Abstract Sim2Real through Approximate Information States

Yunfu Deng, Yuhao Li, and Josiah P. Hanna

***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 and this observation motivates the study of sim2real with simulators that leave out key task details. 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? Our first contribution is to 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 grounded abstract dynamics take the history of states into account. Based on the formalism, we then introduce a method that uses real-world task data to correct the dynamics of the abstract simulator. We then show that this method enables successful policy transfer both in sim2sim and sim2real evaluation.**

***Index Terms*—Sim-to-real transfer, reinforcement learning, state abstraction, robot learning.**

I. Introduction

REINFORCEMENT learning (RL) has demonstrated remarkable success across diverse application domains, from game playing [1] to robotic manipulation [2], navigation [3], and locomotion [4]. 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 standard sim2real methods—including domain randomization [5], system identification, and learned dynamics corrections—assume that the simulator and the target domain share the same state-action space and differ only in dynamics parameters [6]. These methods address parametric mismatch but may not apply when the simulator operates over a different and more abstract state representation than the target robot—a common scenario when constructing a high-fidelity simulator is impractical. Noorani et al. [7] argue that reliance on high-fidelity simulation leads to overfitting to simulator-specific dynamics and that abstract, lower-fidelity simulators may instead enable more generalizable autonomy.

Motivated by these observations, researchers have turned to understanding how to use much more simplified and abstract simulators for sim2real [8]–[11]. More abstract simulators can significantly speed up experimentation and simplify the modeling process, making RL more accessible and efficient to develop [8], [10]. However, a highly simplified simulator runs the risk of ignoring essential task-relevant dynamics in the real world, leading to policies that fail when transferred. Furthermore, the abstract sim2real problem has yet to be mathematically formalized in the literature, and this gap hinders the development of theoretically grounded abstract sim2real methods.

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 for effective sim2real transfer of RL-trained behaviors for a real robot?"

Toward answering this question, we provide the first (to the best of our knowledge) formalization of the abstract sim2real problem using the notion of a state abstraction from the RL literature. Using this formalism, we identify that the key to abstract simulator grounding for successful policy transfer is to learn simulator corrections and policies as functions of abstract state and action histories to mitigate the effect of partial observability induced by abstraction. History-based grounding enables the grounded abstract simulator to implicitly account for unmodelled real-world dynamics. 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 (including a strong domain randomization approach) [12] fail.

II. Related Work

Sim2real transfer has been extensively studied in robotics; see Zhao et al. [13] for a more complete survey. In this section, we focus on the most closely related literature on abstraction in sim2real, simulator grounding methods, and state abstraction in RL.

A. Abstraction and Sim2Real

The sim2real research community has acknowledged the importance of developing the use of capability of robots that

Manuscript received: September 16, 2025; Revised: January 29, 2026; Accepted: March 25, 2026.

This paper was recommended for publication by Editor Aniket Bera upon evaluation of the Associate Editor and Reviewers' comments.This work took place in the Prediction and Action Lab (PAL) at the University of Wisconsin – Madison. PAL research is supported by NSF (IIS-2410981) and the Wisconsin Alumni Research Foundation.

Yunfu Deng and Josiah P. Hanna are with the Department of Computer Sciences, University of Wisconsin–Madison, Madison, WI 53706 USA (e-mail: yunfu.deng@wisc.edu, jphanna@cs.wisc.edu).

Yuhao Li performed this work while at the University of Wisconsin–Madison. He is now with the Manning College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA 01003 USA (e-mail: yuhaoli@umass.edu).

Digital Object Identifier (DOI): see top of this page.

can learn with abstract simulators of the world [7], [9]. 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 [14], raising the question of how robots might also base their actions on abstract models. 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 [15]. Müller et al. show that abstraction can aid sim2real transfer in autonomous driving [16]. Jain et al. improve learning of visual navigation policies by first training in an abstract grid simulator [17]. 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 [8]. Our work differs from these studies in that we provide the first formal description of the abstract sim2real problem and then develop a new abstract simulator grounding method based on this formalism. Cutler et al. also formalize the notion of multi-fidelity simulators and develop a method for transferring knowledge between simulators [18], but focus on value approximation rather than state space mismatches. In contrast, our formal model shows that abstraction induces partial observability, necessitating history-based grounding methods.

B. 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 [19], [20]. Many recent works in the sim2real literature have proposed using real-world data to learn corrections to a given simulator [21]–[24]. These works focus on how to apply corrections and how to learn them with limited data. We focus on similar challenges but differ from prior work by focusing on these challenges in the context of highly abstract simulators where the state abstraction ϕ induces partial observability that standard grounding methods cannot address.

C. 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-abstraction (see Abel [25] for a survey), abstract sim2real is most closely related to the use of abstraction for model-based RL [26], [27]. The key difference between these works and abstract sim2real is that sim2real starts with a given model (i.e., simulator) that is specified with domain knowledge. State abstraction is known to induce partial observability when applied to states in an MDP [28]. When states cannot be fully observed (i.e., in POMDPs), integrating historical information becomes critical [29], [30].

III. Abstract Sim2Real

In this section, we formalize the abstract sim2real problem. After establishing notation for standard sim2real, we define the abstract sim2real setting where the simulator's state representation is reduced (i.e., abstracted) from the full real-world state. We show that this difference fundamentally changes the transfer problem: abstraction induces partial observability, necessitating history-based methods for simulator grounding.

A. Preliminaries

a) 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. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps states to action distributions, with objective to maximize expected discounted return: $J_{\mathcal{M}}(\pi) = \mathbb{E}\Big[\sum_{i=0}^{\infty} \gamma^i \, r\big(s_i, a_i\big)\Big]$.

b) Sim2Real: Let the target domain (real world) be represented by the MDP $\mathcal{M}_t := \big(\mathcal{S}, \mathcal{A}, P_t, r, \gamma\big)$, and the source domain (simulator) by $\mathcal{M}_s = \big(\mathcal{S}, \mathcal{A}, P_s, r, \gamma\big)$. The sim2real problem is to find a policy π using RL in $\mathcal{M}_s$ that maximizes $J_{\mathcal{M}_t}(\pi)$. The main challenge is that $P_s \neq P_t$ and consequently a policy that maximizes $J_{\mathcal{M}_s}$ may fail w.r.t. $J_{\mathcal{M}_t}$. Most sim2real work explicitly or implicitly focuses on what we call *parametric mismatch*: the source and target domain dynamics have approximately similar structure but differ in parameters. With parametric mismatch, the reality gap can be addressed through system identification [31], simple, learned dynamics corrections [23], or domain randomization [5].

B. Problem Formulation

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 reduced from 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: $\phi : \mathcal{S}^t \to \mathcal{S}^s$.[1] 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 structurally different from 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, such as 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 $J_{\mathcal{M}_t}(\pi)$.

[1]This assumption is mild in many robotic systems where high-level semantic state variables are extracted from existing perception and state-estimation pipelines.

Having formalized abstract sim2real, we observe that it raises two primary challenges beyond the standard sim2real problem, both rooted in the fact that state abstractions generally induce partial observability [28]. First, a policy $\pi : \mathcal{S}^s \to \Delta(\mathcal{A})$ trained in the abstract simulator might be missing information critical for optimal control that is otherwise available in $\mathcal{S}^t$. For example, consider a 2D abstract simulator with state-space $\mathcal{S}^s = [x, y, v_x, v_y]$ that models a NAO bipedal robot with state $\mathcal{S}^t \in \mathbb{R}^{30}$ including joint configurations. The abstract state cannot distinguish between a biped with stable stance ready to navigate and one that has become unsteady due to a recent rapid change in direction. Both configurations map to identical values of (x, y, v_x, v_y), yet they require fundamentally different actions.

The second consequence of the partial observability induced by abstraction is that we cannot directly apply simulator grounding methods using experience from $\mathcal{M}_t$. To see this, consider if we have a trajectory, $s_0^t, a_0, s_1^t, ..., s_T^t$, collected by running some policy in $\mathcal{M}_t$. Applying the abstraction, ϕ, to each state in this trajectory results in a trajectory, $s_0^s, a_0, s_1^s, ..., s_T^s$. However, the Markov property, in general, fails to hold in terms of abstract states meaning that $\Pr(s_{i+1}^s | s_i^s, a_i) \neq \Pr(s_{i+1}^s | s_i^s, a_i, s_{i-1}^s, a_{i-1})$ in the target domain [28]. This violation makes the standard grounding objective of trying to make the Markov source domain transitions fit observed target domain data ill-defined, as multiple real states with different dynamics can map to the same abstract state. Consequently, naively attempting to correct $P_s(s_{i+1}^s | s_i^s, a_i)$ to approximate $\Pr(s_{i+1}^s | s_i^s, a_i)$ in the target domain will fail to ground the dynamics of the abstract state to the real world.

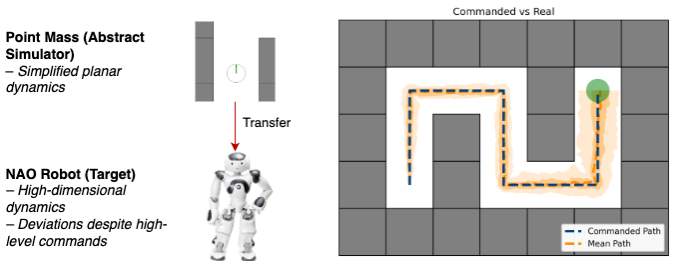

Fig. 1. Identical velocity commands produce perfect tracking in point-mass simulation (blue) but significant deviations on the physical robot (orange: mean trajectory; shaded: 25-75% and 5-95% quantiles) due to unmodeled dynamics.

IV. 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., [22], [23]) with recurrent neural networks (RNNs) so that corrections can be based on full state-action histories. Under this extension, these approaches would learn history-conditioned corrections that transform the simulator's predictions s_{i+1}^s to better approximate the true abstract next state $\phi(s_{i+1}^t)$ 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 presents the training procedure; Algorithm 2 shows policy learning with the grounded simulator. Notably, ASTRA is agnostic to the choice of RL algorithm; we use PPO for all navigation tasks and NAO ball-kicking, SAC for humanoid locomotion and Ant low-level controller training.

A. Grounding with Self-Prediction Losses

ASTRA augments the abstract simulator with a learned latent dynamics model that preserves task-relevant information through self-predictive constraints. The approach consists of three neural components that share a common recurrent backbone:

- **Source history encoder** $\psi^s : \mathcal{H}^s \to \mathcal{Z}$: Maps abstract state-action histories to a latent space $\mathcal{Z}$.
- **Latent dynamics model** $P_{\text{lat}} : \mathcal{Z} \times \mathcal{A} \to \Delta(\mathcal{Z})$: Predicts a distribution over next latent states given the current latent state and action.
- **Reward predictor** $R_{\text{pred}} : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$: Estimates rewards from the target environment.
- **Abstract state predictor** $f_{\text{abs}} : \mathcal{Z} \times \mathcal{A} \times \mathcal{S}^s \to \tilde{\mathcal{S}}^s$: Predicts next abstract states for simulator grounding.

Given an abstracted history $\Phi(h_i^t)$, the encoder produces a latent representation $z_i^s = \psi^s(\Phi(h_i^t)) \in \mathcal{Z}$. This encoding serves as the state representation for both simulator grounding and policy learning. To train these components, we collect paired trajectories following the protocol from Golemo et al. [22]: for each transition (s_i^t, a_i, s_{i+1}^t) collected from the target environment, we reset the abstract simulator to state $\phi(s_i^t)$ and execute the same action a_i to obtain the simulator's prediction. For this paired data, we define the history abstraction operator: $\Phi : \mathcal{H}^t \to \mathcal{H}^s$, where $\Phi(h_i^t) = (\phi(s_1^t), a_1, \phi(s_2^t), a_2, \ldots, \phi(s_i^t), a_{i-1})$ applies the abstraction function ϕ (from Section III) to each state in the target history. This operator constructs abstract histories that reflect real-world dynamics while maintaining the abstract state representation. Our objective is to learn the encoder such that z_i^s contains sufficient information both for grounding the abstract simulator and learning an effective policy. Specifically, we seek a representation $z_i^s = \psi^s(\Phi(h_i^t))$ such that: (i) rewards are predictable from z_i^s, i.e., $\mathbb{E}[r_i^t \mid \Phi(h_i^t), a_i] \approx R_{\text{pred}}(z_i^s, a_i)$, and (ii) the next latent depends on the past only through z_i^s (a Markov latent space), i.e., $P(z_{i+1}^s \mid \Phi(h_i^t), a_i) \approx P(z_{i+1}^s \mid z_i^s, a_i)$. We encourage ψ^s to approximately satisfy these constraints via loss functions inspired by the concept of an *approximate information state* (AIS) [32], [33]. 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

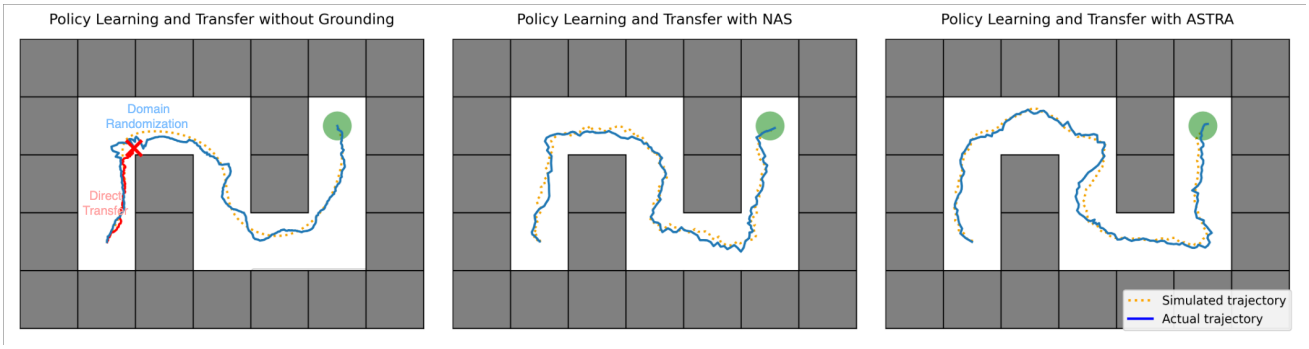

Fig. 2. Illustrations of trajectories learned by different approaches when transferring from PointMaze (abstract point-mass) to AntMaze (quadruped locomotion). Orange dashed lines indicate trajectories generated in the abstract point-mass simulator; blue solid lines show deployment in the target AntMaze environment. **Left:** Without grounding, policies exploit the simplified dynamics to learn aggressive, near-optimal paths in simulation. Direct transfer (red) results in immediate collision at corners because the dynamics of quadrupedal locomotion differ from idealized point-mass movement. Domain randomization (blue) reaches the goal but fails to track the simulated trajectory. **Center:** Incorporating history into grounding (using NAS [22]) enables the PointMaze simulator to implicitly account for the dynamics of quadrupedal locomotion, resulting in policies that learn safer navigation patterns. **Right:** ASTRA learns policies that identify and navigate high-risk regions where abstract and real dynamics diverge most, resulting in higher success rate.

to self-predictive abstractions [34], [35]. We will train the encoder used by ASTRA such that it produces an AIS hidden state representation.

Concretely, we optimize three complementary losses that correspond to the predictive components defined above. First, the latent dynamics model, P_{lat}, outputs parameters $(\mu_i, \log \sigma_i^2)$ of a Gaussian $\mathcal{N}(\mu_i, \sigma_i^2)$ that approximates the distribution of z_{i+1}^s. We train both P_{lat} and encoder ψ^s to minimize the negative log-likelihood $\mathcal{L}_{\text{trans}} = -\sum_i \log \mathcal{N}(z_{i+1}^s \mid \mu_i, \sigma_i)$, where $z_{i+1}^s = \psi^s(\Phi(h_{i+1}^t))$. This drives P_{lat} to mirror real-world transitions in latent space. Second, the reward predictor R_{pred} estimates target environment rewards. We jointly train R_{pred} and ψ^s to minimize $\mathcal{L}_{\text{rew}} = \sum_i \|R_{\text{pred}}(z_i^s, a_i) - r_i^t\|^2$. Third, the abstract state predictor f_{abs} uses the latent state representation as an input to correct the state transition of the abstract simulator so that it more accurately predicts the next abstract state in the real world $\mathcal{L}_{\text{abs}} = \sum_i \|f_{\text{abs}}(z_i^s, a_i, s_{i+1}^s) - \phi(s_{i+1}^t)\|^2$. ASTRA's total training objective combines these losses: $\mathcal{L} = \lambda_1 \mathcal{L}_{\text{trans}} + \lambda_2 \mathcal{L}_{\text{rew}} + \lambda_3 \mathcal{L}_{\text{abs}}$. In our implementation, ψ^s, P_{lat}, R_{pred}, and f_{abs} share a common GRU backbone with three task-specific output heads; we optimize all parameters jointly under $\mathcal{L}$.

B. 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 learns a target encoder $\psi^t : \mathcal{H}^t \to \mathcal{Z}$ that maps target histories into the same latent space as ψ^s, defining $z_i^t = \psi^t(h_i^t)$. To ensure compatibility, ASTRA enforces that z_i^t and z_i^s have similar distributions for corresponding actions. Let $p_i^s := P(z_{i+1}^s | z_i^s, a_i)$ and $p_i^t := P(z_{i+1}^t | z_i^t, a_i)$. The alignment is achieved by minimizing: $\mathcal{L}_{\text{align}} = D(p_i^s, p_i^t)$ where we use Maximum Mean Discrepancy (MMD) as D. For alignment, we freeze ψ^s, P_{lat}, R_{pred} and update only ψ^t; after alignment, ψ^t is kept fixed for deployment. The target encoder then enables real-world deployment of the policy trained in the grounded abstract simulator.

Algorithm 1 ASTRA Simulator Grounding

for epoch = 1 to N_{epochs} **do**
  for trajectory $(s_1^t, a_1, r_1^t, \ldots, s_T^t)$ in dataset **do**
    for $i = 1$ to $T - 1$ **do**
      $h_i^t = (s_1^t, a_1, \ldots, s_i^t, a_{i-1})$
      $z_i^s \leftarrow \psi^s(\Phi(h_i^t))$
      $z_{i+1}^s \leftarrow \psi^s(\Phi(h_{i+1}^t))$
      $(\mu_i, \log \sigma_i^2) \leftarrow P_{\text{lat}}(z_i^s, a_i)$
      $\hat{r} \leftarrow R_{\text{pred}}(z_i^s, a_i)$
      Get abstract simulator's next state s_{i+1}^s from paired data
      $\hat{s}_{i+1}^s \leftarrow f_{\text{abs}}(z_i^s, a_i, s_{i+1}^s)$
      $\mathcal{L}_{\text{trans}} = -\log \mathcal{N}(z_{i+1}^s \mid \mu_i, \sigma_i^2)$
      $\mathcal{L}_{\text{rew}} = \|\hat{r} - r_i^t\|^2$
      $\mathcal{L}_{\text{abs}} = \|\hat{s}_{i+1}^s - \phi(s_{i+1}^t)\|^2$
    end for
    Update $(\psi^s, P_{\text{lat}}, R_{\text{pred}}, f_{\text{abs}})$ with gradient descent on $\mathcal{L}$
  end for
end for

Algorithm 2 Policy Learning with ASTRA-Grounded Simulator

Require: RL algorithm $\mathbb{A}$
Initialize $\mathbb{A}$
for episode = 1 to M **do**
  Initialize abstract simulator at s_1^s and $h_1^s = (s_1^s)$
  for $i = 1$ to T **do**
    $z_i^s \leftarrow \psi^s(h_i^s)$
    $a_i \sim \pi(z_i^s)$
    $z_{i+1}^s \sim P_{\text{lat}}(z_i^s, a_i)$
    $\hat{r}_i \leftarrow R_{\text{pred}}(z_i^s, a_i)$
    Execute a_i in abstract simulator
    $\hat{s}_{i+1}^s \leftarrow f_{\text{abs}}(z_i^s, a_i, s_{i+1}^s)$
    Set simulator state $s_{i+1}^s \leftarrow \hat{s}_{i+1}^s$
    $h_{i+1}^s \leftarrow (h_i^s, a_i, s_{i+1}^s)$
  end for
  Update policy π according to $\mathbb{A}$
end for

V. Experiments

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

To address these questions, we evaluate different sim2real methods in two real robot tasks and simulated navigation and humanoid locomotion with varying abstraction levels. We evaluate ASTRA and six baselines. Direct Transfer (DT) establishes the abstract sim2real gap. Domain Randomization (DR) is a basic application of randomization to the actions of the robot. COMPASS [12] represents a strong domain-randomization style method. Rapid Motor Adaptation (RMA) [36] uses history-based adaptation to infer environment physical parameters online, representing methods designed for parametric uncertainty (i.e. variations in known physical parameters). In order to assess the necessity of learning a self-predictive state representation, we use neural-augmented simulation (NAS) [22] as a representative neural grounding approach to sim2real. NAS grounds the simulator via a recurrent correction function that is trained solely to predict the next abstract state that would occur in the real-world. We also include IQL fine-tuning (DT+IQL) [37], which fine-tunes the DT policy on target-domain data to assess whether direct learning can match simulator grounding. In our simulated experiments, we also consider directly training in the target environment.

A. Legged Robot Navigation - Sim2Sim

Our first sim2sim experiment uses two variants of AntMaze [38] as the target environment and a 2D point-mass abstract simulator. The abstract state space is the location and velocity of the agent whereas the full AntMaze state space includes a 29-dimensional state containing torso pose and joint angles/velocities. To align the action-spaces, we use a separately trained low-level controller (frozen during experiments) to convert high-level velocity commands to joint torques in the target environment. The abstraction gap is substantial as the simulator omits leg contacts, joint dynamics, and orientation drift that affect quadruped locomotion.

We evaluate on two maze configurations: U-Maze requiring a single 90° turn and Long Maze with multiple turns. We evaluate methods with success rate over 100 evaluation trajectories and average performance over 10 seeds for each method. We sample initial and goal positions from Gaussians centered on fixed poses. For ASTRA and NAS, we collect 200 trajectories (average length 500 steps) using a random behavior policy π_0 to generate data for simulator grounding. 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 ($\leq$ 0.02m/s), heading bias ($\pm 5^\circ$), action parameters, and control timestep ($\Delta t \in [0.015, 0.025]$s). As upper bounds, we include policies trained directly in AntMaze: *Target* uses the abstracted state $\phi(s^t)$ while *Full Obs* has access to the complete state.

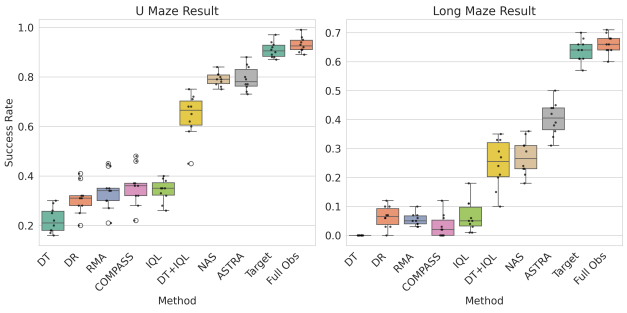

Fig. 3. Success rates on U-Maze (left) and Long Maze (right) navigation tasks. ASTRA achieves the highest success rate in both settings. RMA performs between DR and COMPASS, indicating that adaptation methods designed for parametric uncertainty cannot fully address abstraction gaps. Direct IQL training on target data yields limited performance, while simulator pretraining followed by IQL fine-tuning (DT+IQL) approaches NAS but with higher variance.

The results shown in Figure 3 provide affirmative answers to our empirical questions. Among all methods, ASTRA achieves the highest success rate in both maze configurations, demonstrating the effectiveness of self-predictive grounding for abstract sim2real. RMA's performance between DR and COMPASS confirms that parametric adaptation cannot bridge abstraction gaps. While direct IQL training struggles on complex mazes, DT+IQL fine-tuning substantially improves. ASTRA's consistent lead demonstrates that explicit dynamics grounding outperforms both parametric adaptation and direct fine-tuning. Furthermore, the learned grounding exhibits robustness to morphological variation: when evaluated on a modified Ant with leg segments scaled to $1.25\times$ their original length, the ASTRA policy trained on the standard Ant achieves 65% success rate on U-Maze, compared to 21% for direct transfer.

B. Humanoid Locomotion

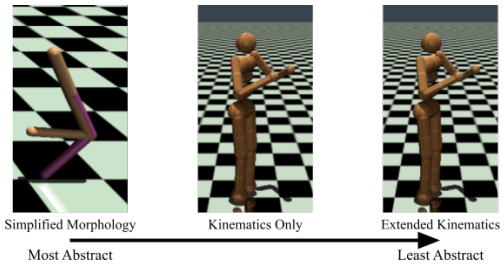

Fig. 4. Abstraction hierarchy used for humanoid locomotion experiments: Walker2D, Kinematics, and Extended Kinematics.

We next examine the influence of different levels of abstraction on abstract sim2real in simulated humanoid locomotion on the RL Humanoid benchmark [39]. We train policies that output joint position commands that a PD controller ($K_p = 200$, $K_d = 10$) converts to torques in the target environment. We use the following abstract simulators with different levels of

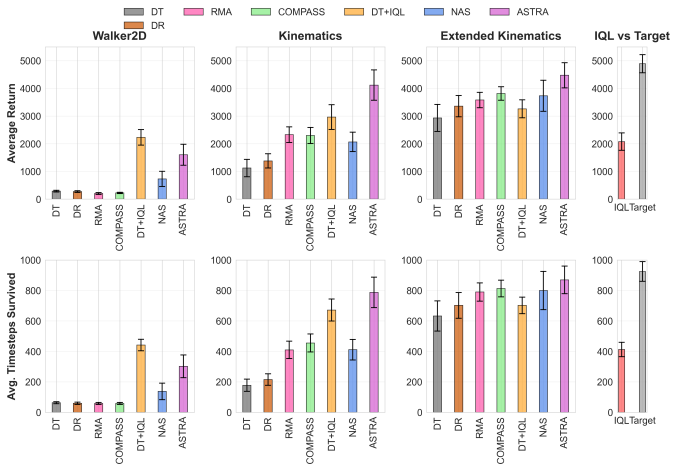

Fig. 5. Humanoid locomotion results across three abstraction levels (10 seeds; higher is better). Left three columns compare all methods; rightmost column compares direct IQL training on target data versus training in the target environment (*Target*).

abstraction: (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 while omitting arm, torso, and center-of-mass (CoM) information. The action space is 4-dimensional, controlling desired positions for four joints: left hip, left knee, right hip, and right knee, with each joint operating around a single axis. All other humanoid joints are fixed at constant values, effectively treated as rigid body components. A PD controller ($K_p = 200$, $K_d = 10$) converts the position commands to torques for these four active joints only. (2) KINEMATICS: Preserves the full humanoid morphology with observations including positions and velocities of all joints. The action space remains same as Walker2D, specifying desired positions for the same four joints (left hip, left knee, right hip, right knee), maintaining complete body structure while abstracting force-level dynamics. (3) EXTENDED KINEMATICS: Augments kinematic abstraction with robot-level information including CoM and translational velocity [40], while maintaining the same 4-dimensional position control action space. This extension enables the policy to access whole-body state information that facilitates transfer to the full humanoid target environment.

We collect 200 trajectories (average length 500 steps) using a suboptimal PPO policy, as random policies fail immediately and thus provide low-relevance data for grounding. All methods in this experiment use GRU policies. As an upper-bound reference, we also train a policy directly in the target environment using the complete Humanoid observation space [39] while maintaining the same 4-dimensional action space, labeled as *Target*. 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 5 reveals how abstraction level critically impacts transfer success. In the skeletal Walker2D setting, DT, DR, and COMPASS terminate quickly (within 63 timesteps). NAS shows improvement through history processing. ASTRA achieves the best performance, maintaining balance longest. DT+IQL fine-tuning achieves strong performance in this extreme case, approaching direct IQL training on target data (rightmost column). However, as simulator fidelity increases, fine-tuning gains diminish: in KINEMATICS and EXTENDED KINEMATICS, DT+IQL shows marginal improvement while ASTRA's advantage grows. 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.

C. Real Robot Evaluation

We validate our approach on a physical NAO bipedal robot, testing transfer from highly abstract simulators to real hardware. The 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.

1) NAO Navigation: The abstract simulator models the robot as a 2D point mass with velocity control. The real NAO executes commands through its walk engine, with observations from on-board localization including position, rotation, and velocity. 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 and distance traveled (m) over three seeds. Average distance is measured over successful trials, longer trajectories indicate that Agent learns more conservative, collision-avoiding behaviors.

For ASTRA and NAS, we augment 50 collected trajectories to 200 through rotational and translational transformations. For the DR baseline, we apply action noise ($\varepsilon \sim \mathcal{N}(0, 0.05)$) and scaling ($\delta \sim \mathcal{U}[-0.1, 0.1]$), consistent with the AntMaze configuration. COMPASS randomizes ground friction $\mu \in \mathcal{U}[0.8, 1.2]$, foot slippage (20% probability), position noise (± 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).

As shown in the navigation results of Table I, ASTRA achieves 73% success rate, significantly outperforming all baselines (17%–53%). For successful trials, we also measure travel distance as a metric for both efficiency and risk, where shorter distances may indicate aggressive corner-cutting. We find that most baselines achieve lower distances than ASTRA but have lower success rates, suggesting they attempt to take corners too tightly and frequently fail. Because ASTRA implicitly accounts for the NAO's low-level walking dynamics, it learns a more conservative policy that travels farther but succeeds more consistently.

2) NAO Ball-Kicking: The NAO must kick a ball into a goal within 30s. The abstract simulator models a 2D point agent and

simplified ball physics. Real-world ball tracking comes from the robot's on-board perception and state-estimation modules which produce noisy position estimates. For each seed, we measure success rate over 20 trials with random starts.

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

TABLE I
NAO ROBOT EVALUATION (MEAN ± STD, 3 SEEDS, 20 TRIALS EACH)

Method	Navigation Task		Ball-Kicking Task
	Success Rate	Distance (m)	Success Rate
DT	0.27 ± 0.21	10.91	0.07 ± 0.03
DR	0.33 ± 0.31	9.30	0.12 ± 0.04
RMA	0.51 ± 0.17	10.33	0.10 ± 0.01
DT+IQL	0.31 ± 0.10	10.72	0.08 ± 0.01
IQL	0.17 ± 0.14	9.77	0.05 ± 0.03
COMPASS	0.50 ± 0.08	12.70	0.40 ± 0.25
NAS	0.53 ± 0.06	12.41	0.37 ± 0.22
ASTRA	**0.73 ± 0.05**	12.33	**0.56 ± 0.05**

For the ball-kicking task of table I, right, 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.

D. Component Analysis

To isolate each training objective's contribution, we ablate ASTRA's loss components on the LONG MAZE navigation task: latent dynamics prediction ($\mathcal{L}_{\text{trans}}$), reward prediction ($\mathcal{L}_{\text{rew}}$), and abstract state correction ($\mathcal{L}_{\text{abs}}$). Since $\mathcal{L}_{\text{abs}}$ provides the essential grounding signal, we evaluate Full ASTRA, ASTRA without $\mathcal{L}_{\text{trans}}$, and ASTRA without $\mathcal{L}_{\text{rew}}$. Using $\mathcal{L}_{\text{abs}}$ alone is equivalent to NAS.

TABLE II
COMPONENT ABLATION (MEAN ± STD OVER 10 SEEDS)

Configuration	Success Rate
Full ASTRA	**0.40 ± 0.05**
w/o dynamics ($\mathcal{L}_{\text{trans}}$)	0.35 ± 0.04
w/o reward ($\mathcal{L}_{\text{rew}}$)	0.29 ± 0.10
NAS	0.27 ± 0.06

Table II shows that both auxiliary objectives contribute to ASTRA's performance, with reward prediction playing the dominant role. Removing $\mathcal{L}_{\text{rew}}$ causes a substantial drop (from 0.40 to 0.29) with doubled variance, indicating that task-relevant reward signals are critical for learning stable, control-oriented representations. Removing $\mathcal{L}_{\text{trans}}$ yields a smaller decrease (to 0.35) with stable variance, suggesting that self-predictive dynamics provide complementary regularization. The gap between Full ASTRA and NAS (0.40 vs 0.27) validates that both objectives contribute substantially beyond pure state prediction.

E. Data Efficiency Analysis

Finally, we examine how performance scales with dataset size using PointMaze-to-AntMaze transfer 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 RL success rate across 5 independent seeds on the Long-Maze task. Figure 6 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.

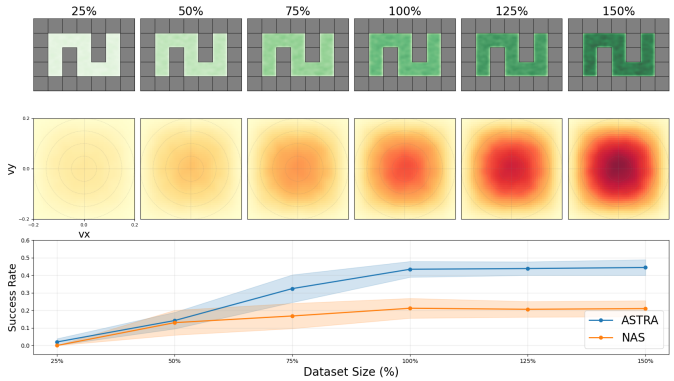

Fig. 6. Dataset efficiency analysis. Top: position coverage heatmap showing visited states. Middle: velocity distribution histogram. Bottom: transfer performance showing mean ± standard deviation across 5 seeds (solid line: mean, shaded region: ±1 std).

VI. CONCLUSION

In this paper, we formalized the abstract sim2real problem, which highlighted the need to learn history-based policies and grounding corrections that account for partial observability induced by abstraction. Based on this formalism, we introduced ASTRA, a method that learns a correction function for the abstract simulator to implicitly capture the impact of variables that were abstracted away. Both sim2sim and sim2real experiments demonstrate that ASTRA enables successful policy transfer from abstract simulators to target domains, with learned grounding showing robustness even under morphological variation of the target robot.

We note a few limitations as a basis for future work. History-based grounding may be insufficient if too much abstraction is applied; identifying acceptable abstraction levels is an interesting future direction. ASTRA requires a known state mapping, limiting applicability to higher-level variables rather than raw sensors, and grounding still requires non-trivial real-world data, though less than direct policy learning. Since all simulators are abstract to some degree [9], future work could provide a more unifying analysis beyond the more abstract end of the spectrum studied here.

REFERENCES

[1] 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*, vol. 602, no. 7896, pp. 223–228, 2022.

[2] 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*, vol. 39, no. 1, pp. 3–20, 2020.

[3] 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 Pre-print*, 2019.

[4] 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*, vol. 4, no. 26, p. eaau5872, 2019.

[5] 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)*. IEEE, 2018, pp. 3803–3810.

[6] J. Yoon, B. Son, and D. Lee, "Comparative study of physics engines for robot simulation with mechanical interaction," *Applied Sciences*, vol. 13, no. 2, p. 680, 2023.

[7] E. Noorani, Z. Serlin, B. Price, and A. Velasquez, "From abstraction to reality: Darpa's vision for robust sim-to-real autonomy," *AI Magazine*, vol. 46, no. 2, p. e70015, 2025.

[8] 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*. PMLR, 2023, pp. 859–870.

[9] 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, arXiv:2012.03806 [cs]. [Online]. Available: http://arxiv.org/abs/2012.03806

[10] A. Labiosa and J. P. Hanna, "Multi-Robot Collaboration through Reinforcement Learning and Abstract Simulation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2025.

[11] A. Labiosa, Z. Wang, Agarwal, Siddhant, W. Cong, G. Hemkumar, A. N. Harish, B. Hong, J. Kelle, C. Li, Y. Li, Z. Shao, P. Stone, and J. P. Hanna, "Reinforcement Learning Within the Classical Robotics Stack: A Case Study in Robot Soccer," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[12] 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*. PMLR, 2023, pp. 734–760.

[13] 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)*, Dec. 2020, pp. 737–744.

[14] 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*, vol. 606, no. 7912, pp. 129–136, Jun. 2022, number: 7912 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41586-022-04743-9

[15] 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.

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

[17] 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*, 2021, pp. 15 141–15 151.

[18] 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)*. IEEE, 2014, pp. 3888–3895.

[19] K. J. Åström and P. Eykhoff, "System identification—A survey," *Automatica*, vol. 7, no. 2, pp. 123–162, Mar. 1971. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109871900598

[20] 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*, vol. 4, Mar. 1987, pp. 1131–1139. [Online]. Available: https://ieeexplore.ieee.org/document/1087968

[21] 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)*. IEEE, 2018, pp. 4243–4250.

[22] 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*. PMLR, 2018, pp. 817–828.

[23] J. P. Hanna, S. Desai, H. Karnan, G. Warnell, and P. Stone, "Grounded action transformation for sim-to-real reinforcement learning," *Machine Learning*, vol. 110, no. 9, pp. 2469–2499, 2021.

[24] 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)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 4397–4402. [Online]. Available: https://ieeexplore.ieee.org/document/9341149/

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

[26] N. Jiang, A. Kulesza, and S. Singh, "Abstraction Selection in Model-based Reinforcement Learning," in *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Jun. 2015, pp. 179–188, iSSN: 1938-7228. [Online]. Available: https://proceedings.mlr.press/v37/jiang15.html

[27] 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, arXiv:2410.02172 [cs]. [Online]. Available: http://arxiv.org/abs/2410.02172

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

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

[30] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 aaai fall symposium series*, 2015.

[31] 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. [Online]. Available: http://arxiv.org/abs/1804.10332

[32] 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*, vol. 23, no. 12, pp. 1–83, 2022.

[33] 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*. PMLR, 2024, pp. 3511–3519.

[34] 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," Jun. 2022, arXiv:2206.08332 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2206.08332

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

[36] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[37] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[38] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *Arxiv Pre-print*, 2020.

[39] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International conference on machine learning*. PMLR, 2016.

[40] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.