Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

State Revisit and Re-explore: Bridging Sim-to-Real Gaps in Offline-and-Online Reinforcement Learning with An Imperfect Simulator

Xingyu Chen , Jiayi Xie , Zhijian Xu , Ruixun Liu , Shuai Yang , Zeyang Liu* , Lipeng Wan , Xuguang Lan*

National Key Laboratory of Human-Machine Hybrid Augmented Intelligence, National Engineering Research Center for Visual Information and Applications, and Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University

chenxingyu_1990@xjtu.edu.cn, {jiayixie, xuzhijian}@stu.xjtu.edu.cn
{liuruixun6343, ShuaiYang106594}@outlook.com, {zeyang.liu,wanlipeng}@xjtu.edu.cn,
xglan@mail.xjtu.edu.cn

Abstract

In reinforcement learning (RL) based robot skill acquisition, a high-fidelity simulator is usually indispensable but unattainable since the real environment dynamics are difficult to model, which leads to severe sim-to-real gaps. Existing methods solve this problem by combining offline and online RL to jointly learn transferable policies from limited offline data and imperfect simulators. However, due to the unrestricted exploration in the imperfect simulator, the hybrid offline-and-online RL methods inevitably suffer from low sample efficiency and insufficient state-action space coverage during training. To solve this problem, we propose a State Revisit and Re-exploration (SR^2) hybrid offline-and-online RL framework. In particular, the proposed algorithm employs a meta-policy and a sub-policy, where the meta-policy aims to find high-quality states in the offline trajectories for online exploration, and the sub-policy learns the robot skill using mixed offline and online data. By introducing the state revisit and explore mechanism, our approach efficiently improves performance on a set of sim-to-real robotic tasks. Through extensive simulation and real-world tasks, we demonstrate the superior performance of our approach against other state-of-the-art methods.

1 Introduction

Robot skill learning based on Reinforcement Learning (RL) [Mnih, 2013; Ibarz *et al.*, 2021; Johannink *et al.*, 2019; Kober *et al.*, 2013] has shown promising prospects to solve complex tasks in real-world scenarios such as dexterous manipulation, locomotion, and drone flight control [Celemin *et al.*, 2019; Xiang and Su, 2019; Andrychowicz *et al.*, 2020; Rudin *et al.*, 2022; Song *et al.*, 2023; Lee *et al.*, 2020b]. As directly training RL policy on real robots is extremely expensive and time-consuming, a simulator is usually indispensable

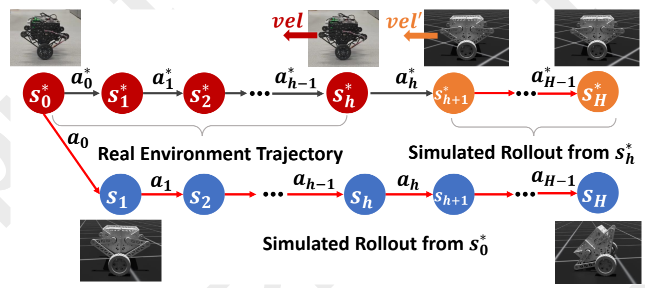


Figure 1: By introducing the state revisit and re-explore mechanism, a potentially valuable state s_h^* could be found to initialize the imperfect simulator for further exploration, which provides a higher probability to access s_H^*.

for learning robot skills. In most cases, a high-fidelity simulator is unattainable since the real environment dynamics are too complex or even unobservable to model. An alternative way is to simplify the environment dynamics and build a less accurate simulator to train the RL policies. However, directly deploying policies trained in the simulator to the real world is usually unfeasible because of the inconsistent environment dynamics, known as the sim-to-real gap problem.

The most straightforward approach is to train RL policy in a simulator and then fine-tune it with real data [Peng *et al.*, 2020; Kalashnikov *et al.*, 2018; Lee *et al.*, 2020a; James *et al.*, 2019], which heavily rely on the quality of the simulator and also need high-cost real robot training. To make the simulation dynamics closer to the real-world dynamics, domain randomization approaches optimize relevant parameters for several randomized simulated dynamics while training RL policies, which could achieve more adaptable policies in the real world [Chebotar *et al.*, 2019; Muratore *et al.*, 2019]. However, these methods usually need manually specified randomized parameters and nuanced randomization distributions, leading to unstable training or conservative policies.

Despite different methods, the cost will increase quickly when an approach involves online RL training on real robots. Therefore, offline RL approaches that train policies exclu-

*Corresponding authors.

sively with pre-collected datasets seem to be more affordable. As the performance of offline RL policies strongly depends on the size and state-action space coverage of the offline dataset, it is natural to combine offline RL on the real-world dataset with online RL in simulators, that is, offline-and-online RL methods. Previous methods such as [Niu *et al.*, 2022; Niu *et al.*, 2023; Xue *et al.*, 2024] borrow ideas from Conservative Q-Learning (CQL) [Kumar *et al.*, 2020], which designs a dynamics-aware policy evaluation scheme to penalize the Q-function learning on simulated state-action pairs with large dynamic gaps, which adaptively adjust the weights of the simulation policy and the real policy. However, due to the unrestricted exploration in the imperfect simulator, these methods inevitably suffer from low sample efficiency and insufficient state-action space coverage during training.

To solve this problem, we propose a State Revisit and Re-explore (SR^2) hybrid offline-and-online RL framework. The basis of our approach is to assume both real and simulation environments share the same state space, and there is only a difference between the transitional dynamics. It is natural that the accumulated errors grow with the exploration horizon increases. As shown in Figure 1, the key idea of our approach is to introduce a mechanism that ensures access to better quality and diversity of state-action data, which effectively improves the data coverage and sample efficiency as well as alleviates the sim-to-real gaps problem. Specifically, we design a hierarchical RL framework that consists of a meta-policy and a sub-policy. The meta-policy guides to trace back to the explored high-value state-action samples and generates more high-quality samples in the imperfect simulator. The sub-policy simultaneously learns the robot skills with offline real-world datasets and simulation rollouts. For the meta-policy, the sub-policy training procedure could be viewed as an interactive environment that provides feedback.

Through extensive simulation and real-world experiments, we demonstrate the superior performance of our approach against other state-of-the-art methods. In addition, we prove that the proposed algorithm guarantees a suboptimality with the polynomial sample complexity in the sim-to-real robot skill learning task.

2 Related Work

2.1 Sim-to-Real Policy Learning

The dynamic gaps between simulators and real environments, known as the sim-to-real gaps problem, have long been recognized as a main challenge in RL-based robot skill learning [Peng *et al.*, 2018; Zhao *et al.*, 2020]. To solve this problem, existing methods could be grouped into three lines. The methods such as [Peng *et al.*, 2020; Kalashnikov *et al.*, 2018; Lee *et al.*, 2020b; James *et al.*, 2019; Kaufmann *et al.*, 2023] adopt the most straightforward solution, which trains the policy in a simulator and then fine-tunes it in real environments. In these methods, a high-fidelity simulator is usually indispensable to guarantee a reliable policy transfer. Another line of works, such as [Chebotar *et al.*, 2019; Muratore *et al.*, 2019; Tobin *et al.*, 2017], are domain randomization methods, which train a policy variety of simulated environments with randomized properties. They close the sim-to-real gap by optimizing the simulation's dynamic parameters. There is also a line of works such as [Eysenbach *et al.*, 2020; Lyu *et al.*, 2024; Liu *et al.*, 2022; Liu *et al.*, 2024a; Niu *et al.*, 2022; Xu *et al.*, 2023; Xue *et al.*, 2024] which design a dynamics adaption mechanism to penalize the high dynamics-gap samples in the online exploration.

2.2 Hybrid Sim-to-Real Learning

Considering that directly learning transferable policy in simulators is extremely difficult, and directly training policies on real robots is unaffordable. The recent hybrid offline and online RL methods [Niu *et al.*, 2022; Niu *et al.*, 2023; Song *et al.*, 2022; Hou *et al.*, 2024; Song *et al.*, 2023] combine the advantages of offline RL [Kumar *et al.*, 2020; Agarwal *et al.*, 2020] and online RL, which provide a prospecting solution to learning policies from offline real-world data and an imperfect simulator. The key to these methods is to design a dynamics-aware policy evaluation scheme to penalize the Q-function learning on simulated state-action pairs with large dynamics gaps, which adaptively adjust the weights of the simulation policy and the real policy. However, due to the unrestricted exploration in the imperfect simulator, these methods inevitably suffer from low sample efficiency and insufficient state-action space coverage during training. Inspired by the efficient exploration methods such as [Uchendu *et al.*, 2023; Ecoffet *et al.*, 2021; Feng *et al.*, 2020; Liu *et al.*, 2024b], there are also a number of works [Wagenmaker *et al.*, 2024; Qu *et al.*, 2024] that aim to improve the state-action space coverage and the sample efficiency in the sim-to-real RL. Different from previous methods, our approach aims to enlarge the state-action space coverage in a meta-learning manner, while we also give an insight to analyze our approach in a theoretical way.

3 Background

3.1 Hybrid offline-and-online RL

Hybrid offline-and-online (H2O) RL consists of two MDPs $\{\mathcal{M}_s, \widetilde{\mathcal{M}_s}\} := (\mathcal{S}, \mathcal{A}, r, \{P_{\mathcal{M}_s}, P_{\widetilde{\mathcal{M}}_s}\}, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r is the reward function, $P_{\mathcal{M}_s}, P_{\widetilde{\mathcal{M}}_s}$ are the transitional dynamics corresponding to the real and the simulated environments, ρ represent the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. Given the offline dataset $\mathcal{D}$, which is generated by a behavior policy π_b in the real environment, the goal of H2O is to learn a transferable policy π_s using RL that maximizes the expected discounted rewards.

The basic idea is built upon the Conservative Q-Learning (CQL). It pushes down the dynamics-gap weighted Q-values and pulls up Q-values on trustworthy real offline data. The learning objective is designed as:

$$\min_Q \max_{d^\phi} \beta[\mathbb{E}_{s,a\sim d^\phi(s,a)}[Q(s,a)] - \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)] + \mathcal{R}(d^\phi)] + \widetilde{\mathcal{E}}(Q, \hat{\mathcal{B}}^\pi \hat{Q}), \tag{1}$$

where $d^\phi(s, a)$ is a particular state-action sampling distribution that is associated with high dynamics-gap samples,

$\mathcal{R}(d^{\phi})$ is a regularization term for d^{ϕ} to control the behavior of $d^{\phi}(s,a)$. $\widetilde{\mathcal{E}}(Q, \hat{\mathcal{B}}^{\pi}\hat{Q})$ represents the modified Bellman error of the mixed data from dataset $\mathcal{D}$ and the simulation rollout samples in online replay buffer B, which are generated by the real MDP $\mathcal{M}$ and the simulated MDP $\widetilde{\mathcal{M}}$.

4 Limitation of H2O

In this section, we theoretically analyze the bottlenecks affecting the sample efficiency of previous H2O RL methods.

Definition 1 (Dynamics Residual)**.** *Let $\mathcal{M}_s$ and $\widetilde{\mathcal{M}}_s$ represent the real and simulated MDPs. There is only a difference between the transitional dynamics $P_{\mathcal{M}_s}$ and $P_{\widetilde{\mathcal{M}}_s}$. The dynamics residual is defined as:*

$$\triangle(s_t, a_t, s_{t+1}) = P_{\mathcal{M}_s}(s_{t+1}|s_t, a_t) - P_{\widetilde{\mathcal{M}}_s}(s_{t+1}|s_t, a_t). \tag{2}$$

As H2O involves the offline and online MDPs, its occupancy distribution could be defined as:

$$d_h(s,a) = \frac{1}{2}\left[d^{\mathcal{M}_s}_{\pi_b,h}(s,a) + d^{\widetilde{\mathcal{M}}_s}_{\pi_s,h}(s,a)\right], \tag{3}$$

where the $d^{\mathcal{M}_s}_{\pi_b,h}(s,a)$, $d^{\widetilde{\mathcal{M}}_s}_{\pi_s,h}(s,a)$ are the occupancy distributions correspond to the real offline data and simulated data.

Denote the occupancy distribution of the optimal policy π^* by $d^{\mathcal{M}_s}_{\pi^*,h}(s,a;\rho)$. Considering that in the real environment, there is an optimal trajectory $\{s_0, s_1, ..., s_h\}$ that π^* can visit, the probability of visiting s_h in $\widetilde{\mathcal{M}}$ could be defined as:

$$\begin{aligned} d^{\widetilde{\mathcal{M}}}_{\pi_s,h}(s,a) &= \mathbb{P}(s_h = s, a_h = a; s_0 \sim \rho) \\ &= \sum_{s_0} \rho(s_0) \prod_{t=0}^{h-1} P_{\widetilde{\mathcal{M}}_s}(s_{t+1}|s_t,a_t)\pi_s(a_t|s_t) \\ &= \sum_{s_0} \rho(s_0) \prod_{t=0}^{h-1} (P_{\mathcal{M}_s}(s_{t+1}|s_t,a_t) - \\ &\qquad \triangle(s_t,a_t,s_{t+1}))\pi_s(a_t|s_t). \end{aligned} \tag{4}$$

Theorem 1. *Within the inaccurate transitional dynamics, there exists an MDP instance such that one has to suffer from an exponential sample complexity in total horizon H in order to explore a state that guarantees the policy to be suboptimal.*

5 Method

Aiming to solve the low sample efficiency and insufficient state-action space exploration problems in the hybrid offline-and-online MDPs, we propose a state revisit and re-explore RL algorithm. Specifically, we introduce a meta MDP into the H2O framework, denoted as $\mathcal{M}_m := (\mathcal{S}_m, \mathcal{A}_m, r_m, P_{\mathcal{M}_m}, \rho_m, \gamma_m)$, where $\mathcal{S}_m$ is the state space, $\mathcal{A}_m$ is the action space, r_m is the reward function, $P_{\mathcal{M}_m}$ is the transitional dynamics, ρ_m is the initial state distribution, and $\gamma_m \in [0,1)$ is the discount factors. Therefore, the proposed framework consists of a meta-policy π_m and a sub-policy π_s, where the π_m guides the generalization of high-quality samples in the imperfect simulator to boost the training of π_s, and π_s adaptively learns the robot skill using mixed offline and online generated data.

5.1 State Revisit and Re-explore

State revisit by meta-policy. Regarding the training procedure π_s as an interactive environment, π_m is expected to find some valuable state-action samples in $\mathcal{D}$ and then re-explore them in the imperfect simulator. Considering that directly finding the valuable states according to the training procedure of π_s is intractable, we simply decompose this process into two steps. First, we sort the state-action samples in $\mathcal{D}$ according to their rewards. Then we randomly choose some samples according to their position in the sorted dataset. We construct the state $s^{(i)}_{m,t} = (\eta^{(i)}, z^{(i)}_t, z^{(i)}_{t-1}, a^{(i)}_{m,t-1})$, where $\eta = \frac{Q_{s,t} - Q_{s,t-1}}{Q_{s,t-1}}$ is the change in Q-value of the sub-policy π_s at iteration t which makes the meta-policy promotes the learning of sub-policy. z_t and z_{t-1} are sampled from a uniform distribution over $(0,1)$ which represent the normalized position index at current and previous iterations. $a_{m,t-1} \in \{0,1\}$ is the action of previous iteration. As all state-action samples are sorted, the position implicitly encodes the reward information. Taking a $s_{m,t}$ as input, π_m decides whether to conduct the exploration. Therefore, the learning objective of meta-policy is defined as:

$$\begin{aligned} \hat{Q}_m \leftarrow \arg\min_{Q_m} \mathbb{E}_{s_{m,t},a_{m,t},s_{m,t+1}\sim B_m}[(Q_m(s_{m,t},a_{m,t}) \\ - \hat{\mathcal{B}}^{\pi_m}\hat{Q}_m(s_{m,t},a_{m,t}))^2], \end{aligned} \tag{5}$$

$$\hat{\pi}_m \leftarrow \arg\max_{\pi_m} \mathbb{E}_{s_{m,t}\sim B_m, a_{m,t}\sim\pi_m}\left[\hat{Q}_m(s_{m,t},a_{m,t})\right], \tag{6}$$

where $\hat{\mathcal{B}}^{\pi_m}$ is the Bellman operator and B_m is the replay buffer.

State re-explore by Sub-policy. As π_m determines to re-explore a pre-visited state or explore from an initial state in the imperfect simulator, the replay buffer B_{s,π_m} used for sub-policy learning contains two kinds of samples, including the samples in dataset $\mathcal{D}$, and the samples generated in the simulated environments under the control of π_m. Therefore, we adopt the objective in Eq.(1) to learn π_s where the only difference is the modified Bellman error, defined as:

$$\begin{aligned} \widetilde{\mathcal{E}}\left(Q, \hat{\mathcal{B}}^{\pi_s}\hat{Q}\right) = \frac{1}{2}\mathbb{E}_{s_t,a_t,s_{t+1}\sim\mathcal{D}}\left[\left(Q - \hat{\mathcal{B}}^{\pi_s}\hat{Q}\right)(s_t,a_t)\right]^2 + \\ \frac{1}{2}\mathbb{E}_{s_t,a_t,s_{t+1}\sim B_{s,\pi_m}}\left[\frac{P_{\mathcal{M}}}{P_{\widetilde{\mathcal{M}}}}\left(Q - \hat{\mathcal{B}}^{\pi_s}\hat{Q}\right)(s_t,a_t)\right]^2, \end{aligned} \tag{7}$$

where $\frac{P_{\mathcal{M}}}{P_{\widetilde{\mathcal{M}}}} = \frac{P_{\mathcal{M}}(s_{t+1}|s_t,a_t)}{P_{\widetilde{\mathcal{M}}}(s_{t+1}|s_t,a_t)}$, $P_{\mathcal{M}}(s_{t+1}|s_t,a_t)$ and $P_{\widetilde{\mathcal{M}}}(s_{t+1}|s_t,a_t)$ are the transitional dynamics of the real and simulation environments.

5.2 Algorithm

We summarize the training procedure of SR^2 in Algorithm 1. To start, we sort the state-action samples according to the rewards and get a sorted dataset $\mathcal{D}^{sort}$. To train the meta policy, we first sample N trajectories. For each of them, based on a random scalar $z^{(i)}_t$ which represents the normalized position in $\mathcal{D}^{sort}$, we adopt a positional function that selects an

Algorithm 1: State Revisit and Re-explore(SR^2)

Data: an offline dataset $\mathcal{D}$ from real environment, an imperfect simulator with biased dynamics $\mathcal{M}_s$
Initialize: Q function Q_m,Q_s, actor network π_m,π_s, sub-policy mixed replay buffer $B_s = \varnothing$, meta-policy replay buffer $B_m = \varnothing$, A reward-based reordered dataset $\mathcal{D}^{sort} = Sort(\mathcal{D}, r)$
for *step* $t = 1, \cdots, T$ **do**
  for *rollout trajectories* $i = 1, \cdots, N$ **do**
    $z_t^{(i)} \sim Uniform(0, 1)$;
    $(s_{\mathcal{D},t}^{(i)}, a_{\mathcal{D},t}^{(i)})$=POS($\mathcal{D}^{sort}, z_t^{(i)}$);
    $s_{m,t}^{(i)} = (\eta^{(i)}, z_t^{(i)}, z_{t-1}^{(i)}, a_{m,t-1}^{(i)})$;
    $a_{m,t}^{(i)} = \pi_m(a_{m,t}^{(i)} | s_{m,t}^{(i)})$;
    $B_s \leftarrow B_s \cup \textsc{Rollout}(\pi_s, \widetilde{\mathcal{M}}_s, a_{m,t}^{(i)}, s_{\mathcal{D},t}^{(i)})$;
  end
  $\pi_s, Q_{s,t+1} \leftarrow \textsc{TrainPolicy}(\pi_s, Q_{s,t}, B_s, \mathcal{D})$;
  for *rollout trajectories* $i = 1, \cdots, N$ **do**
    $r_{m,t}^{(i)} = \Delta Q_{s,t}(s_{\mathcal{D},t}^{(i)}, a_{\mathcal{D},t}^{(i)})$;
    $B_m \leftarrow B_m \cup (s_{m,t}^{(i)}, a_{m,t}^{(i)}, r_{m,t}^{(i)}, {s'}_{m,t}^{(i)})$;
    if *t % meta_policy_update_period = 0* **then**
      $\pi_m, Q_m \leftarrow$ $\textsc{TrainPolicy}(\pi_m, Q_m, B_m)$;
    end
  end
end

initial state-action pair. By constructing the state $s_{m,t}$, the meta-policy π_m takes it as input and decides whether to conduct the re-exploration in the imperfect simulator. Guided by the meta-policy, the sub-policy collects state-action samples in a replay buffer and mixes the samples with offline real data. Then the sub-policy could be optimized in the hybrid offline-and-online learning. According to the feedback of the sub-policy training procedure, the meta-policy could be optimized by Eq.(5) and Eq.(6).

Specifically, in this work, the sub-policy is implemented using H2O, chosen for its advantageous scalability. It is vital to note that employing other algorithms as sub-policy is also feasible due to the portability of SR^2. The meta-policy consists of a random policy and the Deep Q-Network (DQN) algorithm. The random policy function is employed to select state-action pairs from the real environment dataset, whereas the DQN algorithm, given the current sub-policy's training information and the corresponding state-action pair information, decides whether to reset the simulator from the selected state to roll out a new trajectory.

5.3 Theoretical Comparison

In this section, we provide the theoretical analysis, showing that our approach could effectively enlarge the state-action space coverage while yielding a polynomial sample complexity. We focus on comparing SR^2 with finite-horizon pessimistic offline MDPs e.g. CQL, and the hybrid method H2O.

As our approach also involves offline and online MDPs, the occupancy distribution also consists of two parts, like Eq.(3). In our approach, we design a state revisit and re-explore mechanism that automatically finds potential high-value states in the offline dataset by the meta policy π_m and uses them to re-initialize the imperfect simulator and start from them for further exploration. Different from H2O, the probability of visiting s_h in $\widetilde{\mathcal{M}}$ could be defined as:

$$
\begin{aligned}
& d_{\pi_s,h}^{\widetilde{\mathcal{M}}}(s,a) = \mathbb{P}(s_h = s, a_h = a; \pi_m, \rho, \mathcal{D}) \\
&= \frac{1}{2}\sum_{s_k} \frac{1}{|\mathcal{D}|}\pi_m(a_{m,k}|s_{m,k}) \prod_{t=k}^{h-1} P_{\widetilde{\mathcal{M}}_s}(s_{t+1}|s_t,a_t)\pi_s(a_t|s_t) \\
&+ \frac{1}{2}\sum_{s_0} \rho(s_0) \prod_{t=0}^{h-1} (P_{\widetilde{\mathcal{M}}_s}(s_{t+1}|s_t,a_t)\pi_s(a_t|s_t).
\end{aligned} \tag{8}
$$

Definition 2 (Policy concentrability of finite-horizon MDPs)**.** *The policy concerntrability coefficient of π is defined as:*

$$
C^* := \max_{(s,a,h)\in\mathcal{S}\times\mathcal{A}\times[H]} \frac{d_h^*(s,a)}{d_h^\pi(s,a)}, \tag{9}
$$

where $d_h^*(s,a)$ and $d_h^\pi(s,a)$ are the state visitation distributions of optimal policy π^* and the learned policy π at time step h, respectively.

According to Theorem 1, for any states s_h that π^* could visit, the unrestricted exploration in the imperfect simulator suffers from an exponential sample complexity. However, in Eq.(8), guided by the meta policy π_m, there is a probability that π_s finds an explored state s_k and further re-explore it in the imperfect simulator for $h-k$ steps. According to Definition 2, it is easy to conclude that $C^*_{CQL} > C^*_{H2O} > C^*_{SR^2}$. Then we can get Theorem 2.

Theorem 2. *With an appropriate choice of sample strategy and the meta-policy, Algorithm 1 guarantees a suboptimality bound up to a polynomial sample complexity that is lower than CQL and H2O.*

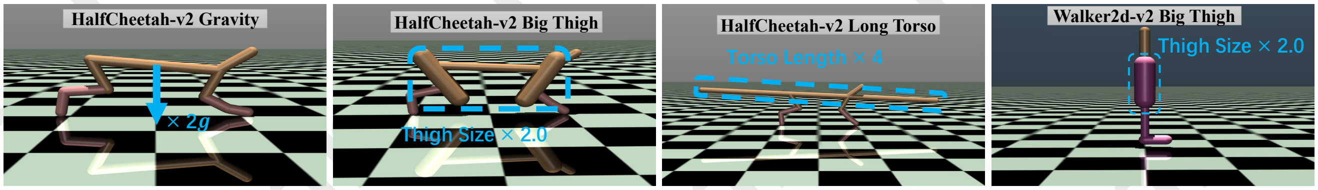
6 Experiments

In this section, we conduct experiments to validate the effectiveness of the proposed algorithm (SR^2) and compare it with other state-of-the-art methods in the field of cross-domain online and offline RL. We begin with a detailed description of the experimental setups, including the environment setting and the implementation of SR^2 and baselines, followed by presenting the results of the benchmark experiments performed within the MuJoCo simulation environment and real robot experiments.

6.1 Experimental Setups

Simulation-based Experiments

To empirically validate the capacity of policy adaptation, we create two types of sim-to-real gaps. In this experiment, the real environment is configured as the standard OpenAI Gym MuJoCo. We select the standard HalfCheetah and Walker2d. Due to the non-interactive properties of the real environment,



(a) HalfCheetah Gravity (b) HalfCheetah Big Thigh (c) HalfCheetah Long Torso (d) Walker2d Big Thigh

Figure 2: Illustrations of modified dynamics in MuJoCo environments

we choose the widely used offline RL benchmark D4RL [Fu *et al.*, 2020]. For the simulated environment, as shown in Figure 2, the dynamics gaps are introduced by modifying the corresponding parameters in the configuration file or adding noise in robot actions. Following [Niu *et al.*, 2023], we modify the gravity coefficient ($\times 2$, **Gravity**), friction coefficient ($\times 0.3$, **Friction**), adding Gaussian noise to joint actuators ($N(0, action\_range)$, **Joint Noise**), increasing thigh size ($\times 2$, **Big Thigh**), increasing thigh joint motion range($\times 2$, **Flexible Thigh**), stretching torso length ($\times 4$, **Long Torso**).

Real-world Experiments

We utilize an 8kg wheel-legged robot in Figure 3 as the platform for our real-world experiments. The robot employs a VMC controller[Pratt *et al.*, 2001] to fix the relative position between the waist joint and the wheeled feet, allowing the RL model to control only the hub motors of a pair of wheels. To facilitate the experiment, we manually collect a dataset of 1M steps from the real world. Subsequently, we deploy H2O, PAR and SR^2 in Isaac Sim for training.

We define the task as standing still, where the real wheel-legged robot maintains balance in place by relying on two wheels under RL control. The state space is defined as $(\theta, \dot{\theta}, \phi, \dot{\phi}, x, \dot{x})$, where θ is the pitch angle, $\dot{\theta}$ is the pitch angular velocity, ϕ is the yaw angle, $\dot{\phi}$ is the yaw angular velocity, x is the linear displacement, and $\dot{x}$ is the linear velocity. In Isaac Sim, we can directly read the aforementioned states, while in the real world, we deploy a Visual-Inertial Odometry based on the onboard stereo camera and IMU data to obtain these states. The actions are defined as the torques τ_l and τ_r of the two wheels. We also define a set of hyperparameters $(c_1, \ldots, c_8)$ to adjust the weights of the penalties for each offset during the trajectory collection procedure. Accordingly, the reward is calculated as:

$$r = 40 - c_1\theta^2 - c_2\dot{\theta}^2 - c_3\phi^2 - c_4\dot{\phi}^2 - c_5x^2 - c_6\dot{x}^2 - c_7\tau_l^2 - c_8\tau_r^2.$$

We aim to maximize the reward when the robot is stationary. To achieve this, we impose penalties on pitch angle offset and displacement offset, assigning relatively higher penalty weights to θ and x. Additionally, to ensure that the output torques do not become excessively large, we also increase the penalty weights on τ_l and τ_r.

Baselines

Our baselines include the following algorithms: the cross-domain online RL algorithm DARC [Eysenbach *et al.*, 2020], the dynamics-aware hybrid offline-and-online RL algorithm H2O [Niu *et al.*, 2022], its improved version H2O+ [Niu *et al.*, 2023], and the recently proposed Policy Adaptation by Representation mismatch algorithm PAR [Lyu *et al.*, 2024]. In our experiments, we use the same set of five seeds. All experiments involve a simulated environment that allows online interaction but has differences in dynamics or morphology, while the real environment is non-interactive but comes with pre-collected datasets, such as HalfCheetah-medium replay, HalfCheetah-medium, HalfCheetah-medium expert, and Walker2d-medium replay. During the execution of the algorithms, evaluations are conducted in a fully interactive real environment. We adopt the experimental setups of H2O and H2O+, involving 1M interactions with the simulator and 1M training steps, which aligns better with sim-to-real research logic, rather than the offline design in PAR[Lyu *et al.*, 2024], where the source domain is non-interactive and the target domain is interactive. Given that the advantages of the meta-policy may not be fully manifested under fewer iterations and training steps, we also conduct a set of SR^2 experiments with 3M training interactions.

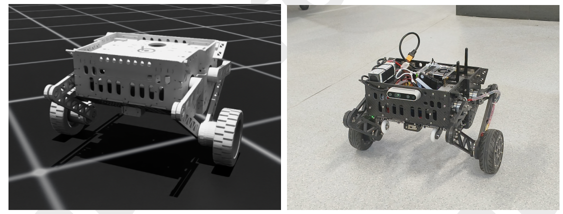


Figure 3: Wheel-legged robot in the sim/real environments.

6.2 Comparative Experiment

Mujoco Benchmark Experiments

Our comparative results shown in Table 1 demonstrate that SR^2 achieves superior or highly competitive performance on 20 out of 23 tasks. Notably, the results of DARC and H2O+ are absent in the HalfCheetah Medium Expert experiment, as we were unable to find a Pytorch implementation of DARC, and H2O+ has not been open-sourced yet. The corresponding data in table 1 are sourced from H2O [Niu *et al.*, 2022] and H2O+ [Niu *et al.*, 2023]. The results of PAR are obtained using its open-source code with only the source and target domain adjusted to align with the experimental settings, but its performance was slightly below expectations, potentially due to default parameter settings. While SR^2 can integrate PAR as a sub-policy, further development will proceed after confirming the suitability of PAR's parameter configurations.

It is important to note that the -m and -mr datasets are considered to have higher data diversity, covering a large num-

Data	Dynamics Gap	DARC	H2O	H2O+	PAR	SR^2(1M)	SR^2(3M)
HalfCheetah-mr	Gravity	5105±460	6813±289	6861±268	5891±71	**7016±173**	**7241±152**
	Friction	5503±263	5928±896	6278±1336	6151±197	**7315±685**	**7608±239**
	Joint Noise	5137±225	6747±427	6985±328	5557±104	**7358±158**	**7567±75**
	Big Thigh	5336±389	6278±305	**6675±231**	5525±117	**6588±142**	**6659±188**
	Flexible Thigh	5554±88	6976±234	**7497±196**	6683±75	**7449±180**	**7702±112**
	Long Torso	45±322	6225±100	**6718±245**	5968±54	6568±187	6569±157
	Mean Return	5863	6573	6947	5978	**7014**	**7236**
HalfCheetah-m	Gravity	5011±456	7085±416	6965±659	5565±497	**7330±188**	**7655±214**
	Friction	6113±104	6848±445	7186±859	6888±207	**7453±306**	**7579±206**
	Joint Noise	5484±171	7212±236	7503±237	478±266	**7614±133**	**8018±106**
	Big Thigh	6302±1832	6625±579	7094±371	5850±200	**7021±87**	**7411±162**
	Flexible Thigh	7266±1771	7005±757	**7805±139**	7114±245	7659±216	**8217±251**
	Long Torso	724±921	6327±602	5484±1382	2299±1311	**6806±309**	**6958±165**
	Mean Return	6054	6896	7187	2367	**7296**	**7688**
HalfCheetah-me	Gravity	4759±353	4707±779	/	1802±1207	**5537±222**	**6734±432**
	Friction	**9038±1480**	6745±562	/	4868±756	4833±340	**8766±450**
	Joint Noise	5288±104	5280±1329	/	565±473	**5599±872**	**7426±515**
	Big Thigh	/	5062±288	/	4514±942	**5243±166**	**7323±203**
	Flexible Thigh	/	**7466±422**	/	6626±717	**7427±676**	**10430±347**
	Long Torso	/	3307±892	/	2591±328	**3894±642**	**6250±652**
	Mean Return	/	5579	/	4250	**5677**	**8410**
Walker2d-mr	Gravity	2969±1043	3366±740	3518±605	3673±109	**4078±66**	**4180±55**
	Friction	3644±213	3916±549	3866±840	4033±51	**4319±46**	**4451±70**
	Joint Noise	-3±0	3045±911	3446±862	3938±105	**4410±38**	**4543±138**
	Big Thigh	57±126	1789±1781	2977±771	3623±128	**4180±146**	**4189±162**
	Flexible Thigh	2511±1048	1891±1001	3535±493	3961±146	**4335±89**	**4476±63**
	Mean Return	1624	2738	3596	3738	**4306**	**4432**

Table 1: Average returns for MuJoCo HalfCheetah and Walker2d tasks.

ber of samples. Consequently, learning from these datasets presents a lower difficulty. However, SR^2 outperforms other algorithms on these simple tasks, primarily due to the role of the meta-policy in reducing sample complexity and improving data utilization efficiency. In the HalfCheetah-me tasks, most baselines fail to perform well, while SR^2 maintains strong competitiveness in this scenario. Moreover, when interactions and training steps are increased to 3M, SR^2's performance shows a significant improvement. This could be attributed to the characteristics of the HalfCheetah-me dataset, which is considered intermediate or expert-level with lower data diversity, limited sample coverage, and higher sample complexity. Baseline models struggle to learn sufficient patterns or features from this data, leading to poor performance. In contrast, SR^2 effectively reduces sample complexity, allowing it to gain a substantial advantage in the more complex HalfCheetah-me tasks.

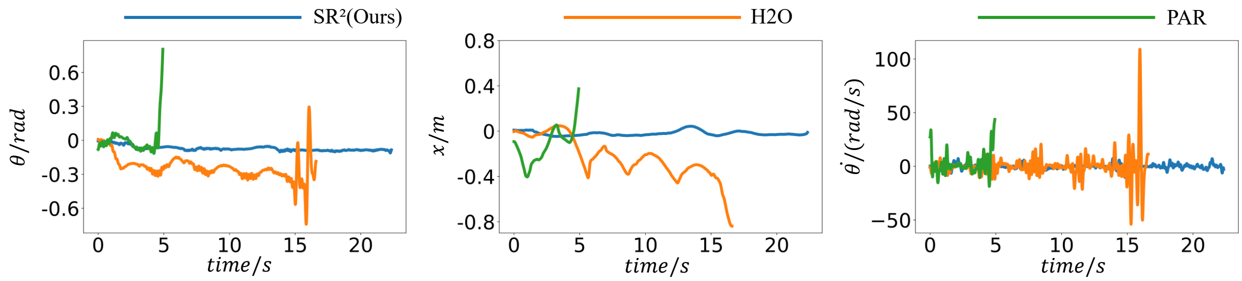
Real Robot Transfer Experiments

The experimental results on the real robot are shown in Figure 4, where SR^2 demonstrates significantly better control performance compared to H2O and PAR. From the pitch angle information in Figure 4(a), it can be observed that PAR loses balance and falls after approximately 5 seconds, while H2O maintains balance for about 15 seconds but gradually becomes unstable and ultimately fails to remain balanced. In contrast, SR^2 exhibits excellent balance throughout the entire experiment. Figure 4(b) further reveals that PAR initially moves backward and subsequently attempts to move forward again, but eventually loses stability. H2O, on the other hand, gradually deviates from the starting point and fails to force the robot to the starting point and stabilize it despite briefly maintaining balance. SR^2 successfully maintains high stability by making fine adjustments near the starting point. Figure 4(c) illustrates the changes in pitch angular velocity, showing that SR^2 achieves more stable performance, while H2O and PAR exhibit significant fluctuations and instability. Considering that the sim-to-real task is more challenging than Mujoco benchmark experiments due to the difficulty in quantifying dynamic gaps and morphological gaps, the overall task complexity is significantly higher. In such a complex task, SR^2 outperforms other baselines primarily due to the role of its meta-policy in improving sample efficiency.

6.3 Analytical Experiments

Ablation Study

In the ablation study, we aim to evaluate the specific contributions of each component in the SR^2 algorithm. The primary components of SR^2 include the meta-policy and sub-policy, with the sub-policy being selected as the H2O algorithm in this study. In the comparison experiments, the meta-policy is composed of a random policy and a DQN policy. The ran-



(a) Pitch Angle (b) Displacement (c) Pitch Angular Velocity

Figure 4: The real robot experiments results of standing still.

Task	H2O	SR^2-random	SR^2-random_dqn
mr-Friction	5928±896	7098±216	**7315±685**
m-Joint Noise	7212±236	7495±261	**7614±133**

Table 2: Ablation study for meta-policy

dom policy is employed to select state-action pairs from the real dataset, while the DQN policy, based on the information from the sub-policy's training and the state-action pairs selected by the random policy, decides whether to reset the simulator from the chosen state to roll out a new trajectory. Through this experiment, we seek to investigate the roles and effects of the random policy and DQN policy within SR^2.

The experiments are divided into three groups for evaluation: the H2O group, the H2O with random policy group(SR^2-random), and the H2O with random policy and DQN policy group (SR^2-random_dqn). The H2O group utilizes the open-source original version of the H2O algorithm. In the H2O with random policy group (SR^2-random), the sub-policy uses H2O, and the meta-policy consists of a random policy, which selects any state-action pair and resets the simulator from the selected state to roll out a new trajectory. In the H2O with random policy and DQN policy group(SR^2-random_dqn), the sub-policy still uses H2O, while the meta-policy is composed of both the random policy and the DQN policy, aligning with the SR^2 in the comparison experiment.

The ablation study results are shown in Table 2. We observe that using only a random policy as the meta-policy (SR^2-random) leads to a significant improvement in model performance. Furthermore, when combining a random policy with a DQN policy as the meta-policy(SR^2-random_dqn), the performance of the model improves even further. This demonstrates that our meta-policy approach effectively enhances model performance by reducing sample complexity and improving data utilization.

Research on Data Validity

Based on our theoretical analysis, when the optimal strategy in the real world accesses the real data state-action pair (s, a), the probability of SR^2 accessing this state-action pair is higher than that of H2O. In other words, SR^2 accesses more valid data compared to H2O. From a broader perspective, we can use H2O's discriminator to validate the authenticity of the data. Therefore, we conduct an experiment. For the data sampled during training, we record the number of data triplets

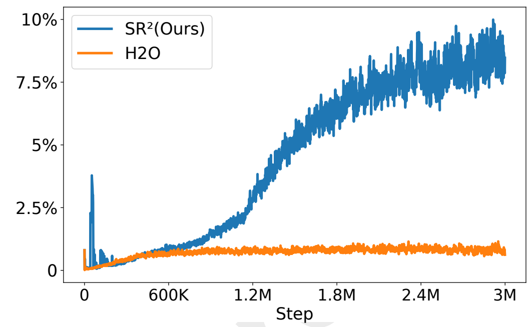


Figure 5: The Proportion of valid (s, a, s')

(s, a, s') in every training step that the H2O's discriminator classified as having a higher real probability than sim probability, and we calculate the proportion of these data triplets (s, a, s') among all the sampled triplets during the training process. If the real probability is judged to be higher than the sim probability, the data will be referred to as valid data.

As shown in Figure 5, in the HalfCheetah-me Long Torso task, the valid data of SR^2, including data triplets (s, a, s'), is generally more abundant than that of H2O. This indicates that from the perspective of the discriminator the data explored by SR^2 are closer to the real distribution.

7 Conclusion

In Reinforcement learning (RL) based robot skill acquisition, a high-fidelity simulator is usually indispensable but unattainable since the real environment dynamics are difficult to model, which leads to severe sim-to-real gaps. To deal with this problem, we propose a State Revisit and Re-explore (SR^2) hybrid offline-and-online RL algorithm in this paper. The proposed algorithm employs a meta-policy and a sub-policy, where the meta-policy aims to find high-quality states in the offline trajectories for online exploration, and the sub-policy learns the robot skill using mixed offline and online data. Through extensive simulation and real-world experiments, we demonstrate the superior performance of our approach against other state-of-the-art methods. In addition, we prove that the proposed algorithm guarantees a suboptimality with the polynomial sample complexity in the sim-to-real robot skill learning task.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under grant No.2024YFB4707600, NSFC under grant No.62125305, No.U23A20339, No.62088102, No.62203348, No.52435010, Shaanxi Natural Science Basic Research Program 2025SYS-SYSZD-083.

References

[Agarwal *et al.*, 2020] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pages 104–114. PMLR, 2020.

[Andrychowicz *et al.*, 2020] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[Celemin *et al.*, 2019] Carlos Celemin, Guilherme Maeda, Javier Ruiz-del Solar, Jan Peters, and Jens Kober. Reinforcement learning of motor skills using policy search and human corrective advice. *The International Journal of Robotics Research*, 38(14):1560–1580, 2019.

[Chebotar *et al.*, 2019] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[Ecoffet *et al.*, 2021] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[Eysenbach *et al.*, 2020] Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*, 2020.

[Feng *et al.*, 2020] Fei Feng, Ruosong Wang, Wotao Yin, Simon S Du, and Lin Yang. Provably efficient exploration for reinforcement learning using unsupervised learning. *Advances in Neural Information Processing Systems*, 33:22492–22504, 2020.

[Fu *et al.*, 2020] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[Hou *et al.*, 2024] Yiwen Hou, Haoyuan Sun, Jinming Ma, and Feng Wu. Improving offline reinforcement learning with inaccurate simulators. *arXiv preprint arXiv:2405.04307*, 2024.

[Ibarz *et al.*, 2021] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[James *et al.*, 2019] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12627–12637, 2019.

[Johannink *et al.*, 2019] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[Kalashnikov *et al.*, 2018] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[Kaufmann *et al.*, 2023] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[Kober *et al.*, 2013] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[Kumar *et al.*, 2020] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[Lee *et al.*, 2020a] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

[Lee *et al.*, 2020b] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[Liu *et al.*, 2022] Jinxin Liu, Hongyin Zhang, and Donglin Wang. Dara: Dynamics-aware reward augmentation in offline reinforcement learning. *arXiv preprint arXiv:2203.06662*, 2022.

[Liu *et al.*, 2024a] Jinxin Liu, Hongyin Zhang, Zifeng Zhuang, Yachen Kang, Donglin Wang, and Bin Wang. Design from policies: Conservative test-time adaptation for offline policy optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[Liu *et al.*, 2024b] Zeyang Liu, Lipeng Wan, Xinrui Yang, Zhuoran Chen, Xingyu Chen, and Xuguang Lan. Imagine, initialize, and explore: An effective exploration method in multi-agent reinforcement learning. In *Proceedings of* *the AAAI Conference on Artificial Intelligence*, volume 38, pages 17487–17495, 2024.

[Lyu *et al.*, 2024] Jiafei Lyu, Chenjia Bai, Jingwen Yang, Zongqing Lu, and Xiu Li. Cross-domain policy adaptation by capturing representation mismatch. *arXiv preprint arXiv:2405.15369*, 2024.

[Mnih, 2013] Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Muratore *et al.*, 2019] Fabio Muratore, Michael Gienger, and Jan Peters. Assessing transferability from simulation to reality for reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1172–1183, 2019.

[Niu *et al.*, 2022] Haoyi Niu, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming Hu, Xianyuan Zhan, et al. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. *Advances in Neural Information Processing Systems*, 35:36599–36612, 2022.

[Niu *et al.*, 2023] Haoyi Niu, Tianying Ji, Bingqi Liu, Haocheng Zhao, Xiangyu Zhu, Jianying Zheng, Pengfei Huang, Guyue Zhou, Jianming Hu, and Xianyuan Zhan. H2o+: an improved framework for hybrid offline-and-online rl with dynamics gaps. *arXiv preprint arXiv:2309.12716*, 2023.

[Peng *et al.*, 2018] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter 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.

[Peng *et al.*, 2020] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[Pratt *et al.*, 2001] Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001.

[Qu *et al.*, 2024] Chengrui Qu, Laixi Shi, Kishan Panaganti, Pengcheng You, and Adam Wierman. Hybrid transfer reinforcement learning: Provable sample efficiency from shifted-dynamics data. *arXiv preprint arXiv:2411.03810*, 2024.

[Rudin *et al.*, 2022] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[Song *et al.*, 2022] Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. *arXiv preprint arXiv:2210.06718*, 2022.

[Song *et al.*, 2023] Yunlong Song, Angel Romero, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, 8(82):eadg1462, 2023.

[Tobin *et al.*, 2017] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[Uchendu *et al.*, 2023] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. Jump-start reinforcement learning. In *International Conference on Machine Learning*, pages 34556–34583. PMLR, 2023.

[Wagenmaker *et al.*, 2024] Andrew Wagenmaker, Kevin Huang, Liyiming Ke, Byron Boots, Kevin Jamieson, and Abhishek Gupta. Overcoming the sim-to-real gap: Leveraging simulation to learn to explore for real-world rl. *arXiv preprint arXiv:2410.20254*, 2024.

[Xiang and Su, 2019] Guofei Xiang and Jianbo Su. Task-oriented deep reinforcement learning for robotic skill acquisition and control. *IEEE transactions on cybernetics*, 51(2):1056–1069, 2019.

[Xu *et al.*, 2023] Kang Xu, Chenjia Bai, Xiaoteng Ma, Dong Wang, Bin Zhao, Zhen Wang, Xuelong Li, and Wei Li. Cross-domain policy adaptation via value-guided data filtering. *Advances in Neural Information Processing Systems*, 36:73395–73421, 2023.

[Xue *et al.*, 2024] Ruiqi Xue, Ziqian Zhang, Lihe Li, Feng Chen, Yi-Chen Li, Yang Yu, and Lei Yuan. Dynamics adaptive safe reinforcement learning with a misspecified simulator. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 74–91. Springer, 2024.

[Zhao *et al.*, 2020] Wenshuai Zhao, Jorge Peña Queralta, and Tomi 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. IEEE, 2020.