Abstract Sim2Real through Approximate Information States

Yunfu Deng
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53706
yunfu.deng@wisc.edu

Josiah Hanna
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53706
jphanna@cs.wisc.edu

Abstract

In recent years, reinforcement learning (RL) has shown remarkable success in robotics when a fast and accurate simulator is available for a given task. When using RL and simulation, more simulator realism is generally beneficial but becomes harder to obtain as robots are deployed in increasingly complex and widescale domains. In such settings, simulators will likely fail to model all relevant details of a given target task. In this paper, we formalize and study the abstract sim2real problem: given an abstract simulator that models a target task at a coarse level of abstraction, how can we train a policy with RL in the abstract simulator and successfully transfer it to the real-world? We formalize this problem using the language of state abstraction from the RL literature. This framing shows that an abstract simulator can be grounded to match the target task if the abstract dynamics take the history of states into account. Based on the formalism, we then introduce a method that uses a small amount of real-world task data and learns to correct the dynamics of the abstract simulator. We then show that these methods enable successful policy transfer both in sim2sim and sim2real evaluation.

1 Introduction

Reinforcement learning (RL) has demonstrated remarkable success across diverse application domains, from game playing Wurman et al. [2022] to robotic manipulation Andrychowicz et al. [2020], navigation Wijmans et al. [2019], and locomotion Hwangbo et al. [2019]. Despite these achievements, deploying RL in complex, real-world scenarios remains non-trivial due to a combination of expensive data collection, partial observability, and intricate physical dynamics. Simulators offer a safer and less costly alternative to real-world learning, but often rely on approximate physics engines that may omit crucial phenomena (e.g., latency) or introduce modeling artifacts Yoon et al. [2023]. Dynamics domain randomization Peng et al. [2018] aims to address the sim2real gap by robustifying agents to varied simulated conditions. However, in practice, it may be difficult to specify a complete set of randomizations that can fully capture the complexity of real systems, leaving policies vulnerable to unmodeled dynamics that result in poor performance when deployed.

To reduce the burden of identifying highly detailed simulators, researchers have turned to more extreme forms of simplified, abstracted simulation Truong et al. [2023]. More abstract simulators can significantly speed up experimentation and simplify the modeling process, making RL more accessible and efficient to develop Labiosa et al. [2025], Labiosa and Hanna [2025]. However, a highly simplified simulator runs the risk of ignoring essential dynamics that can arise in the real world, leading to policies that fail when transferred. An additional limitation of abstract simulators is partial observability: when more details of the environment are stripped away in the abstract simulator's state representation, the agent may not have direct access to critical factors that strongly influence performance in reality. If the learned policy is unaware of such factors, it will learn a policy that fails when transferred to the physical system.

With this motivation in mind, in this paper, we aim to answer the question:

“Can we use real-world data to modify an abstract simulator so that RL in the modified simulator produces a performant policy for a real robot?"

Toward answering this question, we first formalize the problem of abstract sim2real transfer using the notion of a state abstraction from the RL literature. Using this formalism, we identify that the key to grounding the abstract simulator and transferring performant policies is to learn simulator corrections and policies as functions of abstract state and action histories to mitigate the partial observability induced by abstraction. We leverage this insight to develop a new method, ASTRA (Augmented Simulation with self-predicTive abstRAction), that uses a small amount of real-world data to ground an abstract simulator. We validate this method through sim2real tasks with the humanoid NAO robot and sim2sim experiments in navigation and humanoid locomotion, where we show that it enables successful abstract sim2real transfer where other baselines fail.

2 Related Work

Sim2real transfer has been extensively studied in robotics Zhao et al. [2020]. While most work assumes high-fidelity simulators that share state spaces with target domains, we address transfer from *abstract* simulators with fundamentally different state representations. This distinction necessitates novel approaches because abstraction induces partial observability Allen et al. [2021]. We review work on abstraction in sim2real, simulator grounding methods, and reinforcement learning with state abstraction.

Abstraction and Sim2Real The sim2real research community has acknowledged the importance of developing the capability of robots that can learn with abstract simulators of the world Höfer et al. [2020]. In addition to the practical motivation that an abstract simulator is easier to specify, evidence from psychology suggests that humans seamlessly plan actions with abstract models Ho et al. [2022]. Several works have demonstrated the possibility and even advantages of abstract simulators compared to high-fidelity simulators. Nachum et al. showed that a high-level policy for robot navigation can be trained in a high-fidelity simulator but only using an abstract state representation as input Nachum et al. [2019]. Müller et al. show that abstraction can aid sim2real transfer in autonomous driving Müller et al. [2018]. Jain et al. improve learning of visual navigation policies by first training in an abstract grid simulator Jain et al. [2021]. Truong et al. found that reducing a simulator’s fidelity (switching from a dynamics to kinematic motion model) enables improved transfer of visual navigation policies, particularly when the wall-clock time of training is limited Truong et al. [2023]. However, these works use abstraction without formal analysis of its implications. Related to the idea of formalizing abstract sim2real, Cutler et al. formalize the notion of multi-fidelity simulators and develop a method for transferring knowledge between simulators Cutler et al. [2014], but focus on value approximation rather than state space mismatches. We provide the first formal treatment showing that abstraction induces partial observability, necessitating history-based grounding methods.

Simulator Grounding The methods we introduce are most closely related to existing methods for sim2real that ground a simulator’s dynamics to more closely match real-world dynamics. This class of methods includes system identification, by which the parameters of a simulator are tuned based on real-world experimental trials Åström and Eykhoff [1971], Armstrong [1987]. Many recent works in the sim2real literature have proposed using real-world data to learn corrections to a given simulator Ajay et al. [2018], Bousmalis et al. [2018], Golemo et al. [2018], Hanna et al. [2021], Heiden et al. [2021], Karnan et al. [2020]. These works focus on how to apply corrections and how to learn them with limited data. We extend these approaches to highly abstract simulators where ϕ induces partial observability.

Reinforcement Learning with Abstraction Our work uses the theory of state abstraction from the RL literature to formalize and identify methods for the abstract sim2real problem. While there is substantial work in state-abstractionAbel [2022], abstract sim2real is most closely related to the use of abstraction for model-based RL Jiang et al. [2015], Chaudhari et al. [2024]. Unlike these works, sim2real starts with a given simulator. State abstraction is known to induce partial observability when applied to states in an MDP Allen et al. [2021]. When states cannot be fully observed (POMDPs), integrating historical information becomes critical Littman and Sutton [2001], Hausknecht and Stone [2015]. Interestingly, memory can also improve performance in nominally fully observed MDPs, as shown in robotic manipulation Andrychowicz et al. [2020] and continuous control Patil et al. [2024]. Bisimulation defines state equivalence via reward and dynamics Givan et al. [2003]. Subramanian et al. Subramanian et al. [2022] proposed Approximate Information States (AIS), which unify reward and transition prediction to train a history encoder that preserves enough information for near-optimal decision making.

3 Preliminaries

We next formalize the RL objective and then formalize the standard sim2real transfer problem.

Reinforcement Learning We formalize an RL task as a Markov decision process (MDP), $\mathcal{M} = \big(\mathcal{S}, \mathcal{A}, P, r, \gamma\big)$, in which $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ the set of actions, $P \colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the transition dynamics, $r \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, and $\gamma \in [0, 1)$ the discount factor. We assume that states are fully observable (or can be reliably estimated) and that the transition dynamics are unknown. At each timestep i, the agent observes a state $s_i \in \mathcal{S}$, takes an action $a_i \in \mathcal{A}$, and then transitions to a new state s_{i+1} with probability $P(s_{i+1} \mid s_i, a_i)$. It also receives a scalar reward $r_i = r(s_i, a_i)$. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps states to action distributions. The agent's objective is to maximize the expected discounted return: $J_{\mathcal{M}}(\pi) = \mathbb{E}\Big[\sum_{i=0}^{\infty} \gamma^i \, r\big(s_i, a_i\big)\Big]$, where s_i and a_i are generated by following the policy π in $\mathcal{M}$.

Sim2Real We follow prior work in the literature by formalizing the sim2real problem as transfer between two MDPs with different state transition dynamics. Let the target domain (real world) be represented by the MDP, $\mathcal{M}_t := \big(\mathcal{S}, \mathcal{A}, P_t, r, \gamma\big)$, where $\mathcal{S}$ is the target domain state space, P_t describes the true transition dynamics, r is the reward function, and $\gamma \in [0, 1)$ is the discount factor. Let the source environment (simulator) be represented by the MDP, $\mathcal{M}_s = \big(\mathcal{S}, \mathcal{A}, P_s, r, \gamma\big)$, which shares the same state and action spaces with $\mathcal{M}_t$ but approximates real-world dynamics with the transition function, P_s. The sim2real problem is to find a policy, π, using RL in $\mathcal{M}_s$ that maximizes target environment performance, $J_{\mathcal{M}_t}(\pi)$. Under this formalism, the main challenge of sim2real is that $P_s \neq P_t$ and consequently, $J_{\mathcal{M}_s}(\pi) \neq J_{\mathcal{M}_t}(\pi)$. This so-called *reality gap* can cause well-trained policies in $\mathcal{M}_s$ to fail upon deployment in $\mathcal{M}_t$. However, because both MDPs share the same state and action spaces, one can, in principle, adjust or learn modifications to P_s so that it more closely matches P_t. In practice, such modifications are performed with either careful system identification (e.g., Tan et al. carefully calibrate the model of a quadrupedal robot Tan et al. [2018]) or through machine learning (e.g., Golemo et al. [2018]).

4 Abstract Sim2Real

In this section, we formalize the abstract sim2real problem using the theory of state abstraction from the RL literature. This formalization reveals that abstraction induces partial observability, necessitating history-based grounding methods that preserve task-relevant information.

We define an abstract simulator as an MDP over an abstract state-space, $\mathcal{M}_s := (\mathcal{S}^s, \mathcal{A}, P_s, r_s, \gamma)$, where the state space $\mathcal{S}^s$ is deliberately smaller than the real state space $\mathcal{S}^t$. Two examples of abstract state spaces are either compressing states in $\mathcal{S}^t$ to a finite set of discrete abstract states or to have $\mathcal{S}^s$ be a subspace of $\mathcal{S}^t$ (e.g., by dropping some dimensions of the real-world state). We assume the mapping from real-world states to abstract simulator states is known and define it as: $\phi : \mathcal{S}^t \to \mathcal{S}^s$. With slight abuse of notation, we also write $\phi : \mathcal{H}^t \to \mathcal{H}^s$ to denote applying ϕ to every state s^t in a trajectory of target environment states, $h^t \in \mathcal{H}^t$, so as to obtain a trajectory of abstract states, $h^s \in \mathcal{H}^s$. Note that the simulator's transition function, P_s, is necessarily simpler than P_t because ϕ merges or discards details that are present in $\mathcal{S}^t$. In this work, we will assume that the abstract simulator and real-world share the same action space, $\mathcal{A}$. If action spaces differ (e.g., high-level commands vs. low-level torques), we assume access to an action transformation function, like a PD controller or a learned low-level policy, to map actions from $\mathcal{M}_s$ to $\mathcal{M}_t$. As with standard sim2real, our objective is to use RL in $\mathcal{M}_s$ to learn a policy, π, that maximizes $J_{\mathcal{M}_t}(\pi)$.

Abstract sim2real raises two new challenges beyond the standard sim2real problem: the policy learned in the abstract simulator may lack crucial information for optimal control and the simplified transitions, P_s, cannot be modified in their present functional form to match P_t. The root of both challenges is that, in general, state abstractions induce partial observability Allen et al. [2021]. Perhaps the most obvious consequence of partial observability is that a policy, $\pi : \mathcal{S}^s \to \Delta(\mathcal{A})$, trained in the abstract simulator might be missing information that is otherwise available in states in $\mathcal{S}^t$. The second consequence is that we cannot directly apply simulator grounding methods using experience from $\mathcal{M}_t$. To see this, consider if we have a trajectory, $s^t_0, a_0, s^t_1, ..., s^t_T$, collected by running some policy in $\mathcal{M}_t$. Applying the abstraction, ϕ, to each state in this trajectory results in a trajectory, $s^s_0, a_0, s^s_1, ..., s^s_T$. However, the Markov property fails to hold in general meaning that $\Pr(s^s_{i+1}|s^s_i, a_i) \neq \Pr(s^s_{i+1}|s^s_i, a_i, s^s_{i-1}, a_{i-1})$ Allen et al. [2021]. Consequently, naively attempting to make $P_s(s^s_{i+1}|s^s_i, a_i) \approx \Pr(s^s_{i+1}|s^s_i, a_i)$ will fail to ground the dynamics of the abstract state to the real world.

Since abstraction induces partial observability, a straightforward approach is to extend existing neural-network-based grounding methods (e.g., Golemo et al. [2018], Hanna et al. [2021]) with recurrent networks to learn history-conditioned corrections and to train history-based policies. However, solely optimizing for next abstract state prediction has no guarantee of preserving all task-relevant information for control. This observation motivates the method that we introduce in the next section.

5 ASTRA: Augmented Simulation with Self-Predictive Abstraction

A natural approach to abstract sim2real is to use real-world data to align the abstract simulator's dynamics with the real-world's dynamics. A straightforward way to address the partial observability induced by abstraction is to extend neural correction methods (e.g., Hanna et al. [2021], Golemo et al. [2018]) with recurrent neural networks (RNNs) so that corrections can be based on full state-action histories. Under this extension, these approaches would learn a correction function $\psi : \mathcal{H}^s \times \mathcal{A} \times \mathcal{S}^s \to \mathcal{S}^s$ that predicts adjustments to the simulator's next state so that it matches the next abstract state observed in the real world. This approach optimizes the hidden representation of the RNN solely for prediction accuracy through an MSE loss, which may not necessarily lead to a hidden representation that contains all task relevant information. To explicitly shape the hidden state representation toward retaining task-relevant information, we introduce a new method, **ASTRA** (**A**ugmented **S**imulation with self-predic**T**ive abst**RA**ction). The key novelty of ASTRA is to augment simulator grounding with loss terms motivated from the literature on self-predictive state abstractions. Algorithm 1 provides pseudocode for the ASTRA training procedure; Algorithm 2 provides pseudocode for RL training with ASTRA.

5.1 Grounding with Self-Prediction Losses

ASTRA encodes history $h^s_i = (s^s_1, a_1, \dots, s^s_i, a_{i-1})$ with a recurrent encoder θ (e.g., GRU or LSTM), producing a latent vector $z^s_i = \theta(h^s_i) \in \mathcal{Z}$. To train this encoder, we use the paired trajectory collection method from Golemo et al. [2018], executing the same sequence of actions in both the abstract simulator and target environment, yielding corresponding state sequences $(s^s_1, s^s_2, \dots)$ and $(s^t_1, s^t_2, \dots)$ where $s^s_i = \phi(s^t_i)$. Our objective is to learn the encoder such that z^s_i contains sufficient information both for grounding the abstract simulator and learning an effective policy. Specifically, we seek a representation $z^s_i = \theta(h^s_i)$ such that (i) rewards are predictable from z^s_i via a learned reward predictor $\hat{r}$:

$$\mathbb{E}\big[r^t_i \,\big|\, h^s_i, a_i\big] \approx \hat{r}(z^s_i, a_i)\,,$$

and (ii) the next latent depends on the past only through z^s_i (Markov in latent space):

$$P\big(z^s_{i+1} \,\big|\, h^s_i, a_i\big) \approx P\big(z^s_{i+1} \,\big|\, z^s_i, a_i\big)\,.$$

We encourage θ to approximately satisfy these constraints via loss functions that are inspired from the concept of an *approximate information state* (AIS) Subramanian et al. [2022], Patil et al. [2024]. An AIS representation contains sufficient information to predict rewards in the real-world environment as well as the representation at the next time-step. AIS representations are closely related to self-predictive abstractions Guo et al. [2022], Schwarzer et al. [2021]. We will train the encoder used by ASTRA such that it produces an AIS hidden state representation.

Concretely, we augment encoder training with two losses, in addition to an MSE loss on the next abstract state. First, ASTRA learns a *transition model* $\hat{P}(z^s_i, a_i) = (\mu_i, \log \sigma^2_i)$ that outputs the parameters of a Gaussian $\mathcal{N}(\mu_i, \sigma^2_i)$ that approximates the distribution of z^s_{i+1}. We train both the transition model and encoder to minimize the negative log-likelihood $\mathcal{L}_{\text{trans}} = -\sum_i \log\big[\mathcal{N}\big(z^s_{i+1} \mid \mu_i, \sigma_i\big)\big]$, driving $\hat{P}$ to mirror real-world transitions in latent space. Second, we train a *reward model* $\hat{r}(z^s_i, a_i) \rightarrow \hat{r}_i$ that predicts the reward observed in the target environment, r^t_i; we jointly train the reward model and encoder to minimize the mean-squared error $\mathcal{L}_{rew} = \sum_i \|\hat{r}_i - r^t_i\|^2$. Third, as done with other neural grounding methods, ASTRA predicts the next abstract state: $\mathcal{L}_{\text{abs}} = \sum_i \big\| \hat{s}^s_{i+1} - \phi\big(s^t_{i+1}\big) \big\|^2$. ASTRA's total training objective is then $\mathcal{L} = \lambda_1 \mathcal{L}_{\text{trans}} + \lambda_2 \mathcal{L}_{\text{rew}} + \lambda_3 \mathcal{L}_{\text{abs}}$. In our implementation, θ, $\hat{P}$, $\hat{r}$ and f_{abs} share a common backbone with three task-specific heads; we optimize all parameters jointly under $\mathcal{L}$.

Algorithm 1 ASTRA Simulator Grounding

```
1:  for epoch = 1 to $N_{\text{phase1}}$ do
2:      Update $\theta, \hat{P}, \hat{r}$:
3:          Encode source states: $z^s \leftarrow \theta(h^s)$
4:          Predict latent transition: $(\mu, \log \sigma^2) \leftarrow \hat{P}(z^s, a)$
5:          Predict reward: $\hat{r} \leftarrow \hat{r}(z^s, a)$
6:          Predict next abstract state: $\hat{s}^s_{i+1} \leftarrow f_{\text{abs}}(z^s, a)$
7:          Compute losses: $\mathcal{L}_{\text{trans}} = -\log \mathcal{N}(z^s_{i+1} \mid \mu_i, \sigma^2_i)$, $\mathcal{L}_{\text{rew}} = \|\hat{r} - r^t\|^2$, $\mathcal{L}_{\text{abs}} = \|\hat{s}^s_{i+1} - \phi(s^t_{i+1})\|^2$
8:          Aggregate objective: $\mathcal{L} = \lambda_1 \mathcal{L}_{\text{trans}} + \lambda_2 \mathcal{L}_{\text{rew}} + \lambda_3 \mathcal{L}_{\text{abs}}$
9:          Update $(\theta, \hat{P}, \hat{r})$ with $\nabla \mathcal{L}$
10: end for
```

5.2 Target Environment Abstraction

ASTRA trains a policy that takes the learned AIS representation as input. Consequently, we need an encoder to produce these latent states when running the policy in the target environment. To do this, ASTRA abstracts target history h^t into the same latent space $\mathcal{Z}$. Specifically, we learn a target encoder $\zeta : \mathcal{H}^t \rightarrow \mathcal{Z}$ that maps target history into the same latent space, defining $z^t_i = \zeta(h^t_i)$. To ensure compatibility with the source encoder θ, ASTRA enforces that z^t_i and z^s_i have similar distributions for corresponding actions. Let $p^s_i := P(z^s_{i+1}|z^s_i, a_i)$ and $p^t_i := P(z^t_{i+1}|z^t_i, a_i)$, the alignment is achieved by minimizing $\mathcal{L}_{\text{align}} = D\left(p^s_i, p^t_i\right)$ where we use Maximum Mean Discrepancy (MMD) as D. For alignment we freeze $\theta, \hat{P}, \hat{r}$ and update only ζ; after alignment, ζ is kept fixed for deployment. The target encoder then enables deploying the policy trained in the grounded abstract simulator.

We note that both NAS and ASTRA use representations of history to implicitly infer relevant state features that are unmodelled in the abstract simulator. While this approach is a principled method of grounding abstract simulators, if the abstraction, ϕ, is too coarse, even using history may not enable accurate grounding. ASTRA's latent representation guides policy learning to make better use of the partial signals that are available in temporal data.

Algorithm 2 Policy Learning with Grounded Simulator

```
Require: RL algorithm $\mathbb{A}$
1:  Initialize $\mathbb{A}$
2:  for episode $= 1$ to $M$ do
3:      Initialize the abstract simulator states at $s^s_1$ and history $h^s_1$
4:      for $i = 1$ to $T$ do
5:          Encode history $h^s_i$ to latent state $z^s_i \leftarrow \theta(h^s_i)$
6:          Sample action $a_i \sim \pi(z^s_i)$
7:          Sample $z^s_{i+1} \sim \hat{P}(z^s_i, a_i)$
8:          Set $\hat{r}_i \leftarrow \hat{r}(z^s_i, a_i)$
9:          Execute action $a_i$ in the abstract simulator states
10:         Apply RL update with transition $(z^s_i, a_i, z^s_{i+1}, \hat{r}_i)$
11:         Set the abstract simulator to $s^s_{i+1} \leftarrow f_{\texttt{abs}}(z^s_i, a_i)$, and append to $h^s_{i+1}$
12:     end for
13: end for
```

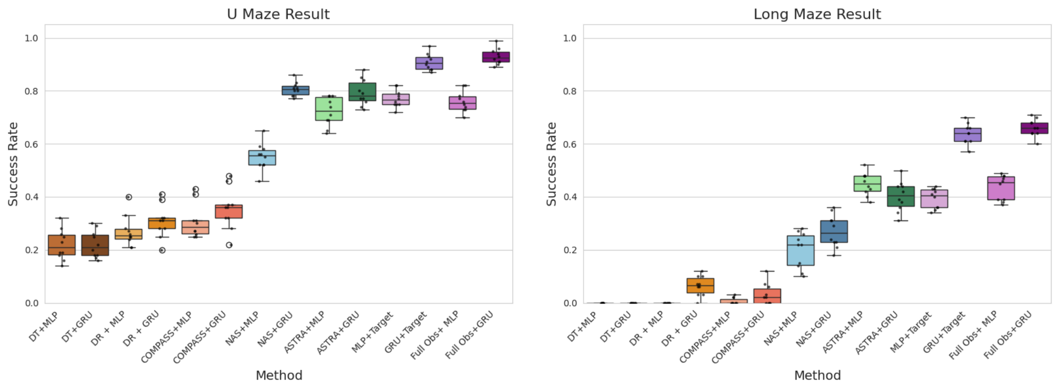

Figure 1: Success rates on navigation tasks.

6 Experiments

In this section, we empirically study abstract sim2real transfer to answer the following three questions: (i) Can history-based approaches with recurrent policies enable transfer of policies trained in abstract simulators? (ii) Does learning a self-predictive representation improve transfer efficacy compared to methods that only optimize prediction accuracy? (iii) How does the level of abstraction affect the relative importance of grounding methods versus domain randomization?

To address these questions, we evaluate transfer across navigation, humanoid locomotion, and real robot tasks with varying abstraction levels. We evaluate ASTRA against four baselines. Direct Transfer (DT) establishes the abstract sim2real gap. In our experiments, DR serves as a minimal domain randomization baseline using only action noise. COMPASS Huang et al. [2023] represents a state-of-the-art domain-randomization style method. NAS Golemo et al. [2018] serves as a strong and representative baseline for neural grounding methods that use real-world data and recurrent networks to learn history-based simulator corrections. While NAS optimizes solely for prediction accuracy, ASTRA incorporates self-predictive losses to preserve task-relevant information.

6.1 Legged Robot Navigation - Sim2Sim

Our abstract simulator uses a point-mass with velocity commands (v_x, v_y) and planar state (x, y, v_x, v_y), while the target AntMaze environment has a 29-dimensional state containing torso pose and joint angles/velocities. This abstraction gap is substantial: the simulator omits leg contacts, joint dynamics, and orientation drift that critically affect quadruped locomotion. A separately trained low-level controller, frozen during all experiments, converts high-level velocity commands to joint torques in the target environment.

We evaluate on two maze configurations: U-Maze with a single 90° turn and Long Maze with multiple turns. We measure success rates over 10 seeds with 100 trajectories each. Initial and goal positions are sampled from Gaussians around fixed poses. For training data, we collect 200 trajectories (average length 500 steps) using a random behavior policy π_0 to generate paired data between domains. For this domain, DR uses action noise ($\varepsilon \sim \mathcal{N}(0, 0.05)$) and scaling ($\delta \sim \mathcal{U}[-0.1, 0.1]$). COMPASS randomizes friction ($\mu \in [0.8, 1.2]$), position noise (± 0.03m), velocity noise (≤ 0.02m/s), heading bias (± 5), action parameters, and control timestep ($\Delta t \in [0.015, 0.025]$s). To upper bound performance, we include policies trained directly in AntMaze: *Target* uses the abstracted state $\phi(s^t)$ while *Full Obs* has access to the complete state.

Figures 1 reveal two key findings. First, GRU policies consistently outperform MLPs across all methods, confirming temporal memory mitigates abstraction-induced partial observability. Second, among GRU policies, ASTRA achieves the highest success rate. NAS, our representative baseline, ranks second, followed by COMPASS, DR, and DT. This performance gap widens in Long Maze.

6.2 Humanoid Locomotion

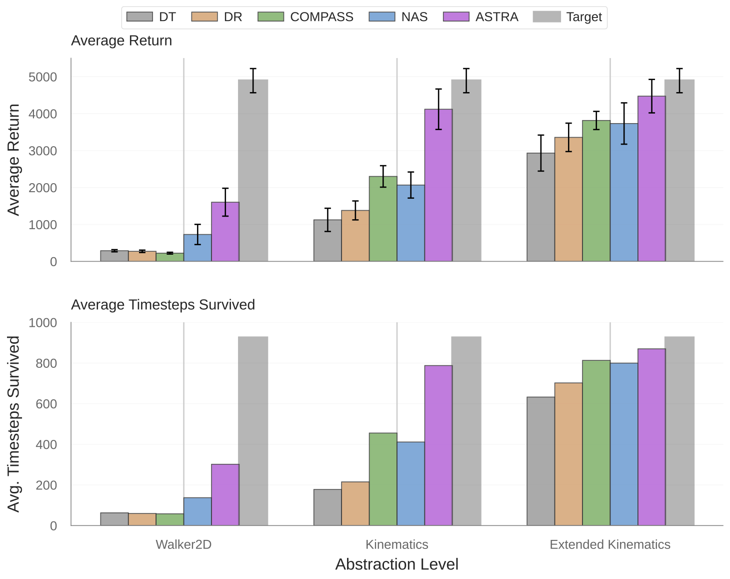

Figure 3: Humanoid locomotion results across three abstraction levels (10 seeds; higher is better).

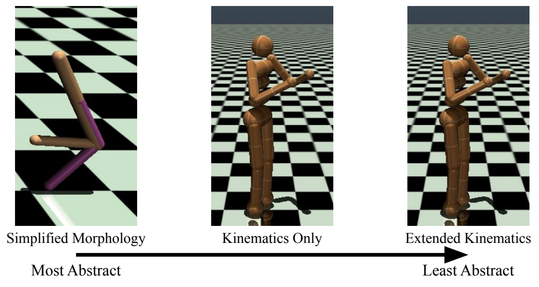

Figure 2: Abstraction hierarchy used for humanoid locomotion experiments: Walker2D, Kinematics, and Extended Kinematics.

We examine how abstraction level affects transfer by testing three variants of the high-DoF Humanoid [Duan et al., 2016], each representing different levels of state and morphological abstraction. All policies output joint position commands that a PD controller ($K_p = 200$, $K_d = 10$) converts to torques in the target environment.

Three abstraction levels systematically vary the simulator fidelity: (1) WALKER2D: The most abstract variant models the humanoid's upper body as a single rigid link, reducing observations to leg positions, joint angles, and foot contacts—omitting arm, torso, and center-of-mass (CoM) information. Only 4 joints are controllable (left/right knee and thigh), each operating around a single axis. (2) KINEMATICS: Preserves the full humanoid morphology with observations including positions and velocities of all joints. Actions are interpreted as desired joint positions rather than torques, maintaining complete body structure while abstracting force-level dynamics. (3) EXTENDED KINEMATICS: Augments kinematic abstraction with robot-level information including CoM and translational velocity [Radosavovic et al., 2024], capturing global dynamics while maintaining position control.

We collect 200 trajectories (average length 500 steps) using a suboptimal PPO policy, as random policies fail immediately and thus produce highly irrelevant data for bipedal locomotion. For humanoid domains, DR uses action noise ($\varepsilon \sim \mathcal{N}(0, 0.05)$) and scaling ($\delta \sim \mathcal{U}[-0.05, 0.05]$). COMPASS additionally randomizes joint friction ($\mu \in [0.8, 1.2]$), observation noise (scaling factor $\delta \sim \mathcal{U}[0.9, 1.1]$), and control timestep ($\Delta t \in [0.015, 0.025]$s).

Figure 3 reveals how abstraction level critically impacts transfer success. In the skeletal Walker2D setting, DT, DR, and COMPASS learn policies that terminate quickly (within 63 timesteps). NAS shows improvement through history-based simulator grounding. ASTRA achieves the best performance, maintaining balance longest Full-body KINEMATICS stabilizes all algorithms and narrows the reality gap, with COMPASS now outperforming NAS. EXTENDED KINEMATICS brings smaller gains, with NAS and COMPASS approaching the target-trained upper bound while ASTRA maintains its lead. These results demonstrate that (i) retaining essential information during abstraction is most effective for transfer, and (ii) when abstraction is severe, ASTRA's self-predictive grounding proves most effective

6.3 Real Robot Evaluation

Finally, we validate our approach on a physical NAO bipedal robot, testing transfer from highly abstract simulators to real hardware. The physical NAO presents unique challenges absent in simulation: imprecise odometry, foot slippage, actuator delays, and camera noise. We evaluate on two complementary tasks that stress different aspects of abstract sim2real transfer.

6.3.1 NAO Navigation

The abstract simulator models the robot as a 2D point mass with velocity control. On the real NAO, high-level velocity control is translated to joint commands through a walk engine and sensor readings are mapped to high-level pose estimates with a state-estimation module. The robot must navigate a physical maze to reach a 0.3-m radius goal zone without wall collisions. Runs are initialized from three distinct start poses per seed; episodes terminate after 500 control steps or upon completion. Performance metrics: success rate, distance traveled (m), and completion time (s) over three seeds.

We augment 50 collected trajectories to 200 through rotational and translational transformations. COMPASS randomizes ground friction $\mu \in \mathcal{U}[0.8, 1.2]$, foot slippage (20% probability), position noise ($\pm$0.03m), velocity noise ($\leq$ 0.02m/s), heading bias ($\pm$5), action noise ($\varepsilon \sim \mathcal{N}(0, 0.05)$, scaling $\delta \sim \mathcal{U}[-0.1, 0.1]$), and control timestep ($\Delta t \in \mathcal{U}[0.015, 0.025]$s).

Table 1 shows ASTRA achieves 73% success rate, significantly outperforming direct transfer (27%). COMPASS reaches 50% through extensive randomization, while NAS achieves 53% using history-based corrections. ASTRA significantly outperforms all baselines at 73%, demonstrating superior handling of unmodeled effects like foot slippage and odometry drift.

Table 1: NAO navigation results (3 seeds).

Method	Success Rate	Distance (m)	Time (s)
DT	0.27 ± 0.21	10.91	86.62
DR	0.33 ± 0.31	9.30	81.08
COMPASS	0.50 ± 0.08	12.70	85.37
NAS	0.53 ± 0.06	12.41	80.77
ASTRA	**0.73 ± 0.05**	12.33	82.54

6.3.2 NAO Ball-Kicking

The abstract simulator uses a 2D point agent with simplified ball physics. Real-world ball tracking uses monocular vision with noise and dropouts. The NAO must kick a ball into a goal within 30s. Each seed evaluates 20 trials with random starts, measuring success rate and completion time.

We collect 200 trajectories using a random policy. COMPASS randomizes ground friction $\mu \in \mathcal{U}[0.8, 1.2]$, ball position/velocity (simulating camera uncertainty), and post-contact ball direction (simulating foot-ball contact variations).

Method	Success Rate	Time (s)
DT	0.07 ± 0.03	13.2
DR	0.12 ± 0.04	17.4
NAS	0.37 ± 0.22	17.2
COMPASS	0.40 ± 0.25	19.2
ASTRA	**0.56 ± 0.05**	21.7

Table 2: Ball-kicking performance over 3 seeds.

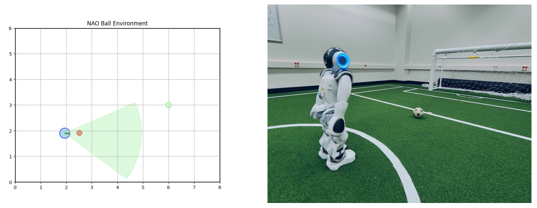

Figure 4: Left: abstract dribble simulator. Right: NAO kicking ball.

Table 2 shows ASTRA achieves 56% success rate, substantially outperforming all baselines. NAS (37%) shows improvement over DT and DR through history processing, but still falls short of ASTRA's performance in handling camera noise and contact uncertainty.

6.4 Data Efficiency Analysis

The amount of real-world data required for effective simulator grounding directly impacts practical deployment. Therefore, this analysis examines how performance scales with dataset size using navigation in sim2sim as a representative task. We evaluate ASTRA's data efficiency compared to the strongest baseline (NAS) across six dataset sizes: 25%, 50%, 75%, 100%, 125%, and 150% of a baseline dataset containing 200 trajectories. Each configuration is evaluated through downstream

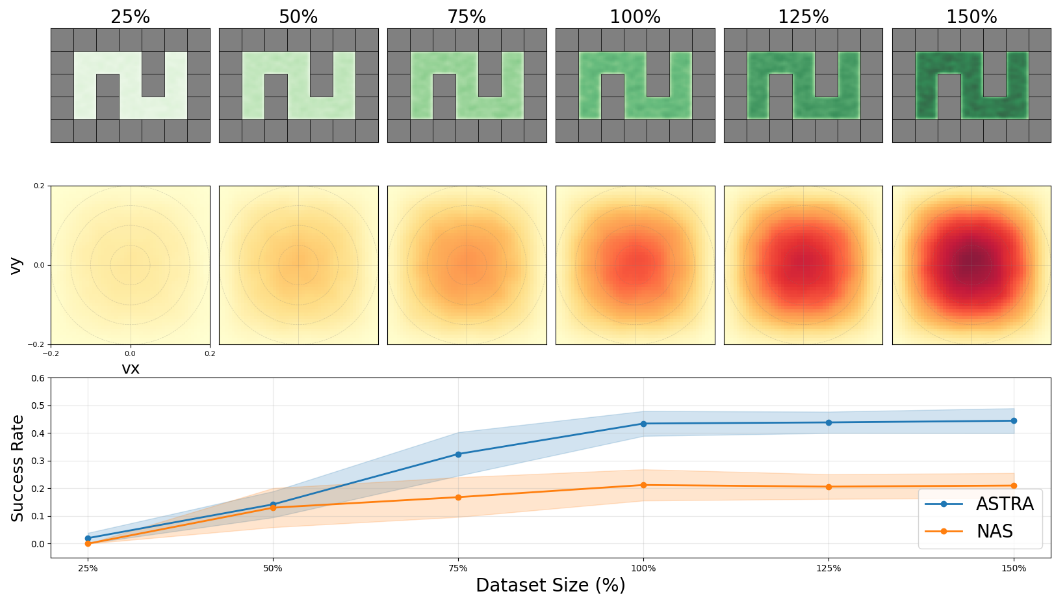

Figure 5: Dataset efficiency analysis. Top: position coverage, middle: velocity distribution, bottom: transfer performance. Shaded regions indicate performance variability across 5 seeds.

RL success rate with MLP policy networks across 5 independent seeds. Figure 5 shows the position coverage (top), velocity coverage (middle), and transfer performance (bottom) as dataset size increases. The results demonstrate clear diminishing returns in data collection for simulator grounding. ASTRA achieves its steepest improvement between 25% and 75% of the baseline dataset, with performance plateauing beyond 100%. This trend also holds for NAS, though at lower absolute performance. Notably, doubling the dataset from 75% (150 trajectories) to 150% (300 trajectories) yields less than 10% improvement in success rate. The variance patterns indicate that performance stability emerges around 100% data, suggesting this represents sufficient coverage of the state space.

7 Conclusion

In this paper, we studied the question of how to enable a robot to use RL in a so-called abstract simulator and the resulting policy transfer to the real-world. We first formalized the abstract sim2real problem which highlighted the need to learn history-based policies and to consider histories of abstract state sequences when grounding the dynamics of an abstract simulator to the real-world. We then introduced a novel method, ASTRA, to learn correction functions for abstract simulators. Finally, we showed in sim2sim and sim2real experiments that this method enables policies trained in abstract simulators to effectively transfer to target domains.

8 Acknowledgment

This research was performed in the Prediction and Action Lab (PAL) at the University of Wisconsin — Madison. PAL research is supported by NSF (IIS-2410981), the Wisconsin Alumni Research Foundation, and Sandia National Labs through a University Partnership Award. The authors would also like to thank Yuhao Li for his assistance with the real-robot experiments.

References

D. Abel. A Theory of Abstraction in Reinforcement Learning, Mar. 2022. URL `http://arxiv.org/abs/2203.00397`. arXiv:2203.00397 [cs].

A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing.

In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3066–3073. IEEE, 2018.

C. Allen, N. Parikh, O. Gottesman, and G. Konidaris. Learning Markov State Abstractions for Deep Reinforcement Learning. 2021.

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.

B. Armstrong. On finding 'exciting' trajectories for identification experiments involving systems with non-linear dynamics. In *1987 IEEE International Conference on Robotics and Automation Proceedings*, volume 4, pages 1131–1139, Mar. 1987. doi: 10.1109/ROBOT.1987.1087968. URL `https://ieeexplore.ieee.org/document/1087968`.

K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4243–4250. IEEE, 2018.

S. Chaudhari, A. Deshpande, B. C. d. Silva, and P. S. Thomas. Abstract Reward Processes: Leveraging State Abstraction for Consistent Off-Policy Evaluation, Oct. 2024. URL `http://arxiv.org/abs/2410.02172`. arXiv:2410.02172 [cs].

M. Cutler, T. J. Walsh, and J. P. How. Reinforcement learning with multi-fidelity simulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3888–3895. IEEE, 2014.

Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial intelligence*, 147(1-2):163–223, 2003.

F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer. Sim-to-real transfer with neural-augmented robot simulation. In *Conference on Robot Learning*, pages 817–828. PMLR, 2018.

Z. D. Guo, S. Thakoor, M. Pîslar, B. A. Pires, F. Altché, C. Tallec, A. Saade, D. Calandriello, J.-B. Grill, Y. Tang, M. Valko, R. Munos, M. G. Azar, and B. Piot. BYOL-Explore: Exploration by Bootstrapped Prediction, June 2022. URL `http://arxiv.org/abs/2206.08332`. arXiv:2206.08332 [cs, stat].

J. P. Hanna, S. Desai, H. Karnan, G. Warnell, and P. Stone. Grounded action transformation for sim-to-real reinforcement learning. *Machine Learning*, 110(9):2469–2499, 2021.

M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9474–9481. IEEE, 2021.

M. K. Ho, D. Abel, C. G. Correa, M. L. Littman, J. D. Cohen, and T. L. Griffiths. People construct simplified mental representations to plan. *Nature*, 606(7912):129–136, June 2022. ISSN 1476-4687. doi: 10.1038/s41586-022-04743-9. URL `https://www.nature.com/articles/s41586-022-04743-9`. Number: 7912 Publisher: Nature Publishing Group.

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 *Conference on Robot Learning*, pages 734–760. PMLR, 2023.

J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, F. Golemo, M. Mozifian, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. K. Liu, J. Peters, S. Song, P. Welinder, and M. White. Perspectives on Sim2Real Transfer for Robotics: A Summary of the R:SS 2020 Workshop, Dec. 2020. URL `http://arxiv.org/abs/2012.03806`. arXiv:2012.03806 [cs].

U. Jain, I.-J. Liu, S. Lazebnik, A. Kembhavi, L. Weihs, and A. G. Schwing. Gridtopix: Training embodied agents with minimal supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15141–15151, 2021.

N. Jiang, A. Kulesza, and S. Singh. Abstraction Selection in Model-based Reinforcement Learning. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 179–188. PMLR, June 2015. URL `https://proceedings.mlr.press/v37/jiang15.html`. ISSN: 1938-7228.

H. Karnan, S. Desai, J. P. Hanna, G. Warnell, and P. Stone. Reinforced Grounded Action Transformation for Sim-to-Real Transfer. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4397–4402, Las Vegas, NV, USA, Oct. 2020. IEEE. ISBN 978-1-72816-212-6. doi: 10.1109/IROS45743.2020.9341149. URL `https://ieeexplore.ieee.org/document/9341149/`.

A. Labiosa and J. P. Hanna. Multi-robot collaboration through reinforcement learning and abstract simulation. *arXiv preprint arXiv:2503.05092*, 2025.

A. Labiosa, Z. Wang, S. Agarwal, W. Cong, G. Hemkumar, A. N. Harish, B. Hong, J. Kelle, C. Li, Y. Li, et al. Reinforcement learning within the classical robotics stack: A case study in robot soccer. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14999–15006. IEEE, 2025.

M. Littman and R. S. Sutton. Predictive representations of state. *Advances in neural information processing systems*, 14, 2001.

M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018.

O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224*, 2019.

G. Patil, A. Mahajan, and D. Precup. On learning history-based policies for controlling markov decision processes. In *International Conference on Artificial Intelligence and Statistics*, pages 3511–3519. PMLR, 2024.

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.

I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-Efficient Reinforcement Learning with Self-Predictive Representations, May 2021. URL `http://arxiv.org/abs/2007.05929`. arXiv:2007.05929 [cs, stat].

J. Subramanian, A. Sinha, R. Seraj, and A. Mahajan. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *Journal of Machine Learning Research*, 23(12):1–83, 2022.

J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. In *Proceedings of Robotics: Science and Systems*, 2018. URL `http://arxiv.org/abs/1804.10332`.

J. Truong, M. Rudolph, N. H. Yokoyama, S. Chernova, D. Batra, and A. Rai. Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation. In *Conference on Robot Learning*, pages 859–870. PMLR, 2023.

E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

J. Yoon, B. Son, and D. Lee. Comparative study of physics engines for robot simulation with mechanical interaction. *Applied Sciences*, 13(2):680, 2023.

W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, Dec. 2020. doi: 10.1109/SSCI47803.2020.9308468.

K. J. Åström and P. Eykhoff. System identification—A survey. *Automatica*, 7(2):123–162, Mar. 1971. ISSN 0005-1098. doi: 10.1016/0005-1098(71)90059-8. URL `https://www.sciencedirect.com/science/article/pii/0005109871900598`.