3, 4] takes this idea further, but current datasets are still orders of magnitude smaller than those powering vision and language applications.

Massively parallelized physics simulation [5, 6] can cheaply generate vast quantities of synthetic robot data. Moreover, applying search techniques such as reinforcement learning (RL) in simulation can yield near-optimal behavior. By exploiting techniques such as extensive randomization of initial conditions, dynamics randomization [7, 8], and automatic scene generation [9, 10], these data sets can obtain extensive coverage over situations a robot is likely to encounter in real.

Unfortunately, simulation-generated data is not a silver bullet. Even when considerable effort is invested in constructing simulators, there is often an inherent, irreducible modeling gap between

*These authors contributed equally.

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

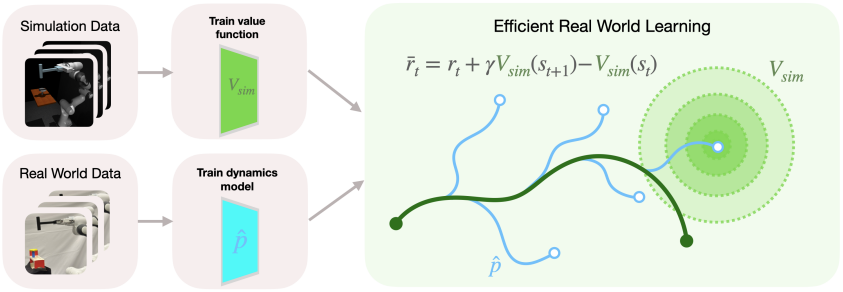

Fig. 2: Depiction of a model-based instantiation of *SGFT*: (1) A value function is learned in simulation to guide real-world exploration with short-horizon reshaped rewards (2) a model fit to the real-world dynamics generates short synthetic rollouts, providing a source of *data augmentation*. Together, this approach leverages simulation data (through V_{sim}) to capture successful long-horizon behaviors, and small amounts of real world data (through $\hat{p}$) to learn how to execute these behaviors in real.

the physics of the simulation and of the real world. Thus, despite impressive performance for many tasks, methods which transfer policies from simulation to reality zero-shot [11, 12, 7, 8] still display failure modes when they encounter situations outside simulated training distribution [13]. While efforts in system identification [14, 15] can resolve some of this modeling error, they do not address situations where the simulation is fundamentally *misspecified*. Namely, cases where no choice of simulator parameters accurately model reality. This arises, for instance, in behaviors like hammering in a nail, where the modeling of high-impact, deformable contact remains an open problem [16, 17].

The question becomes: *can inaccurate simulation models be useful in the face of fundamental misspecifications?* A natural technique to leverage these inaccurate simulation models has been to train a near-optimal policy in simulation, and use it as an initialization for RL fine-tuning in the real-world using standard RL algorithms [13, 18]. Real world RL can overcome misspecification by training directly on data from the target domain. However, *tabula rasa* exploration of the real world dynamics is inefficient, leading to slow real-world policy improvement. Prior work has additionally considered mixing simulated and real data during policy optimization, either through co-training [19], simply initializing the replay-buffer with simulation data [20, 21], or by adaptively sampling the simulated data set and to up-weight transition which approximately match the real-world dynamics [22, 23, 24, 25]. While these data augmentation approaches are effective in regimes where the simulation model is accurate, the do not accelerate the discovery of successful action in regions where the model is fundamentally inaccurate due to misspecification.

In this work we argue that, despite inherently getting the finer details wrong, physics simulators capture the rough structure of real-world dynamics well enough to provide guidance for targeted, efficient real-world exploration. Leveraging this insight, we propose *Simulation-Guided Fine-Tuning* (*SGFT*), a general framework for efficient real-world fine-tuning. *SGFT* takes the perspective that the rough structure of successful behaviors (such as moving an object closer to a goal position) are preserved between simulation and reality, even if the low-level sequences of states and action needed to realize those behaviors in the two domains differ substantially in the short term. Thus, pre-trained simulation policies can be viewed as 'approximate' experts for controlling the real world. The goal of *SGFT* is to rapidly adapt this behavior to the nuisances of real-world dynamics, while preserving as much of the structure of the pre-trained policy as possible.

SGFT optimizes an auxiliary short-horizon H-step look ahead objective wherein the value function from simulation V_{sim} approximately bootstraps long-horizon returns. Reducing the search horizon makes the real-world policy search problem significantly more sample-efficient [26, 27]. Intuitively, V_{sim} roughly captures the behaviors of π_{sim} in a form that is *robust to dynamics shifts*. Indeed, by optimizing the aforementioned objective, *SGFT* discovers sequences of actions which will increase V_{sim} over H-step interval *under the real dynamics*. This will cause the fine-tuned policy to approximately match the observed behavior of π_{sim} in simulation, but when operating in real. Crucially, we keep V_{sim} frozen during training to preserve this strong learning signal during the fine-tuning process. We observe that this prevents *catastrophic forgetting*, wherein policies completely de-learn useful behaviors during fine-tuning. We summarize our contributions as follows.

Hardware Results: We instantiate the *SGFT* framework using two base model-based reinforcement learning algorithms and evaluate the framework in situations where direct sim-to-real transfer fails. Across a variety of dynamic real-world manipulation tasks, we find that *SGFT* learns substantially more performant policies than baseline finetuning methods with substantially fewer samples.

Theory: In Appendix B, we provide theoretical backing for these empirical results. Specifically, we demonstrate 1) *SGFT* can lead to highly effective policies even when there is a large dynamics gap and 2) *SGFT* can be paired with a base model-based RL method to learn effectively in low-data regimes where the learned model is likely to be highly inaccurate. Specifically, we show that our approach enables the use of short-horizon model predictions, which overcomes the fundamental challenge of *compounding errors* [28] faced by MBRL methods.

Related Work: A detailed discussion of prior work is left to appendix A.

Preliminaries: Let $s \in \mathcal{S}$ and $a \in \mathcal{A}$ be state and action spaces. Our goal is to control a real-world environment defined by unknown dynamics $s' \sim p_{real}(\cdot|s,a)$ by solving the MDP $\mathcal{M}_r = (\mathcal{S}, \mathcal{A}, p_{real}, \rho^0_{real}, r, \gamma)$ with initial real-world state distribution ρ^0_{real}, reward function r, and discount factor $\gamma \in [0,1)$. Given policy π, we let $d^\pi_{real}(s)$ denote the distribution over trajectories generated by applying π with initial condition $s_0 = s$. Defining the value function under π by $V^\pi_{real}(s) = \mathbb{E}_{s_t \sim d^\pi_{real}(s)}[\sum_t \gamma^t r_t(s_t)]$, our objective is: $\pi^*_{real} \leftarrow \sup_\pi \mathbb{E}_{s\sim\rho^0_{real}}[V^\pi_{real}(s)]$. Define the optimal value $V^*_{real}(s) := \sup_\pi V^\pi_{real}(s)$. We assume access to a simulation environment $s' \sim p_{sim}(s,a)$ which defines an approximation $\mathcal{M}_{sim} := (\mathcal{S}, \mathcal{A}, p_{sim}, \rho^0_{sim}, r, \gamma)$. We let π_{sim} denote a policy pretrained in $\mathcal{M}_{sim}$, with V_{sim} the associated value function.

2 Simulation-Guided Fine-Tuning

2.1 Reward Shaping and Horizon Shortening

We implement our horizon-shortening approach using the Potential-Based Reward Shaping (PRBS) formalism [29], which replaces the reward $r(s)$ with $\bar{r}(s,s') = r(s) + \gamma\Phi(s') - \Phi(s)$ for some function Φ. Rather than optimizing the original infinite-horizon objective we will instead investigate optimizing the H-step return $\sum_{t=0}^{H-1}\gamma^t\bar{r}_t = \sum_{t=0}^{H-1}\gamma^t r_t + \gamma^H\Phi(s_H) - \Phi(s_0)$, where the equality follows by telescoping out terms. Intuitively, the addition of the potential term encourages policy search algorithms to follow Φ by increasing its value during each transition. 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 biases the returns, optimizing the H-step return from a given initial condition presents a significantly more tractable problem.

2.2 H-step Simulation-Guided Expert Policies

We propose to set $\Phi(s) = V_{sim}(s)$ (i.e. the value of the policy π_{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 π_{sim} from $\mathcal{M}_{sim}$ to $\mathcal{M}_{real}$. We will use 'tilde' notation $\tilde{\pi}_H = \{\pi_{H,0}, \pi_{H,1}, \ldots, \pi_{H,H-1}\}$ 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:

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

$$V^*_H(s) := \sup_{\tilde{\pi}_H} V^{\tilde{\pi}_H}_H(s) \qquad Q^*_H(s,a) = \mathbb{E}_{s'\sim p_{real}(s,a)}\left[\gamma V^*_{H-1}(s') + \bar{r}(s,s')\right] \tag{2}$$

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, $V^{\tilde{\pi}_H}_H$ is equivalent to the planning objective used by model predictive control (MPC) methods [30, 31, 32, 33] with H-step look-ahead and a terminal reward of V_{sim} (when the ground truth dynamics are known). Thus, we define the *H-step simulation guided MPC expert* via: $\pi^*_H(\cdot|s) \leftarrow \max_\pi Q^*_H(s,\pi(s))$, which simply applies the optimal action under the H-step look ahead at each state. Intuitively, π^*_H will greedily follow V_{sim} at every state when $H = 1$, and as we take $H \to \infty$ the behavior of π_H will recover the behavior of π^*_{real}. Thus, π^*_H can be viewed as a policy which has adapted the behavior of π_{sim} to follow V_{sim} *along the real-world*

dynamics, and for smaller values of H we should expect π^*_H to retain more of the behavior of π_{sim}. However, because we do not know the p_{real} we do not know the actions taken by this expert policy.

2.3 The *SGFT* Framework

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

Require: Pretrained policy π_{sim} and value function V_{sim}

$\pi \leftarrow \pi_{sim}$
for each iteration k **do**
  for time step $t = 1, ..., T$ **do**
    $a_t \sim \pi(\cdot|s_t)$
    Observe the state s_{t+1} and the reward r_t.
    $\bar{r}_t \leftarrow r_t + \gamma V_{sim}(s_{t+1}) - V_{sim}(s_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, \bar{r}_t, s_{t+1})$
  end for
  Approx. optimize $\pi \leftarrow \max_\pi Q^*_H(s, \pi(s_j))$
  $\forall s_j \in \mathcal{D}$ using observed transitions $\mathcal{D}$.
end for

Even though we do not have direct access to π^*_H, we can implicitly learn its actions by optimizing policies to maximize the H-step return Equation (3) starting from every initial condition $s \in \mathcal{S}$. Thus, we propose the conceptual *Simulation-Guided Fine-Tuning* (*SGFT*) framework, which is defined via pseudo-code in Algorithm 1. *SGFT* fine-tunes π_{sim} to succeed under the real-world dynamics by iteratively 1) unrolling the current policy to collect transitions from p_{real} and 2) using the current data set $\mathcal{D}$ of transitions to approximately optimize $\pi \leftarrow \max_\pi Q^*_H(s, \pi(s_j))$ at each state s_j the agent has visited. In [32] a model-free method for approximating the optimization in step 2) is proposed, and we discuss how this step can be performed with model-based methods below. By optimizing $\pi \leftarrow \max_\pi Q^*_H(s, \pi(s_j))$, *SGFT* is implicitly attempting to learn and approximate the actions taken π^*_H at every state the agent has visited (which is reminiscent of the learning loop used by DAgger [34], with the obvious caveat that in our case the expert are not directly available).

2.4 Leveraging Short Model Roll-outs

Model-based reinforcement learning (MBRL) holds the promise of learning a generative model $\hat{p}$ to rapidly learn effective policies with significantly less real-world data than model-free methods [28]. However, as discussed in appendix A, the central challenge for MBRL is that small errors in $\hat{p}$ can quickly compound over multiple steps, degrading the quality of predictions. As a consequence, learning a model which is accurate enough to solve long-horizon problems can often take as much data as solving the task with modern model-free methods [35, 36]. By boot-strapping 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 $\tilde{\pi}_H = \{\pi_{H,t}\}_{t=0}^{H-1}$:

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

$$\hat{V}^*_H(s) := \sup_{\tilde{\pi}_H} \hat{V}^{\tilde{\pi}_H}_H(s) \qquad \hat{Q}^*_H(s,a) = \mathbb{E}_{s' \sim \hat{p}(s,a)}\left[\gamma \hat{V}^*_{H-1}(s') + \bar{r}(s,s')\right]. \tag{3}$$

Note the corresponding MPC policy which uses $\hat{p}$ is given by: $\hat{\pi}^*_H(\cdot|s) \leftarrow \arg\max_\pi \hat{Q}^*_H(s, \pi)$. We next discuss two broad approaches which use $\hat{p}$ to approximately learn $\hat{\pi}^*_H$ at each iteration.

Improved Sample Efficiency with Data Augmentation (Algorithm 2). The generative model $\hat{p}$ can be used for *data augmentation* by generating a data set of synthetic rollouts $\hat{\mathcal{D}}$ to supplement the real-world data set $\mathcal{D}$ [28, 37, 38]. The combined data-set 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 [28] 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 then repeatedly a) generates a data set $\hat{\mathcal{D}}$ of synthetic H-step rollouts under the current policy π starting from states in $\mathcal{D}$ b) approximately solves $\pi \leftarrow \max_{\bar{\pi}} Q^*_H(s, \bar{\pi}(s))$ at observed real-world states using the augmented data set $\hat{\mathcal{D}} \cup \mathcal{D}$ and a base model-free method. In Section 3, we implement this approach with SAC [39].

Algorithm 2 Dyna-*SGFT*

Require: Policy π_{sim} and value V_{sim}. Set $\pi \leftarrow \pi_{sim}$.

for each iteration k **do**
  Generate rollout $\{(s_t, a_t, r_t, s_{t+1})\}_{t=0}^{T}$ under π.
  $\bar{r}_t \leftarrow r_t + \gamma V_{sim}(s_{t+1}) - V_{sim}(s_t)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, \bar{r}_t, s_{t+1})$
  Fit generative model $\hat{p}$ with $\mathcal{D}$.
  Generate synthetic branched rollouts $\hat{\mathcal{D}}$ under π.
  Approx. optimize $\pi \leftarrow \max_\pi Q_H^*(s, \pi(s_j))$
    $\forall s_j \in \mathcal{D}$ using augmented dataset $\hat{\mathcal{D}} \cup \mathcal{D}$
end for

Online Planning (Algorithm 3). The most straightforward way to approximate the behavior of π_H^* is simply to apply the MPC controller $\hat{\pi}_H^*$ generated using the current best guess for the dynamics $\hat{p}$. Algorithm 3 provides general pseudo code for this approach, which iteratively 1) rolls out $\hat{\pi}_H^*$ (which is calculated using online optimization and $\hat{p}$ [40]) then 2) updates the model on the current data set of transitions $\mathcal{D}$. This broad approach encompasses a wide array of methods [41, 42] . In Section 3, we implemented this approach using the TDMPC-2 [31].

Algorithm 3 MPC-*SGFT*

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

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

3 Experiments

We aim to answer the following questions: (1) Can *SGFT* facilitate tractable online fine-tuning of policies for dynamic, real-world manipulation tasks? (2) Does *SGFT* improve the sample-efficiency of online fine-tuning over benchmarks? (3) Can *SGFT* learn policies which outperform direct transfer techniques which leverage extensive domain randomization and/or system identification?

To answer these questions, we test a variety of methods on three real-world manipulation tasks illustrated in Figure 3, demonstrating that both instantiations of *SGFT* excel at learning policies with minimal real-world data. Specifically, we consider **Hammering**, **Insertion**, and **Pushing** tasks, and detail the task set-ups and sim-to-real gaps in Appendix D. Each of these tasks is evaluated using a Franka FR3 robot operating with either Cartesian position control or joint position control. We evaluate the following classes of methods:

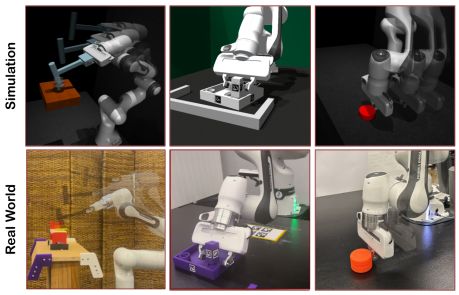

Fig. 3: **Sim-to-Real Setup** Simulation setup for pretraining (**top**) and execution of real-world fine-tuning (**bottom**) of real-world hammering (**left**), insertion (**middle**), and pushing (**right**).

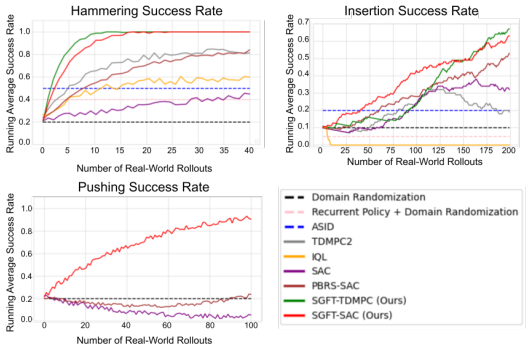

Fig. 4: **Real-world success rates during the course of online fine-tuning.** We plot task success rates over number of finetuning rollouts for the tasks described in Sec. 3. We see that *SGFT* yields significant improvements in success and efficiency.

***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. We use $H = 1$ for all our expirements. ***SGFT*-TDMPC-2** uses TDMPC-2 [31] as a backbone. The base method learns a critic, a policy, and an approximate dynamics model through interaction data. It then performs MPC using the approximate model and learned critic as a terminal reward. 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) for the online planning objective. For our experiments, we use $H = 4$ and default hyperparameters [31].

Baseline Fine-tuning Methods. The **SAC** baseline fine-tunes the pre-trained policy to solve the original MDP $\mathcal{M}_{real}$ using SAC [39] – it does not use of shaping or horizon shortening. **PBRS** fine-tunes the policy under a reshaped infinite-horizon MDP using the reshaped reward $\bar{r}$ and SAC [39] as the policy optimizer; namely, this approach does not leverage horizon shortening. **TDMPC-2** fine-tunes the entire TDMPC-2 architecture [31] 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 [43]. It does not make use of reward shaping or horizon shortening. 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 [11]. **ASID** [14] 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 then deployed zero-shot.

3.1 Analysis

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

The second class of comparison methods include offline pretraining with online finetuning techniques like IQL [43] and SAC [39]. Whether model-free or model-based, the SGFT finetuning methods (ours) substantially outperform these techniques in terms of efficiency and asymptotic performance. Moreover, they prevent *catastrophic forgetting*, wherein finetuning leads to periods of sharp degradation in the policies effectiveness. This suggests that simulation can offer more guidance during real-world policy search than just an initialization for subsequent finetuning. Our full system consistently leads to significant improvement from fine-tuning, achieving 100% success for hammering within a few minutes and pushing within an hour, and 70% success for the long-horizon insertion task within two hours. The fact that SGFT outperforms both TD-MPC2 [31] and PBRS-SAC, suggests that efficient finetuning requires a *combination* of both short model rollouts plus value-driven reward shaping. And lastly, 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 H), and visualizations of transferred value functions (Appendix G).

4 Limitations and Future Work

In this work, we present *SGFT*, a technique for efficient sim-to-real finetuning 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 rather than low-dimensional states would make this paradigm 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

[1] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[2] D. C. team. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics Science and Systems (RSS)*, 2024.

[3] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

[4] O.-X.-E. Collaboration. Open x-embodiment: Robotic learning datasets and rt-x models. 2024.

[5] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

[6] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning. In *NeurIPS-2021 Datasets and Benchmarks Track*, 2021.

[7] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[8] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[9] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.

[10] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi, and R. Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/27c546ab1e4f1d7d638e6a8dfbad9a07-Abstract-Conference.html.

[11] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: rapid motor adaptation for legged robots. In *RSS*, 2021.

[12] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[13] L. M. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. In *ICRA*, 2022.

[14] M. Memmel, A. Wagenmaker, C. Zhu, P. Yin, D. Fox, and A. Gupta. ASID: active exploration for system identification in robotic manipulation. *CoRR*, abs/2404.12308, 2024.

[15] P. Huang, X. Zhang, Z. Cao, S. Liu, M. Xu, W. Ding, J. Francis, B. Chen, and D. Zhao. What went wrong? closing the sim-to-real gap via differentiable causal discovery. In J. Tan, M. Toussaint, and K. Darvish, editors, *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pages 734–760. PMLR, 2023. URL https://proceedings.mlr.press/v229/huang23c.html.

[16] B. Acosta, W. Yang, and M. Posa. Validating robotics simulators on real-world impacts. *IEEE Robotics and Automation Letters*, 7(3):6471–6478, 2022.

[17] J. Levy, T. Westenbroek, and D. Fridovich-Keil. Learning to walk from three minutes of data with semi-structured dynamics models. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=evCXwlCMIi.

[18] Y. Zhang, L. Ke, A. Deshpande, A. Gupta, and S. S. Srinivasa. Cherry-picking with reinforcement learning. In *RSS*, 2023.

[19] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *CoRR*, abs/2403.03949, 2024. doi:10.48550/ARXIV.2403.03949. URL https://doi.org/10.48550/arXiv.2403.03949.

[20] L. Smith, I. Kostrikov, and S. Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

[21] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[22] B. Eysenbach, S. Asawa, S. Chaudhari, S. Levine, and R. Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*, 2020.

[23] J. Liu, H. Zhang, and D. Wang. Dara: Dynamics-aware reward augmentation in offline reinforcement learning. *arXiv preprint arXiv:2203.06662*, 2022.

[24] K. Xu, C. Bai, X. Ma, D. Wang, B. Zhao, Z. Wang, X. Li, and W. Li. Cross-domain policy adaptation via value-guided data filtering. *Advances in Neural Information Processing Systems*, 36:73395–73421, 2023.

[25] H. Niu, Y. Qiu, M. Li, G. Zhou, J. Hu, X. Zhan, et al. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. *Advances in Neural Information Processing Systems*, 35:36599–36612, 2022.

[26] C. Laidlaw, S. J. Russell, and A. Dragan. Bridging rl theory and practice with the effective horizon. In *Advances in Neural Information Processing Systems*, volume 36, pages 58953–59007, 2023.

[27] Y. Li, R. Wang, and L. F. Yang. Settling the horizon-dependence of sample complexity in reinforcement learning. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 965–976. IEEE, 2021. doi:10.1109/FOCS52979.2021.00097. URL https://doi.org/10.1109/FOCS52979.2021.00097.

[28] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[29] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

[30] A. Jadbabaie, J. Yu, , and J. Hauser. Unconstrained receding-horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 46(5):776–783, 2001.

[31] N. Hansen, H. Su, and X. Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=0xh5CstDJU.

[32] W. Sun, J. A. Bagnell, and B. Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *arXiv preprint arXiv:1805.11240*, 2018.

[33] M. Bhardwaj, S. Choudhury, and B. Boots. Blending mpc & value function approximation for efficient reinforcement learning. *arXiv preprint arXiv:2012.05909*, 2020.

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

[35] X. Chen, C. Wang, Z. Zhou, and K. Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

[36] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *arXiv preprint arXiv:2110.02034*, 2021.

[37] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[38] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pages 2829–2838. PMLR, 2016.

[39] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

[40] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[41] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[42] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International conference on machine learning*, pages 7444–7453. PMLR, 2019.

[43] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2021.

[44] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *ICRA*, 2019.

[45] F. Ramos, R. Possas, and D. Fox. Bayessim: Adaptive domain randomization via probabilistic inference for robotics simulators. In *RSS*, 2019.

[46] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-hand object rotation via rapid motor adaptation. In *CoRL*, 2022. URL https://proceedings.mlr.press/v205/qi23a.html.

[47] W. Yu, J. Tan, C. K. Liu, and G. Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *RSS*, 2017. URL http://www.roboticsproceedings.org/rss13/p48.html.

[48] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[49] A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[50] H. Hu, S. Mirchandani, and D. Sadigh. Imitation bootstrapped reinforcement learning. *arXiv preprint arXiv:2311.02198*, 2023.

[51] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36, 2024.

[52] A. Gupta, A. Pacchiano, Y. Zhai, S. M. Kakade, and S. Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6255f22349da5f2126dfc0b007075450-Abstract-Conference.html.

[53] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, page 1433–1438. AAAI Press, 2008. ISBN 9781577353683.

[54] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[55] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine. Variational inverse control with events: A general framework for data-driven reward definition. *NeurIPS*, 2018.

[56] K. Li, A. Gupta, A. Reddy, V. H. Pong, A. Zhou, J. Yu, and S. Levine. Mural: Meta-learning uncertainty-aware rewards for outcome-driven reinforcement learning. In *ICML*, 2021.

[57] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[58] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[59] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *Int. J. Robotics Res.*, 43(4):572–587, 2024. doi:10.1177/02783649231224053. URL https://doi.org/10.1177/02783649231224053.

[60] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019. URL http://arxiv.org/abs/1912.06680.

[61] C.-A. Cheng, A. Kolobov, and A. Swaminathan. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems*, 34:13550–13563, 2021.

[62] T. Westenbroek, F. Castaneda, A. Agrawal, S. Sastry, and K. Sreenath. Lyapunov design for robust and efficient robotic reinforcement learning. *arXiv preprint arXiv:2208.06721*, 2022.

[63] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, jul 1991. ISSN 0163-5719. doi:10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

[64] T. Wang and J. Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

[65] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33: 14129–14142, 2020.

[66] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

[67] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *International conference on machine learning*, pages 703–711. PMLR, 2017.

[68] C.-A. Cheng, X. Yan, N. Ratliff, and B. Boots. Predictor-corrector policy optimization. In *International Conference on Machine Learning*, pages 1151–1161. PMLR, 2019.

[69] C.-A. Cheng, X. Yan, and B. Boots. Trajectory-wise control variates for variance reduction in policy gradient methods. In *Conference on Robot Learning*, pages 1379–1394. PMLR, 2020.

[70] T. Che, Y. Lu, G. Tucker, S. Bhupatiraju, S. Gu, S. Levine, and Y. Bengio. Combining model-based and model-free rl via multi-step control variates. 2018.

[71] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[72] L. Grune and A. Rantzer. On the infinite horizon performance of receding horizon controllers. *IEEE Transactions on Automatic Control*, 53(9):2100–2111, 2008.

[73] J. Wang, Q. Xue, L. Li, B. Liu, L. Huang, and Y. Chen. Dynamic analysis of simple pendulum model under variable damping. *Alexandria Engineering Journal*, 61(12):10563–10575, 2022.

[74] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation, 2023. URL https://arxiv.org/abs/2305.12821.

[75] Y. Du, O. Watkins, T. Darrell, P. Abbeel, and D. Pathak. Auto-tuned sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1290–1296. IEEE, 2021.

A Related Work

Simulation-to-Reality Transfer. In this work, we assume that perception 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 [44, 45, 14], it cannot overcome inherent model misspecification, as we show in our experimental evaluation. Learning adaptive policies to account for changing real-world dynamics [46, 11, 47] can account for some types of dynamics change, but is unable to guide exploration and adapt beyond the training range. Perhaps most related is [13, 18], where simulation policies are fine-tuned with off-policy RL. However, besides initialization, simulation is not used to guide exploration throughout fine-tuning. In contrast, our work proposes using simulation information throughout the process of fine-tuning to improve efficiency.

Fine-tuning in Reinforcement Learning. Our work is related to algorithms for fine-tuning in RL with online data collection, primarily from offline RL or imitation initializations. These algorithms typically aim to provide an initialization that can continue improving with standard off-policy RL [48, 49, 43, 50, 51]. 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. In this work, we utilize the pretraining in simulation not just for initialization, but also to provide guidance throughout the improvement process.

Reward Design in Reinforcement Learning. A significant component of our methodology is learning dense reward shaping from simulation to guide real-world fine-tuning. This is closely tied to the problem of reward design and reward inference in RL [52]. While a challenging problem tabula rasa, prior techniques have attempted to infer rewards from expert demos [53, 54], success examples [55, 56], LLMs [57, 58] or heuristics [59, 60]. We rely on simulation to provide reward supervision using the PBRS formalism [29], and shorten the horizon of the learning task to improve the sample comlexity of real-world learning [61, 62]. A final complementary line of work to our comes from [22, 23, 24, 25] which relabels rewards from off-task (simulated) data, effectively up-weighting transitions which approximately match the dynamics observed in the target domain. While these works focus on the *retrieval* of useful samples from prior data sets with shifted dynamics, our approach uses prior data to guided the discovery of novel sequences of states and actions in the target domain. In principle, these techniques could be used in conjunction, although we leave this to future work.

Model-Based Reinforcement Learning. A significant body of work on model-based RL learns models for the dynamics to perform data augmentation [63, 64, 28, 65, 66] for downstream policy optimization algorithms, plan online using the model [41, 42] and model predictive control (MPC), or to be used as a control variate to reduce variance of policy gradient methods [67, 68, 69, 70]. 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 [31, 33, 71] yielding easier decision-making problems, but learning such a critic from scratch can still require large amounts of on-task data. We demonstrate that for many real-world continuous control problems critics learned entirely in simulation can be robustly transferred to the real-world and substantially accelerate model-based learning.

B Theoretical Analysis

In this section we analyze the effectiveness of the H-step simulation-guided expert π^*_H. Specifically, we seek suboptimality bounds for this agent and to understand when the behavior of this idealized policy will be robust to the errors made by MBRL techniqus 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 $\hat{p}$, as this will be the case we face when fine-tuning in the real-world with a small number of samples. To set the stage for this analysis, we first relate the results from several prior theoretical analyses from the RL literature.

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:*

$$V^*_{real}(s) - V^{\hat{\pi}^*_H}_{real}(s) \leq O\left(\frac{\gamma}{1-\gamma}\alpha H + \frac{\gamma^H}{1-\gamma^H}\epsilon\right), \tag{4}$$

where $\hat{\pi}^*_H$ *is the MPC policy under the approximate mode as in* **??**

This result is a direct translation of [33, Theorem 3.1] to the notation 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 π^*_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 γ^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 α 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 α, which will arise in low-data regimes.

However, because results like **??** 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. That is: 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 [61], which is similar to properties from other works [62, 72]:

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

$$\max_a \mathbb{E}_{s'\sim p_{real}(s,a)}[\gamma V_{sim}(s')] - V_{sim}(s) \geq -r(s). \tag{5}$$

Namely, V_{sim} is improvable with respect to $\mathcal{M}_{real}$ if there exists a policy which 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 (such as 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 short horizon to reach these desirable states. V_{sim} is learned under the simulation dynamics and policy π_{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 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 $s' = f_{real}(s,a)$ for some real and simulated transition maps f_{real} and f_{sim}. Further assume for simplicity that π_{sim} is deterministic. Specifically, consider the case where $s = (s_1, s_2) \in \mathcal{S} \subset \mathcal{R}^2$, $a \in \mathcal{A} = \mathcal{R}$, and the dynamics are given by:

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

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

These are the equations of motion for a simple pendulum [73] under an Euler discretization with time-step Δt, 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 an unmodeled terms $e(s_1, s_2)$, which might correspond to complex frictional or damping terms. 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 π_{sim} and π_{real} can differ substantially for a large gap $e(s_1, s_2)$.

Main Insight. To make this property precise, we observe the fact: suppose 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 reasonable to assume V_{sim} is improvable. This intuition is high-lighted by our real-world learning examples in cases where we use *SGFT* with a prediction horizon of $H = 1$; in these cases the learned policy is able to greedily follows V_{sim} at each state and reach the goal, even in the face of large dynamics gaps. 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:*

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

where $\hat{\pi}^*_H$ is the MPC policy under the approximate mode as in **??**.

Proof can be found in the Appendix. At a high-level, the proof uses arguments similar to [72] to bound the suboptimality of the expert policy π^*_H, and then combines this bound with the perturbation bounds from [33] to bound how errors in the dynamics lead to additional suboptimaly. Note that that the dependence on H and α is identical to the bound from Proposition **??** above. However, the scaling for the term involving ϵ 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 is 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 $\hat{\pi}^*_H$.

C Proofs

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

Lemma 1. *[33, Lemma A.1.] Suppose that $\|\hat{p}(s,a) - p_{real}(s,a)\|_1 \leq \alpha$. Further 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. Then, for each policy $\tilde{\pi}$ we may bound the H-step returns under the model and true dynamics by:*

$$\|\hat{V}^{\tilde{\pi}}_H(s) - V^{\tilde{\pi}}_H\|_\infty \leq \gamma\left(\frac{1-\gamma^{H-1}}{1-\gamma}\frac{\Delta r}{2} + \gamma^H \frac{\Delta V}{2}\right) \cdot \alpha H. \tag{7}$$

Proof. This result follows imediatly from the proof of [33, Lemma A.1.], with changes to notation and noting that we assume access to the true reward. □

Lemma 2. *Suppose that $\|\hat{p}(s,a) - p_{real}(s,a)\|_1 \leq \alpha$. Further 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. Then for each state $s \in \mathcal{S}$ we have:*

$$V^{\hat{\pi}^*_H}_H(s) - V^*_H(s) \leq \left(\frac{1-\gamma^{H-1}}{1-\gamma}\Delta r + \gamma^H \Delta V\right) \tag{8}$$

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

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

$$V^*_H(s) \leq \hat{V}^{\tilde{\pi}^*_H}(s) + \gamma\left(\frac{1-\gamma^{H-1}}{1-\gamma}\frac{\Delta r}{2} + \gamma^H \frac{\Delta V}{2}\right) \cdot \alpha H. \tag{9}$$

$$\hat{V}^{\hat{\pi}^*_H}_H(s) \leq V^{\hat{\pi}^*_H}_H(s) + \gamma\left(\frac{1-\gamma^{H-1}}{1-\gamma}\frac{\Delta r}{2} + \gamma^H \frac{\Delta V}{2}\right) \cdot \alpha H. \tag{10}$$

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

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* $V^*_H(s) \geq V^*_{H-1}(s)$ *for each* $s \in \mathcal{S}$. *Then for each* $s \in \mathcal{S}$ *we have:*

$$V^*_H(s) \geq V^*_{H-1}(s) \tag{11}$$

Proof. Fix an initial condition $s_0 \in \mathcal{S}$. Let π be arbitrary, and fix the shorthand $\pi^* = \{\pi^*_0, \ldots, \pi^*_{H-1}\}$ for the time-varying policy $\pi^* \leftarrow \max_{\hat{\pi}} V^{\hat{\pi}}_{H-2}(s_0)$. Then, concatenate these policies to define: $\bar{\pi} = \{\pi^*_1, \ldots, \pi^*_{H-2}, \pi\}$, which is simply the result of applying the optimal policy for the $(H-1)$-step look ahead objective Equation (3) starting from s_0, followed by applying π for a single step. Letting the following distributions over trajectories by generated by π^*, by the definition of V^*_H:

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

Now, since our choice of π used to define $\bar{\pi}$ was arbitrary, we choose π to be deterministic and 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 assumption made for the result. This choice of policy grantees that:

$$\mathbb{E}\Bigg[\gamma^H V_{sim}(s_H) - \gamma^{H-1} V_{sim}(s_{H-1}) + \gamma^H r(s_{H-1})\Bigg] \geq 0. \tag{12}$$

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

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* π *which satisfies* $A^*_H(s,\pi) = Q^*_H(s,\pi) - V^*_H(s) \geq -\delta$ *will satisfy:*

$$V^*_{real}(s) - V^{\pi}_{real}(s) \leq \gamma^H \epsilon + \frac{\delta}{1-\gamma}. \tag{13}$$

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:

$$Q^*_H(s,\pi) + \delta \geq V^*_H(s) \tag{14}$$

$$V^*_H(s) \geq V^*_{H-1}(s) \tag{15}$$

$$Q^*_H(s,\pi) = \mathbb{E}[\gamma V^*_{H-1}(s') + \bar{r}(s,s')] \tag{16}$$

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:

$$\gamma\mathbb{E}\left[Q^*_H(s',\pi) + \bar{r}(s,s')\right] + \gamma\delta \geq \gamma\mathbb{E}\left[V^*_H(s') + \bar{r}(s,s')\right] + \geq \mathbb{E}\left[\gamma V^*_{H-1}(s') + \bar{r}(s,s')\right] = Q^*_H(s,\pi). \tag{17}$$

That is:

$$\gamma\mathbb{E}\left[Q^*_H(s',\pi)\right] + \bar{r}(s) + \gamma\delta \geq Q^*_H(s,\pi). \tag{18}$$

Alternatively:

$$\bar{r}(s) \geq Q^*_H(s,\pi) - \gamma\mathbb{E}[Q^*_H(s',\pi)] - \gamma\delta. \tag{19}$$

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

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

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

Thus, we have the lower-bound:

$$V^{\pi}_{real}(s) \geq Q^*_H(s,\pi) + V_{sim}(s_0) - \frac{\gamma\delta}{1-\gamma} \tag{20}$$

Next, we may bound:

$$\begin{aligned}
V^*_H(s_0) + V_{sim}(s_0) &\geq \mathbb{E}_{\rho^{\pi^*_{real}}(s_0)}\left[\gamma^H V_{sim}(s_H) + \sum_{t=0}^{H-1}\gamma^t r(s_t)\right] \qquad (21)\\
&= \mathbb{E}_{\rho^{\pi^*_{real}}(s_0)}\left[\gamma^H V_{sim}(s_H) - \gamma^H V^*_{real}(s_H) + \gamma^H V^*_{real}(s_H) + \sum_{t=0}^{H-1}\gamma^t r(s_t)\right] \\
&= \mathbb{E}_{\rho^{\pi^*_{real}}(s_0)}\left[\gamma^H V_{sim}(s_H) - \gamma^H V^*_{real}(s_H)\right] + V^*_{real}(s_0).
\end{aligned}$$

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

$$V^*_H(s_0) + V_{sim}(s_0) \geq V^*_{real}(s) - \gamma^H\epsilon. \tag{22}$$

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 (20) and Equation (21), we obtain that:

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

from which the state result follows immediately. □

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 γ. □

D Environment Overviews:

Hammering is a highly dynamic task involving force and contact dynamics that are impractical to precisely model in simulation. We construct such a task where 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 agent must hit the nail with high force repeatedly. The dynamics are inherently misspecified between sim and real here due to the infeasibility of precisely modeling the properties of the nail and its contact behavior with the hammer and board.

Insertion [74] 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 dynamics shift between sim and real, requiring real-world fine-tuning to reconcile.

E 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 velocites 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 $r(t) = -10 \cdot r_{\text{nail}-\text{goal}}(t)$ where $r_{\text{nail}-\text{goal}} = (\mathbf{r}_{\text{nail}})_z - (\mathbf{r}_{\text{goal}})_z$ 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. 2.

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 $r(t) = -r_{\text{ee}-\text{goal}}(t) - r_{\text{obj}-\text{goal}}(t) + r_{\text{threshold}}(t) - r_{\text{table}}(t)$ where $r_{\text{ee}-\text{goal}}(t) = \|\mathbf{r}_{\text{ee}}(t) - \mathbf{r}_{\text{obj}}(t) + [3.5\text{cm}, 0.0\text{cm}, 0.0\text{cm}]\|$ represents the distance of the end effector to the back of the puck, $r_{\text{obj}-\text{goal}}(t) = \|(\mathbf{r}_{\text{obj}}(t))_x - 55\text{cm}\|$ represents the distance of the puck to the goal (which is the edge of the table along the x dimension), $r_{\text{threshold}}(t) = \mathbb{I}[r_{\text{obj}-\text{goal}}(t) \geq 2.5\text{cm}]$ 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:

$$x_{\text{distance}}(t) = \text{clip}\left(|\mathbf{r}_{\text{pos1},x}(t) - \mathbf{r}_{\text{pos2},x}(t)|, 0.0, 0.1\right)$$
$$y_{\text{distance}}(t) = \text{clip}\left(|\mathbf{r}_{\text{pos1},z}(t) - \mathbf{r}_{\text{pos2},z}(t)|, 0.0, 0.1\right)$$
$$z_{\text{distance}}(t) = \text{clip}\left(|\mathbf{r}_{\text{pos1},y}(t) - \mathbf{r}_{\text{pos2},y}(t)|, 0.0, 0.1\right)$$

Let the success condition be defined as:

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

The reward function is now:

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

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.

F Implementation Details

Algorithm Details. We use SAC as our base off-policy RL algorithm for training in simulation and finetuning in the real world. For our method, we additionally add in two networks: a dynamics model which 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 μ_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 3×10^{-4}, batch size of 256, and discount factor $\gamma = .99$.

Finetuning 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 finetuning. During finetuning, 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 finetuning. We don't train on any simulation data during real-world finetuning because we empirically found it didn't help finetuning performance in our settings.

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]

G 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 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 finetuning the learned policy.

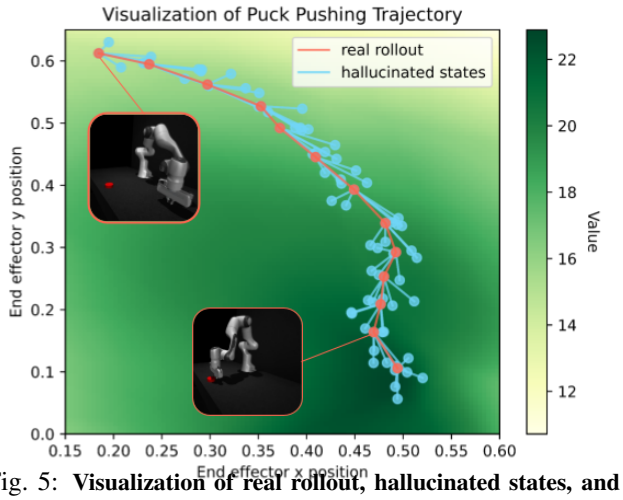

Fig. 5: **Visualization of real rollout, hallucinated states, and value function.** The red dots indicate states along a real rollout in simulation. The blue dots indicate hallucinated states branching off real states generated by the learned dynamics model. The green heatmap indicates the value function 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.

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.

H Sim-to-Sim Experiments

Here we additionally test each of the proposed methods on the sim-to-sim set-up from [75], 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 [75]. 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'.

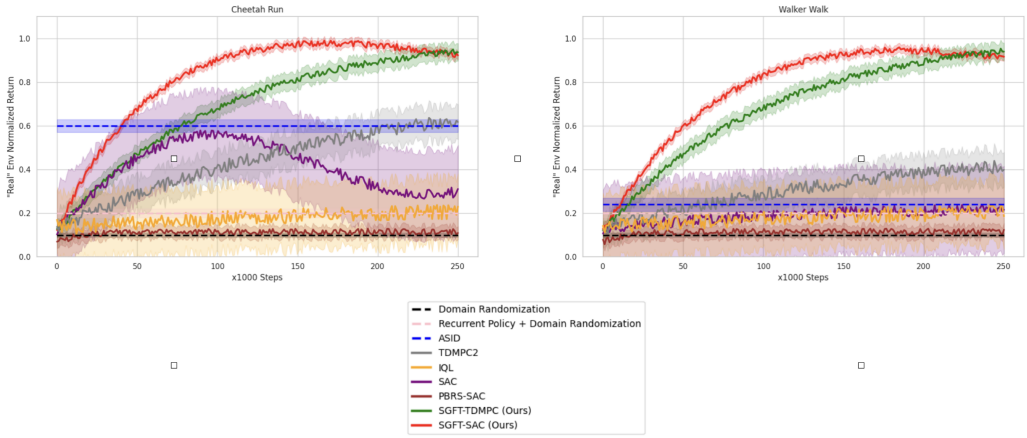

Fig. 6: **Normalized Rewards for Sim-to-Sim Transfer**. We plot the normalized rewards for two sim-to-sim transfer tasks, where the rewards are normalized by the maximum reward achieved by any method.