Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

Hindsight States: Blending Sim & Real Task Elements for Efficient Reinforcement Learning

Simon Guist, Jan Schneider, Alexander Dittrich, Vincent Berenz, Bernhard Schölkopf and Dieter Büchler

Abstract—**Reinforcement learning has shown great potential in solving complex tasks when large amounts of data can be generated with little effort. In robotics, one approach to generate training data builds on simulations based on dynamics models derived from first principles. However, for tasks that, for instance, involve complex soft robots, devising such models is substantially more challenging. Being able to train effectively in increasingly complicated scenarios with reinforcement learning enables to take advantage of complex systems such as soft robots. Here, we leverage the imbalance in complexity of the dynamics to learn more sample-efficiently. We (i) abstract the task into distinct components, (ii) off-load the simple dynamics parts into the simulation, and (iii) multiply these virtual parts to generate more data *in hindsight*. Our new method, Hindsight States (HiS), uses this data and selects the most useful transitions for training. It can be used with an arbitrary off-policy algorithm. We validate our method on several challenging simulated tasks and demonstrate that it improves learning both alone and when combined with an existing hindsight algorithm, Hindsight Experience Replay (HER). Finally, we evaluate HiS on a physical system and show that it boosts performance on a complex table tennis task with a muscular robot. Videos and code of the experiments can be found on webdav.tuebingen.mpg.de/his/.**

I. Introduction

Reinforcement learning (RL) holds great potential for devising optimal behavior for challenging robotics tasks. Difficulties in these tasks often arise from contact in object manipulation, the requirement to perform fast but accurate motions, or from hard-to-control systems like soft robots. While it has been shown that RL can handle such problems, the major drawback of RL methods to require large amounts of interactions with the environment remains. This fact makes learning on real robots challenging.

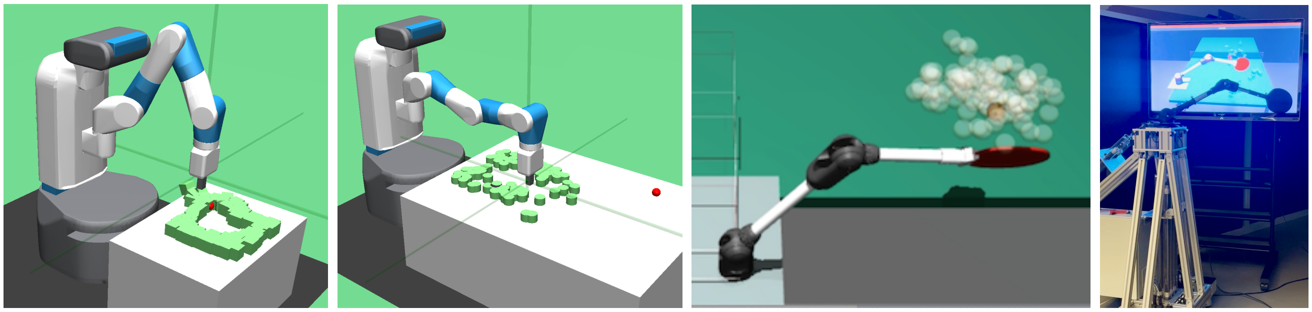
Training in simulation and transferring the resulting policy to the real system is one approach to alleviate this challenge. However, even small inaccuracies in the simulation can cause the simulated and real system to behave differently, a problem known as *reality gap*. This problem is worse for complex systems like soft robots, which are hard to model accurately. Sim-to-real techniques attempt to mitigate the impact of the reality gap. Many popular techniques accomplish more effective policy transfer by making the learned policies more robust to variations in the task dynamics through domain randomization [26, 41]. However, these methods are generally computationally expensive and require precise fine-tuning of the parameters, such as the amount of randomization on the dynamics parameters. These downsides of sim-to-real approaches accumulate as tasks advance in complexity, thus, necessitating learning many complex tasks partly or completely on the real system.

Hybrid sim and real (HySR) [5] is a way to ease the training of complex tasks on real systems. The method is based on the insight that for some tasks, certain parts of the environment have simpler dynamics than others. For instance, many robotic ball games consist of a robot, whose dynamics are more complex compared to those of the ball. The mismatch in modeling complexity is exacerbated if the robot is equipped with soft components, which render modeling even more difficult. The idea behind HySR is to keep the complicated parts of the task real, whereas the simpler parts are simulated. This strategy yields significant practical benefits, while facilitating the transfer to the entirely real system. First, the approach simplifies setting up the task since fewer real objects have to be handled that, for instance, are subject to wear-and-tear or require safety considerations. Second, it can greatly simplify the reset mechanism in episodic RL tasks. Robots are actuated and can typically reset themselves, while environment resets often require manually moving all objects to their initial positions. Third, data-augmentation techniques can be added to the virtual objects that would not be feasible on a fully real system. Lastly, the simulation provides information about, e.g., the exact positions of virtual objects or about contacts that occur, which might not be easily attainable for real objects. This data can be used to generate labels or rewards that can aid the training process. While HySR can simplify practical aspects of the training, the large number of real robot interactions required for training can still cause problems for real-world RL.

In this work, we present an approach that can significantly reduce the number of interactions with the real system required for learning. The key idea behind our method, Hindsight States (HiS), is to pair the data of a single real instance with additional data generated by concurrently simulating multiple distinct instances of the virtual part. In the example of robot ball games, our method simulates the effect of the robot's actions on several virtual balls simultaneously. We relabel the virtual part of each roll-out with this additional virtual data *in hindsight*. Intuitively, the agent experiences what would

This work was supported by the Max Planck Institute for Intelligent Systems.

All authors are affiliated with the MPI for Intelligent Systems, Max-Planck-Ring 4, 72076 Tübingen, Germany.



(a) Pushing (b) Sliding (c) Simulated robot table tennis (d) Real robot table tennis

Fig. 1: Visualization of HiS applied to the tasks considered in this work. Rather than training with a single object, HiS uses multiple virtual objects in parallel to generate more data and experience higher rewards with increased probability early on in the training.

have happened if it had encountered a different virtual part but applied the same actions as in the original episode. This relabeling process enables the RL agent to generate extra experience without collecting further transitions on the real system. The hindsight data can then be used in addition to the regular training data by any off-policy RL algorithm, such as Soft Actor-Critic (SAC) [12]. Particularly for tasks with sparse rewards, the additional data is beneficial since it increases the probability that the agent experiences positive rewards early in the training. The contributions of this paper are the following:

1) *Extending HySR for sample efficient training*
   Our main contribution is to devise a sample efficient way to train in the Hybrid Sim and Real (HySR) setting. To that end, we formalize the HySR training and, based on it, develop a novel algorithm, called Hindsight States (HiS).
2) *Evaluation of the interplay between HER and HiS*
   Our experiments demonstrate that the combination of HiS and HER achieves higher sample efficiency than each method by itself. We argue that HiS and HER improve task performance in distinct and complementary manners.
3) *Thorough experimental validation of HiS on a variety of tasks*
   We show improved performance of HiS on the original HySR real robot table tennis task. Additionally, we investigate to what extent HiS can be applied to manipulation tasks and report more efficient training in these regimes (see Fig. 1 for an overview of all experiments). Another finding is that HiS allows for more efficient training in entirely simulated environments w.r.t. wall-clock time.

II. Related Work

Our new method, HiS, shares some aspects with and takes inspiration from other works. In the following section, we discuss these similarities.

Hindsight Experience Replay (HER) [1] improves sample efficiency for sparse goal-conditioned RL tasks by relabeling the goal states of transitions in hindsight. The method replaces the original goal with a state reached by the agent. Then it trains an off-policy RL agent on a combination of original and relabeled data. This way, the agent receives additional positive feedback for reaching the relabeled goal, which can be more instructive than the negative feedback for missing the original goal. Rauber et al. [31] extend the idea to on-policy algorithms through the use of importance sampling. Dynamic Hindsight Experience Replay (DHER) [9] is a version of HER that supports dynamic goals, which change during the episode. The method makes the idea of relabeled goals applicable to tasks like grasping moving objects. While HER samples hindsight goals uniformly, recent methods prioritize goals based on how instructive the resulting transitions are for the agent. These approaches sample hindsight goals that guide the agent toward the true goal [2, 32, 10] or result in a large temporal difference (TD) error [22]. Other methods sample hindsight goals uniformly and prioritize more informative transitions when sampling from the replay buffer, akin to PER [33]. Zhao and Tresp [40] prioritize trajectories that demonstrate difficult behavior. They quantify difficulty by the increase of the system's energy over the course of the trajectory. Beyene and Han [3] primarily sample hindsight transitions that incur a large TD error. Similar to HER and its extensions, our method also relabels in hindsight. However, it focuses on exchanging the virtual part of the state rather than the goal.

HiS and sim-to-real techniques utilize simulations to devise a policy that transfers to reality. Different techniques, such as domain randomization [26, 28, 23] and domain adaptation [15, 14, 39] were proposed to mitigate the impact of the reality gap. HiS, on the other hand, assumes the reality gap to be sufficiently small for certain components of the environment and devises a way to be more efficient in the hybrid sim and real setting. Sim-to-real methods could be used on top of HiS to close the remaining gap.

Other works have investigated leveraging real and simulated data of the same task to improve learning performance. For instance, Kang et al. [16] learn a perception system using simulated and a reward model based on real data. They use model predictive control with the learned reward model to solve quadrotor navigation tasks. Di-Castro et al. [8] combine cheap but imprecise simulated with expensive real transitions to learn a policy for the real task. HiS, in contrast, investigates more efficient training by mixing simulated and real components in each transition.

Algorithm 1 Hindsight States (HiS)

```
Initialize off-policy RL algorithm $\mathbb{A}$, replay buffer $R$, choose criterion $c$
for episode $i = 1, 2, \dots$ do
    Sample $\mathbf{s}^{\mathrm{v}}_{1:T} \sim \mathcal{D}_{\mathrm{rec}}$ and initialize real system to $\mathbf{s}^{\mathrm{r}}_1$
    for $t = 1, \dots, T$ do
        Sample action $\mathbf{a}_t$ from $\mathbb{A}$ given $\mathbf{s}_t = [\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}}_t]$
        Apply $\mathbf{a}_t$ to the real system and observe $\mathbf{s}^{\mathrm{r}}_{t+1}$
        $\mathbf{s}^{\mathrm{v}}_{t+1} \sim \mathtt{sim}([\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}}_t], \mathbf{a}_t)$ or take $\mathbf{s}^{\mathrm{v}}_{t+1}$ from $\mathbf{s}^{\mathrm{v}}_{1:T}$
    end for
    for $t = 1, \dots, T$ do                                  ▷ standard replay
        Store $([\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}}_t], \mathbf{a}_t, r([\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}}_t], \mathbf{a}_t), [\mathbf{s}^{\mathrm{r}}_{t+1}, \mathbf{s}^{\mathrm{v}}_{t+1}])$ in $R$
    end for
    Initialize temporary buffer $R_{\mathrm{temp}}$                  ▷ HiS replay
    for $j = 1, 2, \dots$ do
        Sample $\mathbf{s}^{\mathrm{v}_m}_{1:T} \sim \mathcal{D}_{\mathrm{rec}}$
        for $t = 1, \dots, T$ do
            $\mathbf{s}^{\mathrm{v}_m}_{t+1} \sim \mathtt{sim}([\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}_m}_t], \mathbf{a}_t)$ or take $\mathbf{s}^{\mathrm{v}_m}_{t+1}$ from $\mathbf{s}^{\mathrm{v}_m}_{1:T}$
            $e^m_t = ([\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}_m}_t], \mathbf{a}_t, r([\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}_m}_t], \mathbf{a}_t), [\mathbf{s}^{\mathrm{r}}_{t+1}, \mathbf{s}^{\mathrm{v}_m}_{t+1}])$
            Store relabeled transition $e^m_t$ in $R_{\mathrm{temp}}$
        end for
    end for
    Every few episodes:
        Add transitions from $R_{\mathrm{temp}}$ to $R$ using criterion $c$
    Every few steps:
        Perform optimization on $\mathbb{A}$ with replay buffer $R$
end for
```

Utilizing offline RL [19, 20] can also alleviate some of the problems related to training on real systems by making use of existing datasets. The central issue of offline RL is the distribution shift between the behavior policy, which collected the dataset, and the trained policy [20]. Recent works constrain the trained policy to be similar to the behavior policy [36, 34, 17], modify the objective of the Q-function to be conservative with respect to samples not in the dataset [18, 38], or penalize uncertainty in rollouts of learned models [37, 29].

III. Method

The core components of HiS comprise the generation of parallel virtual trajectories, as well as the criteria to select among the additional transitions generated with HiS. This section introduces each of these components, as well as the HySR setup that HiS builds upon.

A. RL Preliminaries

We consider tasks formulated as a discounted Markov decision process, which is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, r)$, where $\mathcal{S}$ denotes the state space and $\mathcal{A}$ the action space. The transition dynamics $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ represent the probability of reaching state $\mathbf{s}_{t+1} \in \mathcal{S}$ after executing action $\mathbf{a}_t \in \mathcal{A}$ in state $\mathbf{s}_t \in \mathcal{S}$. Furthermore, the agent receives a scalar reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$ at each time step. The goal of RL is to find an optimal policy $\pi^\star$, which maximizes the discounted total return

$$R = \mathbb{E}_\pi \left[\sum_t^T \gamma^t r_t \right], \tag{1}$$

where T is the time horizon and $\gamma \in [0, 1)$ is the discount factor. The agent collects transitions $e_t = (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ by interacting with the environment and uses them to iteratively update its policy π. Many popular off-policy RL algorithms, such as SAC [12] or Deep Q-Networks (DQN) [24], store these transitions in a buffer $\mathcal{D}$ [21] and replay them when updating, for instance, the action value function $Q(\mathbf{s}_t, \mathbf{a}_t)$. This replay buffer is fixed-sized and contains the most recent transitions. A ring buffer, which replaces the oldest with the most recent transition, is a popular implementation. The Q-function update involves minimizing a variant of the TD error

$$\delta(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') = r + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a}), \tag{2}$$

where variations might include an alternative way to select the next action $\mathbf{a}'$ or an n-step TD error formulation $\delta_n = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n \max_{\mathbf{a}'} Q(\mathbf{s}_{t+n}, \mathbf{a}') - Q(\mathbf{s}_t, \mathbf{a}_t)$. The TD error represents a measure of surprise: the higher the absolute value of δ, the more does this particular transition influence the update of the Q-function. For this reason, the TD error metric is commonly used to establish a ranking among transitions, for example in prioritized sweeping [25, 27] and prioritized experience replay (PER) [33]. The Q-function loss employs the replay buffer $\mathcal{D}$ and the squared TD error

$$J(\theta) = \mathbb{E}_{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \sim \mathcal{D}} \left[\left(r + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q_{\theta^{\mathrm{old}}}(\mathbf{s}', \mathbf{a}') - Q_\theta(\mathbf{s}, \mathbf{a}) \right)^2 \right], \tag{3}$$

where the old parameters θ^{old} are kept fixed during optimization of the current policy parameters θ and transitions are sampled from $\mathcal{D}$ according to a distribution that traditionally is uniform but can vary such as in PER.

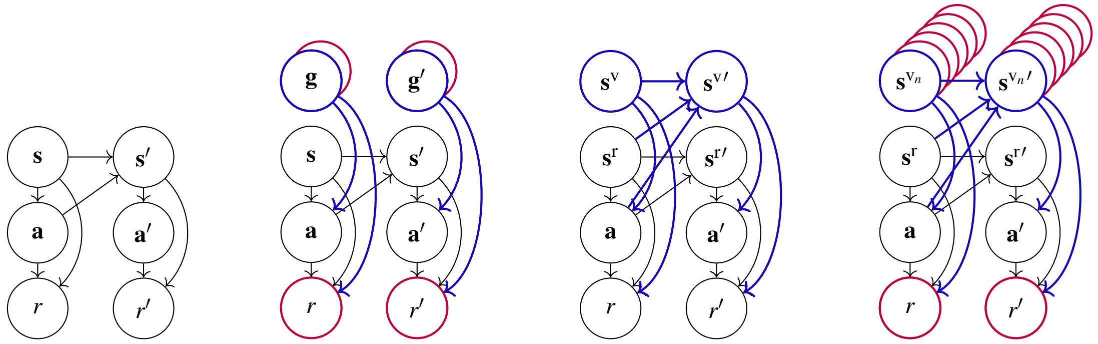
B. Hybrid Sim and Real Training

Hybrid Sim and Real (HySR) training [5] is the foundation for Hindsight States (HiS). Intuitively, the idea of HySR is to gain practical benefits by offloading the part of the task that is easier to model to the simulation. At the same time, the difficult part is kept real during training. More formally, HySR assumes that the Markovian state $\mathbf{s}$ splits into a real state $\mathbf{s}^{\mathrm{r}}$ and a virtual state $\mathbf{s}^{\mathrm{v}}$

$$\mathbf{s} = [\mathbf{s}^{\mathrm{r}}, \mathbf{s}^{\mathrm{v}}] \tag{4}$$

and that the dynamics governing the real part $p(\mathbf{s}^{\mathrm{r}}_{t+1}|[\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}}_t], \mathbf{a}_t)$ are more complex than those governing the virtual part $p(\mathbf{s}^{\mathrm{v}}_{t+1}|[\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}}_t], \mathbf{a}_t)$. To better transfer the learned policy after HySR training to the fully real setup, HySR keeps the complicated dynamics real and expects the virtual dynamics to be sufficiently accurate. Another requirement is that the virtual state does not influence the real state

$$p(\mathbf{s}^{\mathrm{r}}_{t+1}|[\mathbf{s}^{\mathrm{r}}_t, \mathbf{s}^{\mathrm{v}}_t], \mathbf{a}_t) = p(\mathbf{s}^{\mathrm{r}}_{t+1}|\mathbf{s}^{\mathrm{r}}_t, \mathbf{a}_t) \tag{5}$$



(a) MDP (b) HER (c) HySR (d) HiS

Fig. 2: Graphical models of a general MDP, HER, HySR, and HiS. All blue and red elements indicate a difference to the general MDP. Red indicates components that are relabeled in hindsight. HER extends the MDP with a goal state and relabels goals. HySR separates the state into a virtual and a real part. The additional arrows indicate the dynamics. HiS combines key ideas from HySR and HER that are state separation and dynamics as well as relabeling, respectively.

since mapping forces from the virtual to the real part can be intricate, especially onto complex or unknown dynamics.

The virtual dynamics do not necessarily simulate the part for the full duration of the training. Before the first contact between real and virtual part, HySR replays recorded data of the virtual part instead of simulating it to minimize the accumulation of model error. To that end, HySR first uniformly samples a virtual trajectory $\tau^{\text{v}^{\text{rec}}}$ from a database $\mathcal{D}_{\text{rec}}$ containing N_{rec} recorded trajectories.

$$\mathcal{D}_{\text{rec}} = \left[\mathbf{s}_1^{\text{v}_n^{\text{rec}}}, \dots, \mathbf{s}_T^{\text{v}_n^{\text{rec}}}\right]_{n=1}^{N_{\text{rec}}} \tag{6}$$

After contact of the virtual and the real part, the simulated dynamics determine the consecutive motion, as no information about the latter part can be deduced from the recordings. A complete HySR training trajectory,

$$\tau^{\text{HySR}} = [e_1^{\text{rec}}, \dots, e_{t_{\text{c}}}^{\text{rec}}, e_{t_{\text{c}}+1}^{\text{sim}}, \dots, e_T^{\text{sim}}], \tag{7}$$

hence, consists of both transitions replayed from the database $e_t^{\text{rec}} = \left([\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{v}^{\text{rec}}}], \mathbf{a}_t, r([\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{v}^{\text{rec}}}], \mathbf{a}_t), [\mathbf{s}_{t+1}^{\text{r}}, \mathbf{s}_{t+1}^{\text{v}^{\text{rec}}}]\right)$ and simulated transitions $e_t^{\text{sim}} = \left([\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{v}^{\text{sim}}}], \mathbf{a}_t, r([\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{v}^{\text{sim}}}], \mathbf{a}_t), [\mathbf{s}_{t+1}^{\text{r}}, \mathbf{s}_{t+1}^{\text{v}^{\text{sim}}}]\right)$, assuming that the contact happened at time t_{c}. During training, the policy samples actions conditioned on the complete state from Eq. (4), in which the information of whether the virtual parts are replayed or simulated is not included.

HySR is particularly suitable for applications where (i) the complexity of the virtual and real dynamics is unbalanced, (ii) training with the real instance of the virtual part is handy or safer, (iii) contact of the separated parts occurs relatively far into each roll-out to make better use of the recorded data, (iv) and the influence that the virtual part has on the real part is negligible and the effect in the opposite direction is known and transfers well to the real world. All these points apply to many ball games, such as football, cricket, baseball, or basketball, where an object has to reach a goal state through an interaction with the player. Although, the efficacy of HySR has only been shown for robot table tennis so far, training in the HySR setting is not limited to ball games. For instance, for tasks like slide-pushing or throwing objects (in contrast to picking them up), we could combine objects of different weights or materials with a single arm motion. Points (i) to (iv) also apply to tasks such as avoiding moving obstacles since contact should be avoided in the first place. For example, a robot waking in a cluttered and dynamic environment or an autonomous car driving through a crowded city.

C. Hindsight States

HySR also enables the generation of additional data, and *Hindsight States* (HiS) is our proposed way to leverage them for efficient training. HiS implements the idea that we can generate additional virtual data in the HySR setting. Specifically, it generates trajectories $\tau^{\text{HiS}} = [e_1^m, \dots, e_T^m]$ with $m = 1, \dots, M$ for each HySR trajectory $\tau^{\text{HySR}} = [e_1, \dots, e_T]$. For each of these M trajectories, we sample a different series of virtual states $\mathbf{s}_{1:T}^{\text{v}_m} \sim \mathcal{D}_{\text{rec}}$. We augment the HySR trajectory to reflect what would have happened if the virtual states were $\mathbf{s}_{1:T}^{\text{v}_m}$. In particular, we generate the hindsight transitions

$$e_t^m = \left([\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{v}_m}], \mathbf{a}_t, r([\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{v}_m}], \mathbf{a}_t), [\mathbf{s}_{t+1}^{\text{r}}, \mathbf{s}_{t+1}^{\text{v}_m}]\right) \tag{8}$$

for each transition $e_t = \left([\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{v}}], \mathbf{a}_t, r([\mathbf{s}_t^{\text{r}}, \mathbf{s}_t^{\text{v}}], \mathbf{a}_t), [\mathbf{s}_{t+1}^{\text{r}}, \mathbf{s}_{t+1}^{\text{v}}]\right)$ generated by HySR. The real states $\mathbf{s}_t^{\text{r}}, \mathbf{s}_{t+1}^{\text{r}}$ and the action $\mathbf{a}_t$ remain the same as in the HySR transition, but the virtual states are relabeled in hindsight with the different instances of the virtual states $\mathbf{s}_t^{\text{v}_m}, \mathbf{s}_{t+1}^{\text{v}_m}$, which are either taken from the pre-recorded or simulated data. The reward of the hindsight transitions is then calculated according to the relabeled state. Note that in this manner, HiS generates a wide bouquet of how the real part could have interacted with instances of the virtual object. For this reason, HiS provides valuable feedback on the quality of this particular action sequence in different situations. In the table tennis example depicted in Fig. 1c, the racket hits a whole cloud of balls with different speeds and trajectories; hence, the resulting balls arrive at very distinct landing points.

a) Picking Strategy: The naive approach in which all HiS trajectories are added to a replay buffer and sampled by an off-policy algorithm can be detrimental to the learning process. A critical influence on the sample complexity of RL algorithms is the degree of *off-policyness* of transitions in the replay buffer [11]. Off-policy data are challenging because they have been collected under multiple dissimilar policies, but the expectation in the return in Eq. (1) is computed w.r.t. the distribution induced by the current policy π. This distributional shift adds bias and variance to the estimation of the return and leads to incorrect estimations of the Q values. Typically, the size of the buffer determines the degree of off-policyness of the buffer since older transitions are more likely to stem from older and hence dissimilar policies [11]. The additional HiS trajectories add to the off-policyness of the buffer since the actions in the HiS trajectories were generated conditioned on the original HySR states instead of the relabeled states.

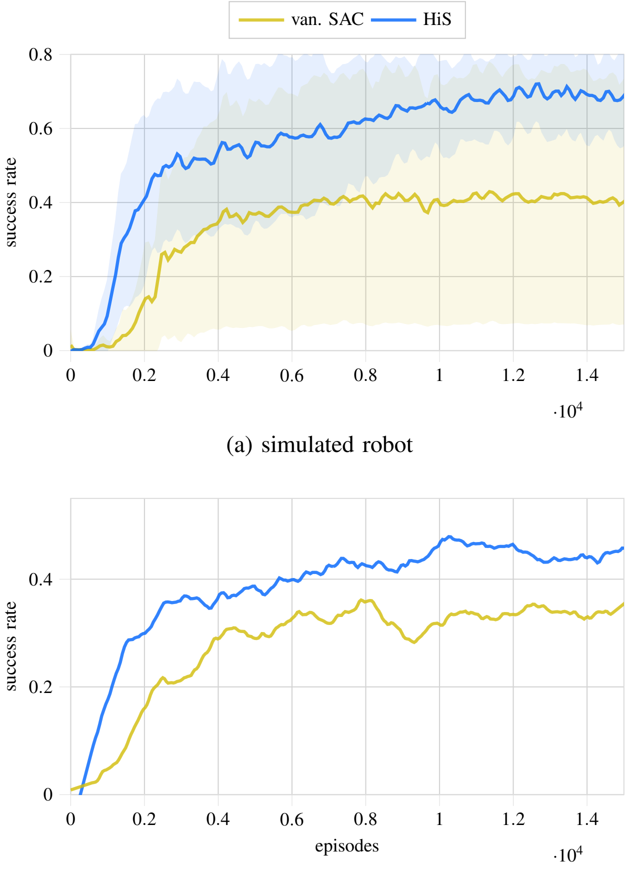
For this reason, HiS training includes a selection strategy that chooses the transitions that accelerate the learning process most. We propose to rank HiS transitions based on the reward $r(\mathbf{s},\mathbf{a})$ or the TD error $\delta(\mathbf{s},\mathbf{a},r,\mathbf{s}')$ from Eq. (2). The reasoning behind prioritizing higher rewards is to experience high rewards early on, which is especially useful in a sparse reward setting. The TD error corresponds to the amount of influence on the Q-function update. Hence, it is a sensible choice for ranking transitions. The selection process is based on the absolute value of the TD error $|\delta(\mathbf{s},\mathbf{a},r,\mathbf{s}')|$ since updating with both high and low-value transition ascribes to bringing the action value function closer to its optimal version Q^*. It is instrumental how these measures (TD error or reward) are applied to the transitions of the HiS trajectories. We study two ways: (i) applying the measure on each transition and (ii) on the whole set of transitions of a trajectory by taking the sum of the reward or TD error over the whole trajectory. The first implementation represents the straightforward approach, whereas the second way tests the hypothesis that all transitions of a trajectory with high value of the measure are significant. The idea of the latter point has been explored in the context of prioritized replay, where it turned out that transitions correlating in time have correlating TD errors [4].

Having defined the criteria to rank HiS transitions, we now present the methodology for determining the number of HiS transitions to be added to the replay buffer. For inclusion into the buffer, criterion c has to both (i) exceed a threshold ψ_c and (ii) be among the k_c best transitions according to the criterion. Case (i) prevents transitions from being added that are among the best of their peers for this round but too low in value. Condition (ii) controls the maximum amount of off-policy data in the replay buffer. Algorithm 1 summarizes HiS in detail.

b) Relation to HER: HiS shares some aspects with HER [1]. HER trains a goal-conditioned policy and relabels the desired goal with the actually achieved goal in hindsight. Both HER and HiS use hindsight data to learn more sample efficiently, and are especially useful in a sparse reward setting. In contrast, HiS relabels virtual states instead of goals.

Fig. 2 depicts the differences between a standard MDP (Markov decision process), HySR, HER, and HiS with respect to their corresponding graphical models. Note that the HER setting also allows for multiple goals in each transition and [9] introduced dynamics into the goal space of HER. As can be seen in Fig. 2, goals and dynamic goals [9] can be seen as special cases of virtual states. Therefore, our method can also be applied to the goal-conditional setting. The difference to HER, when applied to this setting, is that HER will sample goals such that they correspond to specific achieved goals (e.g., the goal sampled in hindsight is the final position of the robot gripper). HiS, however, will sample from the initial goal distribution and then use a prioritization strategy to select from the set of sampled goals. With enough goals sampled and a corresponding prioritization strategy, HiS could converge toward doing the same as HER (e.g., HiS samples enough goals, such that the prioritization strategy can pick goals that are similar to what the HER strategy would have chosen). In this setting, the additional complexity of HiS compared to HER seems unnecessary. However, in the general HySR setting, HER cannot be applied to virtual states, while the application of HiS is possible.

On a task, that has both a goal state and a virtual state, HER can be applied on top of HiS.



(b) real robot

Fig. 3: Results on the table tennis task. HiS increases sample-efficiency and asymptotic performance with respect to the number of robot steps.

c) Combining HiS and HER: HER and HiS are addressing the problem of sparsity during learning from two different directions. HER generates additional experiences that can be beneficial to the learning process because they solve the goal and have high rewards, but because only goals are relabeled, HER does not directly help exploring the state space. On the other hand, by creating additional virtual states of the environment, HiS helps exploring the state space. Contrary to HER, experiences created by HiS might not reach the goal or be of high reward. To get some of the benefits of HER in a goal-directed HySR setting, HiS could sample numerous hindsight trajectories until some of these trajectories also reach the goal. For a challenging and sparse task, however, this strategy might be computationally expensive or even infeasible.

Instead, a combination of HER and HiS can be beneficial for such tasks, and utilize the strengths of both. Practically, the combination can be implemented by following Algorithm 1, and adding complete HiS episodes into the replay buffer that is used by HER. HER relabels the goal of the HiS trajectories, and the resulting trajectories with relabeled virtual state and relabeled goal can then be sampled to optimize the policy.

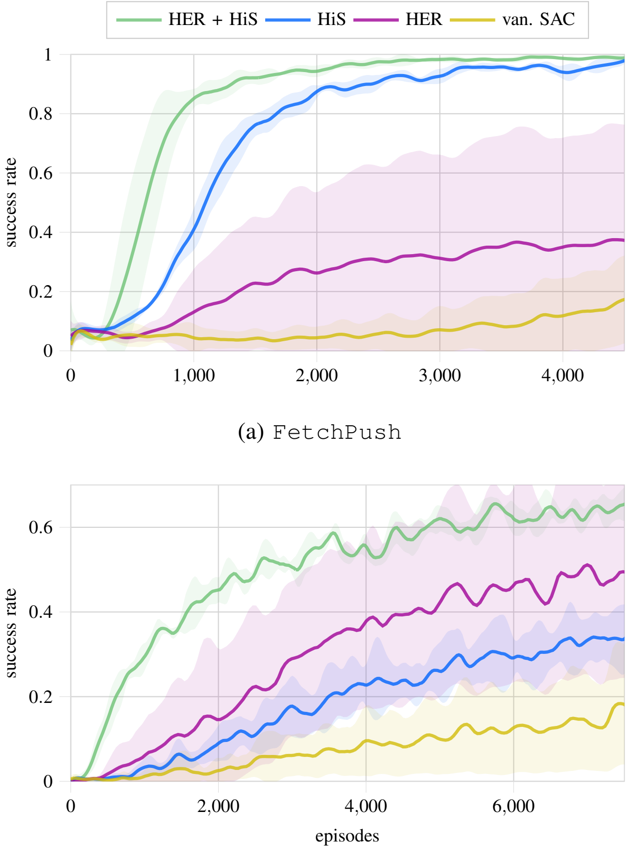
IV. Experiments

Our new method HiS promises improved sample efficiency in the HySR setting. The evaluation of HiS is carried out in two regimes: We run extensive experiments on 1) several simulated tasks, where the HySR assumptions apply, and 2) a challenging real robot table tennis task. Furthermore, we investigate to what extent HiS is useful in manipulation tasks and illustrate the efficiency of the combination of HER and HiS.

A. Applying HiS to the Original HySR Table Tennis Task

The work that originally introduced HySR [5] learned to play table tennis with a muscular robot and naturally separated between the hard-to-control soft muscular robot [7, 6] and the relatively simple ball model. Different from the original task, we refrain from using the hand-crafted dense reward function and use a simple sparse reward that returns one in case the ball lands within a circle of radius 40 cm and zero otherwise. To apply HiS, we replay 20 parallel instances of the virtual ball sampled from a set of 100 recorded ball trajectories. We apply HiS on top of SAC [12] and compare with vanilla SAC as a baseline. For trajectory selection, we use the reward criterion with threshold $\psi_c = 0.5$. The Appendix contains an in-depth description of the experiment details and the parameters of each experiment. It also contains an evaluation of different selection criteria.

The experiments depicted in Fig. 3 demonstrate that HiS enables us to learn this task with fewer robot time steps in both, the fully simulated and the HySR setting. The simulated experiments from Fig. 3a show the mean and variance of ten runs with different random seeds. HiS achieves a maximum average performance of appr. 70% success rate, whereas vanilla SAC reaches appr. 40%. HiS matches the asymptotic performance of SAC with appr. 29% of the samples that SAC needs to reach its asymptotic performance (2k and 7k episodes). Another observation is that the variance between different runs is smaller compared to SAC.



(a) FetchPush

(b) FetchSlide

Fig. 4: Results on the simulated Gym robotics tasks FetchPush and FetchSlide. HER and HiS both learn faster than SAC with respect to the number of robot steps. Combining HER and HiS gives the best results.

HiS is also more efficient when using the real robot. Fig. 3b shows one training run for vanilla SAC and HiS. HiS achieves around 45% success rate, while SAC reaches only appr. 35%. Similarly to the simulated experiment, HiS matches SAC's maximum success rate with appr. 38% of the number of samples collected on the real robot (3k and 8k episodes).

B. HySR and HiS on Manipulation Tasks

FetchPush and FetchSlide [1] are two benchmark manipulation tasks for goal-conditioned RL. These two tasks are based on existing robot hardware and simulated using MuJoCo [35]. In the Pushing task, the goal is to move a box to a target location. The robot's gripper is locked to prevent grasping. The Sliding task is slightly different: Here the objective is to move a puck to a goal position that is outside the robot's reach, thus, necessitating a sliding strategy. Both tasks satisfy the HySR assumptions and, hence, our method could be applied to learn a policy on a real robot. In both tasks, the manipulated object adheres to simple dynamics. The arm is rigid and small disturbances of the contact with the object are quickly compensated by a position controller. Thus, the influence of forces transferred from the object to the robot is small.

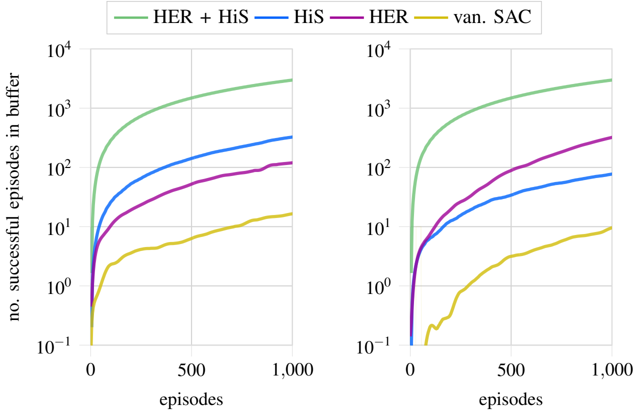
However, both tasks use a robot with relatively simple dynamics that can be simulated accurately. Sim-to-real training has been shown to be successful in solving these tasks [1] and would therefore be the better choice for this kind of system. However, when executing similar tasks with a more complex robot, for instance one equipped with soft parts, or a less accurate position controller, the larger discrepancies between simulation and reality would likely degrade the performance of the transferred policy. HySR would be well-suited for learning a policy on such a robot. Our experiments show that on top of the practical benefits coming with HySR, HiS could greatly improve sample efficiency for these tasks, especially when combined with HER. We will analyze the results in detail in the next subsection. Furthermore, we will show, that when learning a policy in a sim-to-real fashion, HiS can still help to speed up learning during the simulation stage.

C. Combining HER and HiS

In Section III-C, we have discussed the differences and similarities between HER and HiS. Due to the goal-conditioned nature of the `FetchPush` and `FetchSlide` tasks, they are suitable to experimentally investigate the relationship of HER and HiS. Fig. 4 shows the results on the Pushing and Sliding tasks. To apply HiS, we simulate 100 parallel instances of the virtual object. For virtual trajectory selection, we use trajectories where the robot moves the object. Similar to the table tennis task, we apply HiS and/or HER on top of SAC, which also serves as a baseline.

On both tasks, HER and HiS learn faster than vanilla SAC. HiS learns significantly faster than HER on the Pushing task, solving it almost perfectly in only 4k episodes, while HER surpasses HiS on the Sliding task. A possible interpretation of these results is that in tasks with sparser goals, such as the slide task with a larger table, HER generates more successful trajectories compared to HiS. Conversely, HiS performs well in tasks where objects rather than goals are more sparse. HER and HiS in combination achieve the best performance on both tasks. Similar to the table tennis task, we find that HiS alone as well as in combination with HER significantly reduces the variance between runs.

A key reason why hindsight methods seem to work well is that they generate additional positively labeled experiences. To illustrate this phenomena, we look at a simple metric, the number of successful trajectories put into the RL buffer during the first 1000 episodes of training. These trajectories are either collected on-policy or generated in hindsight if HiS and/or HER are involved. We filter out trivial episodes that are labeled successful but where the object does not change position. Such episodes contain little useful information. We note a strong correlation between the number of trajectories labeled successful in Fig. 5 and the results from Fig. 4. For both tasks, they show the same arrangement between the algorithms that we compare. This finding illustrates well how HER and HiS complement each other. For the two tasks studied, at the initial phases of the training, the combination of HER and HiS generates an order of magnitude more successful trajectories than HER or HiS individually. The intuition behind this is that HiS generates a lot of hindsight trajectories, and HER labels them successful.



(a) `FetchPush` (b) `FetchSlide`

Fig. 5: Number of successful trajectories added to the RL buffer within the first 1000 episodes of learning for the `FetchPush` and `FetchSlide` tasks. HER and HiS generates an order of magnitude more successful trajectories than HER or HiS on their own.

D. Efficiency of HiS for Entirely Simulated Tasks

HiS was originally designed to speed up learning in a hybrid sim and real setting, where real data is expensive and simulated data is cheap. Therefore, the experiments so far, even those run only in simulation have been evaluated on the number of steps that would have been executed on a real robot. But does it make sense to apply HiS in purely simulated tasks, e.g., in a sim-to-real setting for the simulation stage? In this setting, instead of minimizing the number of robot steps, the goal is usually to decrease total wall-clock time and computation. Compared to the baseline RL algorithm, HiS increases computation because it simulates extra virtual states, sorts this data according to the selection criterion and then feeds it into the replay buffer of the algorithm. However, as shown earlier, HiS also increases sample efficiency during learning.

Tab. I shows the evaluation of HiS regarding its efficiency for simulated tasks. We compare HiS to vanilla SAC and the combination of HER and HiS to only HER. To compare total computational cost, we restrict the learning to only one CPU (Intel Xeon W-2145 with 3.70GHz clock speed). The effect of increased computation depends on the task and the number of additional virtual objects. Wall-clock time for the same number of steps when using HiS increases only by about 2% for the table tennis task but by about 60% for the manipulation tasks. To quantify improvements in sample efficiency, we look at the number of robot time steps HiS takes to reach the asymptotic or final performance of vanilla SAC and similarly the number of robot time steps HER and HiS takes to reach the asymptotic or final performance of HER. As we have shown in Fig. 3 and Fig. 4, we achieve significant improvements by using HiS. Therefore, combining these two effects, even when evaluated in wall-clock time instead of in robot time steps, HiS compares favorably. In conclusion, we have shown that HiS can be beneficial for settings where we want to optimize a policy only in simulation.

For this reason, HiS' range of applications extends beyond tasks that fulfill all the HySR assumptions as discussed in Section III-B. In simulated environments, differences in the complexity of the dynamics of different parts of the task do not need to be considered. However, HiS also assumes that the virtual state should not affect the real state. For purely simulated tasks, these effects onto the robot can be modeled, but a remaining issue is that multiple objects, that do not influence each other, can exert forces onto the robot simultaneously. To overcome these discrepancies, one potential solution for tasks where contacts are sparse, could be to split the training into two stages: First, training with HiS and ignoring these discrepancies. This stage could help getting into contact with interesting objects. And later fine-tuning those contacts without HiS.

V. Conclusion and Future Work

In this work, we presented Hindsight States (HiS), a method for sample-efficient training in a hybrid sim and real setup, where multiple instances of the virtual part of a task are paired with the single real part. HiS leverages this additional data by selectively choosing the data to train on. We evaluated HiS on a variety of tasks in simulation and showed that it improves sample efficiency on a complex real-world muscular robot task. We further showed that combining HER and HiS leads to even better performance than applying each method individually.

A limitation of our work is the requirement that the virtual part does not influence the real part. In many tasks, this influence cannot be neglected, particularly when dealing with heavier objects. One way around, could be to map the forces from the simulation back onto the joint torques of a real robot.

Another strategy would be to split learning into multiple stages. Stages where the condition is fulfilled could be learned with HiS and completely real training otherwise. As an example, for a grasping task, the stage where the robot learns to get to the position just before the gripper touches the object could be learned using HiS.

HiS's pronounced impact during the initial exploration stage of the training explains a large part of the lower variability compared to the baselines (SAC with and without HER). In absence of HiS, some runs encounter complete exploration failure, contributing to the increased variance. The potential of HiS during the later stages of the training warrants further investigation and refinement. For instance, in complex tasks that demand extensive interactions, a strategy of intermittently resampling virtual objects throughout the episode — contrary to the current approach of resampling only at the beginning — could improve results. This approach leverages the possibility that objects resampled in close proximity to the robot can generate more interesting experiences, for example, due to the higher likelihood of additional contacts.

TABLE I: Evaluation of HiS regarding its efficiency for purely simulated tasks. There are two effects. First, using HiS alone or HiS in combination with HER increases total wall-clock time to perform the same number of time steps. However, secondly, on average HiS needs less of those time steps to reach matching performance (here: the asymptotic or final performance of SAC as evaluated during the experiments), and HER and HiS also need less time steps to reach matching performance (the asymptotic or final performance of HER as evaluated during the experiments). For the tasks studied in this work, the second effect is larger, which results in less average wall-clock time to reach the same performance, even for learning purely in simulation.

	Task		
	Pushing	Sliding	Table Tennis
Comparison of HiS to SAC			
total wall-clock time	163.9 %	156.3 %	102.3 %
robot time steps for matching performance	16.7 %	50.8 %	30.7 %
wall-clock time for matching performance	27.4 %	79.4 %	31.4 %
Comparison of HER+HiS to HER			
total wall-clock time	165.6 %	162.3 %	
robot time steps for matching performance	11.7 %	42.6 %	
wall-clock time for matching performance	19.4 %	69.1 %	

The focus of this work was to devise a more sample efficient training scheme in the HySR setting. However, as shown in Section IV-D, HiS can also be beneficial for purely simulated tasks. Finding the best strategies to apply HiS to simulated tasks, especially to those that do not satisfy the HySR conditions holds a lot of potential.

Future work can also focus on combining HiS with curriculum learning by using data augmentations. Data augmentations such as transformations as well as perturbations of the virtual objects, which itself can be physically plausible, facilitates generalization. In addition, one can adapt the laws of physics of the virtual part such as gravity or the speed of time to change the difficulty of the whole task. In this manner, a curriculum could be added that, e.g., slows down or accelerates time during the training, so the robot reaches high reward regions faster. Subsequently, adjusting time gradually to its true speed toward the end of the training might improve the transfer of the performance to the real world.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in Neural Information Processing Systems*, 30, 2017.

[2] Chenjia Bai, Peng Liu, Wei Zhao, and Xianglong Tang. Guided goal generation for hindsight multi-goal reinforcement learning. *Neurocomputing*, 359:353–367, 2019.

[3] Sofanit Wubeshet Beyene and Ji-Hyeong Han. Prioritized hindsight with dual buffer for meta-reinforcement learning. *Electronics*, 11(24):4192, 2022.

[4] Marc Brittain, Josh Bertram, Xuxi Yang, and Peng Wei. Prioritized sequence experience replay. *arXiv preprint arXiv:1905.12726*, 2019.

[5] Dieter Büchler, Simon Guist, Roberto Calandra, Vincent Berenz, Bernhard Schölkopf, and Jan Peters. Learning to play table tennis from scratch using muscular robots. *IEEE Transactions on Robotics*, 38(6):3850–3860, 2022.

[6] Dieter Büchler, Roberto Calandra, and Jan Peters. Learning to control highly accelerated ballistic movements on muscular robots. *Robotics and Autonomous Systems*, 159: 104230, 2023.

[7] Dieter Büchler, Heiko Ott, and Jan Peters. A Lightweight Robotic Arm with Pneumatic Muscles for Robot Learning. In *International Conference on Robotics and Automation (ICRA)*, Stockholm, May 2016. doi: 10.1109/icra.2016.7487599.

[8] Shirli Di-Castro, Dotan Di Castro, and Shie Mannor. Sim and real: Better together. *Advances in Neural Information Processing Systems*, 34:6868–6880, 2021.

[9] Meng Fang, Cheng Zhou, Bei Shi, Boqing Gong, Jia Xu, and Tong Zhang. DHER: Hindsight experience replay for dynamic goals. In *International Conference on Learning Representations*, September 2018.

[10] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. *Advances in Neural Information Processing Systems*, 32, 2019.

[11] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[13] D. F. B. Haeufle, M. Günther, A. Bayer, and S. Schmitt. Hill-type muscle model with serial damping and eccentric force–velocity relation. *Journal of Biomechanics*, 47(6):1531–1536, April 2014. ISSN 0021-9290. doi: 10.1016/j.jbiomech.2014.02.009.

[14] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pages 1480–1490. PMLR, 2017.

[15] 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.

[16] Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *International Conference on Robotics and Automation (ICRA)*, pages 6008–6014. IEEE, 2019.

[17] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] 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.

[19] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. *Reinforcement learning: State-of-the-art*, pages 45–73, 2012.

[20] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv:2005.01643 [cs, stat]*, November 2020. URL http://arxiv.org/abs/2005.01643. arXiv: 2005.01643.

[21] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992. Publisher: Springer.

[22] Peng Liu, Chenjia Bai, Yingnan Zhao, Chenyao Bai, Wei Zhao, and Xianglong Tang. Generating attentive goals for prioritized hindsight reinforcement learning. *Knowledge-Based Systems*, 203:106140, 2020.

[23] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26), 2019.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. URL http://arxiv.org/abs/1312.5602.

[25] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Mach Learn*, 13(1):103–130, October 1993. ISSN 1573-0565. doi: 10.1007/BF00993104. URL

https://doi.org/10.1007/BF00993104.

[26] Fabio Muratore, Fabio Ramos, Greg Turk, Wenhao Yu, Michael Gienger, and Jan Peters. Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, page 31, 2022.

[27] Jing Peng and Ronald J. Williams. Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 1(4):437–454, March 1993. ISSN 1059-7123. doi: 10.1177/105971239300100403. Publisher: SAGE Publications Ltd STM.

[28] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, May 2018. doi: 10.1109/ICRA.2018.8460528.

[29] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pages 1154–1168. PMLR, 2021.

[30] Antonin Raffin. RL Baselines3 Zoo. github.com/DLR-RM/rl-baselines3-zoo, 2020.

[31] Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Jürgen Schmidhuber. Hindsight policy gradients. *arXiv:1711.06006 [cs]*, February 2019. URL http://arxiv.org/abs/1711.06006. arXiv: 1711.06006.

[32] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. *Advances in Neural Information Processing Systems*, 32, 2019.

[33] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[34] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[35] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[36] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[37] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. *arXiv:2005.13239 [cs, stat]*, May 2020. URL http://arxiv.org/abs/2005.13239. arXiv: 2005.13239.

[38] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization. *Advances in Neural Information Processing Systems*, 34: 28954–28967, 2021.

[39] Jingwei Zhang, Lei Tai, Peng Yun, Yufeng Xiong, Ming Liu, Joschka Boedecker, and Wolfram Burgard. VR-goggles for robots: Real-to-sim domain adaptation for visual control. *IEEE Robotics and Automation Letters*, 4 (2):1148–1155, 2019.

[40] Rui Zhao and Volker Tresp. Energy-based hindsight experience prioritization. In *Conference on Robot Learning*, pages 113–122. PMLR, 2018.

[41] Wei Zhu, Xian Guo, Dai Owaki, Kyo Kutsuzawa, and Mitsuhiro Hayashibe. A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–16, 2021. ISSN 2162-2388. doi: 10.1109/TNNLS.2021.3112718.