EMERGENT DEXTERITY VIA DIVERSE RESETS AND LARGE-SCALE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

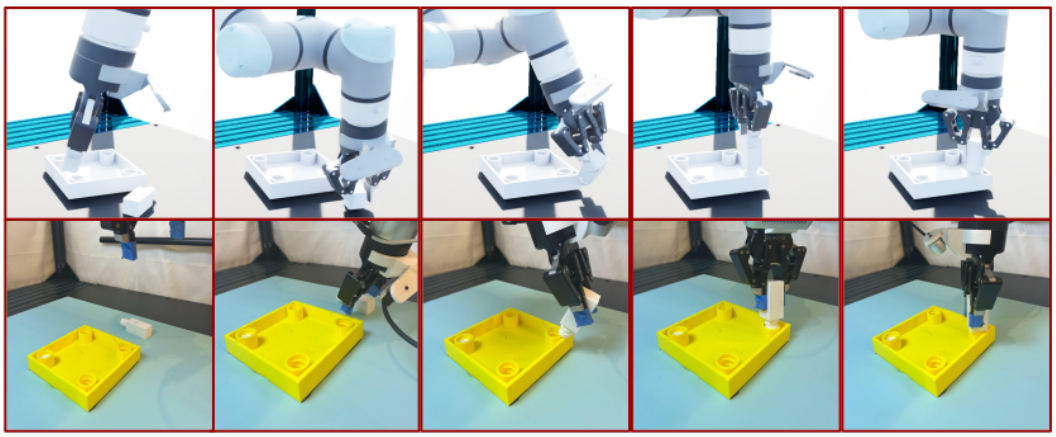


Figure 1: `OmniReset`generates a large, diverse dataset in simulation which can be used to solve complex real-world manipulation tasks when cotrained with a small amount of real-world data.

ABSTRACT

Reinforcement learning in GPU-enabled physics simulation has been the driving force behind many of the breakthroughs in sim-to-real robot learning. However, current approaches for data generation in simulation are unwieldy and task-specific, requiring extensive human effort to engineer training curricula and rewards. Even with this engineering, these approaches still struggle to reliably solve long-horizon, dexterous manipulation tasks. To provide a seamless tool for robotic data generation in simulation, we introduce a simple framework that enables on-policy reinforcement learning to reliably solve an array of such tasks with a single reward function, set of algorithm hyper-parameters, no auto-curricula, and no human demonstrations. Our key insight is careful usage of diverse simulator resets for simplifying long-horizon exploration challenges. Our proposed system, `OmniReset` automatically generates these resets with minimal human input and gracefully scales with compute to solve dexterous, contact-rich long-horizon tasks. `OmniReset` outperforms baselines on easier versions of our tasks, and scales to tasks with complexities beyond the reach of existing techniques. Finally, we use this data-generation methodology to create a large dataset of trajectories in simulation, and show that augmenting it with a small amount of real-world data enables successful real-world transfer for complex manipulation tasks. Project webpage: https://sites.google.com/view/omnireset

1 INTRODUCTION

Reinforcement learning (RL) in massively GPU-parallelized simulation environments (Mittal et al., 2023; Todorov et al., 2012) has powered recent success stories in robotics (Akkaya et al., 2019; Hwangbo et al., 2019). The typical sim-to-real paradigm first trains a state-based policy with RL and then distills this expert into a deployable visuomotor policy. In principle, once a robust expert policy has been obtained, this framework enables generating vast datasets for robot learning with

extensive *coverage* over initial conditions, tasks, and physical parameters. However, in practice, obtaining such a capable expert requires extensive human effort in the form of task-specific reward engineering or hand-crafted training curricula (Akkaya et al., 2019; Handa et al., 2023). Even with this engineering, current RL techniques struggle to reliably scale to long-horizon dexterous manipulation tasks Heo et al. (2023).

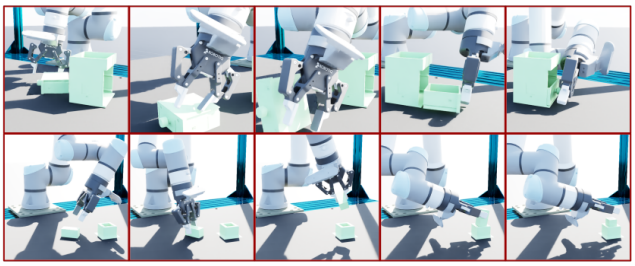


Figure 2: `OmniReset` applied to drawer assembly and peg insertion. We highlight drawer assembly where `OmniReset` learns to push, flip, and insert the drawer from scratch.

The central challenge blocking progress on such problems is the high-dimensional search space inherent to long-horizon continuous control problems. A unifying theme behind prior works which leverage RL for dexterous manipulation is to lessen the exploration burden placed on RL by implicitly *restricting the space of behaviors the algorithm must search over*. Current state-of-the-art approaches leverage demonstrations generated by either human experts Bauza et al. (2025) or motion planners Khandate et al. (2023) to bootstrap exploration. These approaches leverage a wide array of techniques such as resetting from the demonstrations (Bauza et al., 2025), adding demo-following auxiliary rewards (Peng et al., 2018), or adding explicit Behavior-Cloning terms to RL objectives (Nair et al., 2018; Tang et al., 2024). Even when leveraging these priors, these approaches often still require auto-curricula to stabilize training (Bauza et al., 2025; Tang et al., 2024), adding significant algorithmic complexity.

We contend that these approaches have two key limitations, and instead argue that *restricting reinforcement learning as little as possible* is the key to unlocking its full potential for dexterous manipulation. The first limitation is that trajectories generated by human experts and motion planners are overwhelmingly biased towards *high quality behaviors* (e.g. obtaining "good" grasp points on an object). However, during training the current RL policy will inevitably produce sub-optimal behaviors (e.g. grasping objects at points from which it is infeasible to solve the task). Unfortunately, expert trajectories alone do not provide RL algorithms with the coverage over behaviors that is required to learn to go from sub-optimal behaviors to optimal behaviors. Secondly, these approaches will inevitably have gaps in coverage over the state-space. It is generally too costly to obtain human demonstrations with dense coverage over a wide array of initial conditions, while motion planners and other heuristics can often struggle to find viable plans for complex dexterous tasks.

We introduce `OmniReset`, a simple, scalable approach for generating diverse reset distributions for manipulation tasks that enables PPO (Schulman et al., 2017) to master the tasks depicted in Figure 2 without demonstrations, curricula or task-specific reward engineering. The philosophy behind our approach is simple: bypass exploration challenges as much as possible by using resets to approximately cover all 'reasonable' states on the path to the goal, and then let dexterous behaviors naturally arise from the optimization. Attempting to cover this space of behaviors may initially appear intractable, however, our key insight is that the space of 'reasonable' manipulation behaviors is actually surprisingly small. Indeed, effective manipulation can be roughly broken down into combinations of reaching towards objects, grasping, non-prehensile motions such as pushing or flipping, and contact-rich behaviors that occur near goal states such as screwing or insertion.

In short, `OmniReset` introduces a set of scalable techniques for automatically generating diverse reset distributions over these behaviors using minimal high-level problem specifications from the user. Whereas prior works on manipulation found that naively scaling the number of parallel environments provides minimal performance gains for existing RL algorithms Singla et al. (2024), we demonstrate that pairing the diverse resets from `OmniReset` with large-scale simulation enables truly dexterous behaviors to emerge when training RL from scratch. As we detail in our experiments, `OmniReset` provides RL with enough information to automatically solve problems backwards from the end of the task, unlocking the true potential of large-scale approximate dynamic programming and eschewing the need for explicitly designed curricula which attempt to mimic these learning dynamics (Florensa et al., 2017).

Our contributions are as follows - (1) We introduce `OmniReset`, a simple, scalable framework for generating diverse but meaningful resets for training manipulation policies with RL. `OmniReset` bypasses the need for prior demonstrations, auto-curricula and RL hyper-parameter or reward tuning, (2) We demonstrate that `OmniReset` can effectively leverage large-scale simulation to naturally scale to dexterous tasks beyond the reach of existing approaches, (3) We show this generated data can be used to bootstrap visuomotor policy learning, be transferrable to the real-world with a small amount of real-world co-training.

2 Related Work

Exploiting Resets and in Reinforcement Learning: Exploiting simulator resets for RL is a natural idea which has been explored in many contexts. Prior theoretical works (Kakade & Langford, 2002) have suggested more uniform sampling over initial states, but do not provide practical algorithms. The primary focus of many works is to make learning more tractable by generating an explicit curriculum over resets(Tang et al., 2023; Dennis et al., 2020; Bauza et al., 2025), for instance through a "reverse-curriculum" of states going backwards from the goal (Florensa et al., 2017) or using a learned dynamics model to propose viable resets (Edwards et al., 2018; Ivanovic et al., 2019). In contrast, a second category of methods leverage *demonstrations* (whether human or automatically generated) to generate feasible pathways to the goal (Tao et al., 2024; Resnick et al., 2018; Salimans & Chen, 2018; Bauza et al., 2025; Tang et al., 2024). In contrast to these prior works, we show that neither human demonstrations, nor a curriculum is needed, but rather that the simple approach for generating diverse resets in `OmniReset` naturally scales to various long-horizon manipulation problems without added algorithmic complexity.

Exploration Strategies for Reinforcement Learning: RL practitioners have designed a variety of exploration strategies to effectively uncover goal-reaching paths for a *fixed set of initial conditions*, with uninformative rewards. A major line of work is bonus-based exploration, where agents receive intrinsic rewards for visiting novel or unpredictable states. Count-based methods reward visits to rarely seen states (Ostrovski et al., 2017; Bellemare et al., 2016; Martin et al., 2017), while curiosity-based methods provide bonuses based on prediction errors (Burda et al., 2018; Sancaktar et al., 2022; Pathak et al., 2017). Other approaches (Osband et al., 2016; 2019; Russo et al., 2018) promote temporally correlated exploration by injecting stochasticity at the policy or value-function level. Finally, diversity-driven methods optimize for behavioral variety (Eysenbach et al., 2019; Rajeswar et al., 2023). These approaches are complimentary to our work; our main contribution is demonstrating that large scale-scale parallelization and resetting schemes lead to the emergence of surprising levels of dexterity without the need for advanced exploration incentives.

Leveraging Demonstrations: An alternative is to increasingly rely on human demonstrations and imitation learning to overcome difficult long-horizon exploration. Approaches include adding auxiliary BC loss terms to RL objectives (Nair et al., 2018; Hester et al., 2018; Rajeswaran et al., 2018), simply adding demonstrations to the replay buffer (Vecerik et al., 2017), and introducing reward shaping terms which encourage RL agent to follow demonstrations (Tang et al., 2024; Reddy et al., 2019; Koprulu et al., 2024; Peng et al., 2018). Other works have sought to squeeze more information out demonstrations by automatically translating existing demonstrations to new initial conditions and scenes (Mandlekar et al., 2023) or by robustifying BC policies by promoting recovery behavior (Ke et al., 2023; Ankile et al., 2024). While our work complementarily pushes the limits of what behaviors can be learned entirely from scratch, we expect that demonstrations (when available) can also be incorporated into our framework to further accelerate learning.

3 Generating Diverse Resets for Learning Dexterous Manipulation

This section introduces `OmniReset`, a scalable framework for generating diverse reset distributions for complex contact-rich manipulation tasks from minimal high-level task descriptions from the user. We show that these reset distributions, when used correctly, can enable simple RL algorithms to produce surprisingly robust, long-horizon dexterous manipulation behavior with minimal task-specific engineering. The core tenet behind our framework is to reduce the burden on the user, enabling them to specify problems in a parsimonious fashion, while spending as little time as pos-

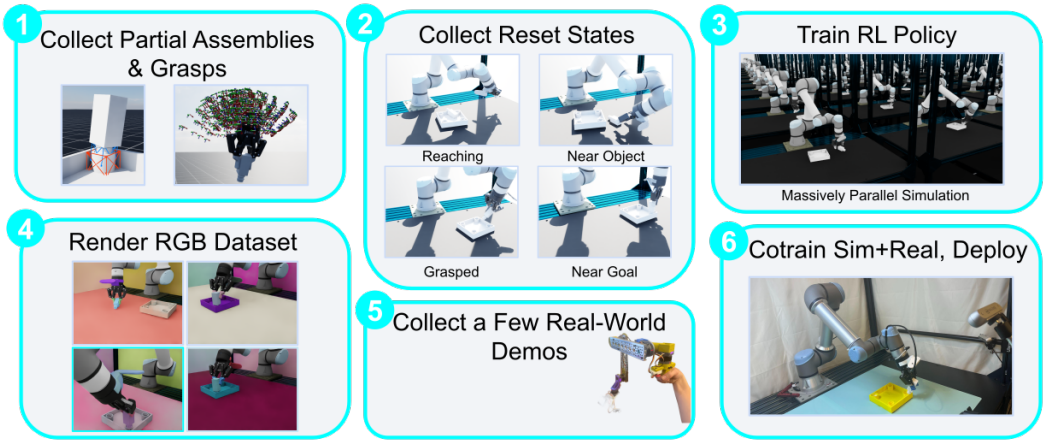


Figure 3: **Sim-to-Real Pipeline with `OmniReset`** (1) After generating partial assemblies and grasps from the simulator, (2) we collect reset states: reaching, near object, grasped, and near goal. (3) We then train a state-based RL policy initialized from these reset states, which is used to (4) train student-teacher distillation to get a RGB policy. (5) By finetuning this RGB policy on a mix of simulation data and small set of real demonstrations, (6) we deploy the RGB-based policy in the real world.

sible tweaking rewards and algorithm hyperparameters. Towards this end, we begin by defining notation used throughout the paper and formalizing the pieces of information the user is asked to specify for each task.

Notation and Reinforcement Learning Problem: We will let $s \in \mathcal{S}$ denote the state space of the simulator, and let $a \in \mathcal{A}$ denote the action taken by the robot. We will denote the transition dynamics of the simulator by $s' \sim P(\cdot|s)$, and we will use the notation $a \sim \pi(\cdot|s)$ to denote control policies for the robot. Note that in simulation we will learn everything from compact state representations, only moving to vision-based distillation for transfer to the real-world (Sec. 4).

Requirement 1 *The user has specified a set of goal configurations $\mathcal{G} \subset \mathcal{S}$ for the task.*

Requirement 2 *The user has provided a bounding box in the robots workspace $\mathcal{W} \subset \mathcal{S}$ specifying the range over which the robot is expected to successfully manipulate objects.*

Requirement 3 *The user has specified a bounding box of near-goal states $\mathcal{G} \subset \mathcal{NG} \subset \mathcal{S}$ which contains the goal states and contact rich states the robot will encounter when solving the task.*

Requirements 1 and 2 are standard components of defining manipulation tasks in simulation. We provide an expanded discussion on Requirement 3 below, but Figure 4b provides examples illustrating how these near-goal states are very intuitive to design for new tasks. Roughly speaking, we only ask the user to define 'points of interest' where contact-rich behaviors are expected to occur (in no specific order), which we argue is a relatively weak ask from the user.

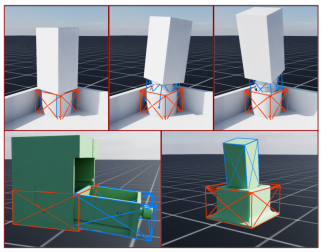
Reinforcement Learning Problem: `OmniReset` uses the specifications defined by Requirements 1-3 to automatically construct reinforcement learning problems for dexterous manipulation tasks. We formalize this as a Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho)$, where r is the reward and γ is the discount factor, and $s_0 \sim \rho$ is the distribution over initial conditions. Of critical importance in our system is the initial state distribution ρ. While this is typically assumed to be a fixed distribution provided by the user, we propose that in `OmniReset`(Sec. 3.1), we will specifically generate an extremely broad distribution of interesting reset-states ρ. Given this generated set of initial states, we can then optimize for the discounted sum of rewards - $J(\pi) = \mathbb{E}_{s_0 \sim \rho}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$, where the expectation is also taken w.r.t the actions generated by π. In practice, we also parameterize the MDP with dynamics parameters such as friction coefficients, masses, etc. We find empirically that the right choice of generating ρ makes the above-mentioned objective very effective at solving complex, dexterous, long-horizon problems. In Sec. 3.1, we will show how to practically generate a broad reset distribution ρ from minimal user specifications, and then in Sec. 3.2 we will show how ρ can be used with large-scale RL to learn complex behavior.

3.1 GENERATING DIVERSE RESETS

We now describe the core techniques which comprise `OmniReset` and enable users to construct diverse, meaningful reset distributions for dexterous manipulation by providing the high-level specification defined by Requirements 1-3. Before describing the reset distributions in detail, we first discuss several key considerations which guided the development of our pipeline.

Challenges in Generating Feasible Resets: Generating *feasible* reset states in physics simulation environments – i.e. states where contact constraints are satisfied – is non-trivial. Failing to do so can lead to objects getting 'stuck' inside of one another or explosions in velocities as the simulator attempts to resolve contacts. Such pathologies are detrimental to learning and must be systematically filtered out. Collision checkers can be used to filter out invalid state proposals, but there are still several key challenges to consider - **(1)** Collision checkers are subject to fundamental trade-offs between accuracy and computation time, especially for complex non-convex geometries, **(2)** A large number of invalid states may need to be rejected before a valid sample is found, especially true for high-precision problems, **(3)** Massively parallelized simulators (Mittal et al., 2023) take environment steps in lock step to maximize GPU utilization, leading to pipelining challenges. Thus, we use an initial offline phase and collision checker to generate a dataset D of feasible resets using the sampling strategies described below.

(a) **Grasp Sampling.** We display the grasp poses sampled for the table leg. The grasp sampling ranges are broad, moderate, and narrow from left-to-right. Our method uses the broad range.



(b) **Near-Goal States.** The user specifies bounding boxes around (portions of objects) which comprise Near-Goal States.

Figure 4: Tools for reset generation

Generating Diverse Reset Distributions: We compose the set of reset-states D from four different components using the specifications provided by the user in Requirements 1-3 - $D = D^R \cup D^{NO} \cup D^G \cup D^{NG}$, where D^R is a *reaching* dataset, D^{NO} is a near-object dataset, D^G is a *grasped* dataset, and D^{NG} is a *near-goal* dataset. We then define the reset distribution $\rho \sim Uniform(D)$. Before generating these datasets, we first use the built in grasp sampler from IsaacLab (Mittal et al., 2023) (see Figure 4a) to generate a large dataset of feasible grasp points on the objects, recording the relative pose between the gripper and the object. We then generate proposed resets for each of the distributions as follows:

1) Reaching Resets: We spawn the object at random posses slightly above the tabletop within the workspace $\mathcal{W}$, and allow them to come fall and come to rest.

2) Near-Object Resets: We first spawn the objects at the states from D^R. We then select a grasp point on the object, spawn the gripper at that point with some pose offset randomization, and spawn the gripper in open or closed configurations with 50% probability each.

3) StableGrasp Resets: We randomly spawn the objects that are to be manipulated in the air at random poses throughout $\mathcal{W}$. We then select a grasp point on the object and spawn the gripper at that point, with the gripper closed on the object. We run the simulation for several steps to ensure the grasp is stable and objects have settled.

4) Near-Goal Resets: We randomly sample objects in the set of near-goal states $\mathcal{NG}$. Next we step the simulator while applying small random forces for two seconds, taking inspiration from (Tang et al., 2024). Finally, we then move the gripper to one of the precomputed grasp points. We found that the jostling provided by the second step helped provide more uniform coverage over $\mathcal{NG}$ as the simulator resolves contacts.

These reset distributions are depicted pictorially in Fig 3 (Panel 2). Together, we found that each of these datasets was necessary for arriving at a single reset formulation which enables RL to solve the diversity of tasks we consider. Together, the reaching dataset D^R and grasped dataset D^G provide broad coverage over the state-space and enable the RL algorithm to learn effective policies over all of the workspace $\mathcal{W}$ with no gaps in coverage. We found that the near object dataset D^{NO} and near-goal dataset D^{NG} were necessary to provide dense coverage over contact rich states, enabling behaviors such pushing, flipping, screwing, and insertion to emerge naturally from RL training.

As we depict in Figure 4b, we found that specifying the near-goal states $\mathcal{NG}$ was typically quite natural. In practice, to specify $\mathcal{NG}$ we simply specified a bounding box around the parts of different object which must inherently interact in order to reach the goal, such as the threads on the table leg and the corresponding hole it must be screwed into, or a drawer and the corresponding slot on a dresser. The goal of this work is to understand *what properties reset distributions must have to enable scalable RL training for contact rich tasks*, and to take an initial step towards making the generation of these resets as automatic as possible. While `OmniReset` still requires a small amount of human effort to specify new task descriptions, our experimental results demonstrate that this approach enables us to scale to tasks far beyond the complexity of current methods.

3.2 Design Decisions for Scalable RL Training

Once `OmniReset`has generated a broad set of initial reset-states, these are used to seed a large-scale RL process to learn dynamic control policies for performing long-horizon, contact-rich manipulation. In this section, we discuss the design decisions that are necessary for an RL algorithm to solve the complexity of tasks we consider, when combined with the resets proposed in Sec. 3.1.

Sufficient Reset Coverage: We consistently found that increasing the diversity of meaningful reset states the policy is exposed to simplified training and yielded more capable, robust policies. For example, we found generating a wide range of sub-optimal grasps for the task accelerated and stabilized training substantially. We ablate this design decision in more detail in Sec.5.

Scaling Parallel Environments: Scaling the number of parallel environments is an equally important decision. When combined with increasing reset diversity, we found that scaling the batch size used by PPO consistently simplified the process of training a performant policy, enabling us to obviate curricula and task dependent rewards. We ablate the number of parallel environments below.

Asymmetric Actor-Critic: We asymmetric actor-critic approach (Pinto et al., 2017) for our learning architecture. The actor observations include a history of the five previous time-steps for the state of the robot, the poses of all objects in the scene, and the previous actions taken by the policy. The critic takes in these observations as well as additional privileged parameters of the environment. We found that conditioning the actor on larger observation spaces led to less stable training and led us to only provide this information to the critic.

Generalized State-Dependent Exploration Noise: We employ the policy noise parameterization from (Raffin et al., 2022). In short, gSDE has a separate prediction head which determines the gaussian exploration noise at each time-step and is conditioned on the features of the final layer of the policy network. This approach enables the actor to learn different temporally-correlated exploration strategies in different regions of the state-space, crucial for solving heterogeneous multi-stage tasks.

Reward Structure: Employing the previous design decisions enabled us to converge on a simple, common reward function for all tasks of the form:

$$r(s_t, a_t) = \text{Success}(s_t) + \text{DenseSuccess}(s_t) + \text{Dense}(s_t) + \text{Smooth}(s_t, a_t), \tag{1}$$

where $\text{Success}(s_t)$ is a binary $\{0, 1\}$ reward which is activated on the goal set $\mathcal{G}$, $\text{DenseSuccess}(s_t) = 0.1 * [\exp(-d^o_{xyz}) + \exp(-d^o_{rp})]$ where d^o_{xyz} and d^o_{rp} are l_2 distances on object displacement from the goal, $\text{Dense}(s_t) = 1 - \tanh d^g_{xyz}$ where d^g_{xyz} is l_2 distance on displacement from gripper to object origin, and $\text{Smooth}(s_t, a_t) = -0.0001 * [\|a_t\| + \|a_t - a_{t-1}\| + \|v_t\|]$ is an action penalty encouraging smooth robot actions. We found that each of these terms was necessary to stabilize training. However, performance of the algorithm was generally insensitive to the choice of weights for the different components, which are kept fixed across experiments.

4 Distillation and Real-World Transfer

We highlight the utility of our learned data generation policies by performing distillation into visuomotor policies that can be deployed directly in the real world from pixels. The robot has access to 224×224 RGB camera images from a front-facing camera, a side-facing camera an a wrist camera. We use the photo-realistic rendering capabilities of IsaacLab (Mittal et al., 2023) to generate a visuomotor dataset of $10,000$ expert rollouts and action labels while recording the associated camera views, which can then be used to perform standard student-teacher training Chen et al. (2021). The student RGB policy is composed of a Gaussian-MLP policy head, uses a pre-trained ResNet-18 encoder as a visual backbone, and takes in the last five observations and actions.

Visual Randomizations To tackle the visual gap, we following DextrAH-G (Singh et al., 2025) to create realistic scenes. Every four seconds of simulation steps, we randomize dome-light HDRI backgrounds and light intensity. We additionally randomize object color, robot color, workspace colors (table, background curtains). For better sim2real transfer, we calibrate our three cameras in real and match their pose in simulation. We additionally randomize camera pose and FOV slightly to be robust to calibration error. In addition to these randomizations, we also add data augmentations such as color jitter, gaussian blur, random grayscale, and gaussian noise.

Co-Training with Real World Data: To enable successful transfer, we developed a simple co-training recipe for improving the simulation-trained policy by incorporating a small number number of demonstrations collected in the real world. While direct finetuning on real world data leads to poor performance, we experiment with different co-training recipes - finetuning on different combinations of data from simulation and reality and freezing different components of the network. The best results we obtained are reported in more detail in Sec.5, and involve freezing the ResNet encoder during finetuning with a data mixture of 95% simulated data and 5% real world data.

5 Experiments

We aim to address the following questions experimentally - (Q1) Does `OmniReset`outperform baselines and scale to tasks beyond current methods? (Q2) How do our key design decisions affect performance? (Q3) Can our experts be used to generate diverse data for sim-to-real transfer?

5.1 Task Descriptions

We consider the three tasks described below. We will refer to the distributions we define below as the **Entire Task Distributions**, as they define the actual set of initial conditions we intend to solve the task from. These three tasks are depicted in Figures 1 and 2. For each task we consider both `Easy` and `Hard` settings, but note our renderings in Figures 1 and 2 are all from `Hard` settings.

We emphasize that the policies depicted in Figures 1 and 2 were trained on the `Hard` settings.

1) Drawer: The `Drawer` task is based on the task by the same name from (Heo et al., 2023). For `Drawer Hard`, the drawer is spawned over the of x-y coordinates $(x - y) \in [-0.2, 0.2] \times [-.15, .15]$, the orientation of the drawer is drawn completely at random. `Drawer Easy` keeps the drawer at a fixed orientation with $(x, y) \in [-0.1, 0.12] \times [-.10, .12]$.

2) Screw: The `Screw` task is based on the `square_table` task from (Heo et al., 2023). To solve the task the agent must pick up the a table leg, insert it into the table, then screw it in. Both settings the orientation of the leg is randomized completely. `Screw Hard` uses $(x, y) \in [-0.2, 0.2] \times [-.15, .15]$ and `Screw Easy` uses $(x, y) \in [0.08, 0.13] \times [-0.025, 0.025]$.

3) Peg: We design a custom peg insertion task where the robot must grasp a rectangular peg and insert it into a receptacle. This involves precise grasping and insertion motions. For both settings we randomize the orientation of the leg uniformly. `Peg Hard` uses $(x, y) \in [-0.2, 0.2] \times [-.15, .15]$ and `Peg Easy` uses $(x, y) \in [0.08, 0.13] \times [-0.025, 0.025]$.

5.2 Baseline Algorithms

We compare to the following baselines, which bootstrap learning with expert demonstrations, providing them more prior information about how to solve the task than `OmniReset`.

1) BC-PPO: We add a Behavior-Cloning (BC) loss to the PPO objective to construct a baseline emblematic of numerous works combining BC and RL objectives (Hester et al., 2018; Rajeswaran et al., 2017). When training this algorithm, the environment is always reset from the standard reset distributions ρ^S described above and uses our standard reward structure (Eq 1)

2) DeepMimic: We use DeepMimic-style reward augmentation (Peng et al., 2018) on top of our reward structure from Equation 1. During resets, a random demonstration is chosen, and the agent is reset from a random point along the demonstration and is incentivizes to stay close to the demonstration by the auxiliary reward.

3) Demo Curriculum: This baseline defines an auto curriculum over resets from the standard reset distribution ρ^S and the demos provided by the user. This demo is constructed in the spirit of the one used in (Bauza et al., 2025), but we instead use PPO as the base RL algorithm to ensure a fair comparison. This approach biases samples towards resets from which the policy achieves some success, as these resets provide a strong learning signal. This approach uses our standard rewards.

5.3 Policy Training and Evaluation in Simulation

We scale `OmniReset` to the `Hard` versions of each task, providing full learning curves in Figure 9 in the Appendix. Here, in Fig 5, we primarily compare to baselines on the `Easy` versions of the tasks, since the baselines struggled to make any meaningful progress on the harder variants of the tasks. We report success rates for each method when initialized over the entire task distribution, as well as when explicitly spawned from states near the goal. Looking at the `Easy` versions of the tasks, we see that the baselines can make some progress towards solving the task from the near-goal states, but struggle to solve the entire task from the beginning. This is even more pronounced on the `Screw Hard` variant, where the baselines achieve near-zero success rates. In contrast, `OmniReset` reliably achieves a final success rate of >97% success on each of the tasks.

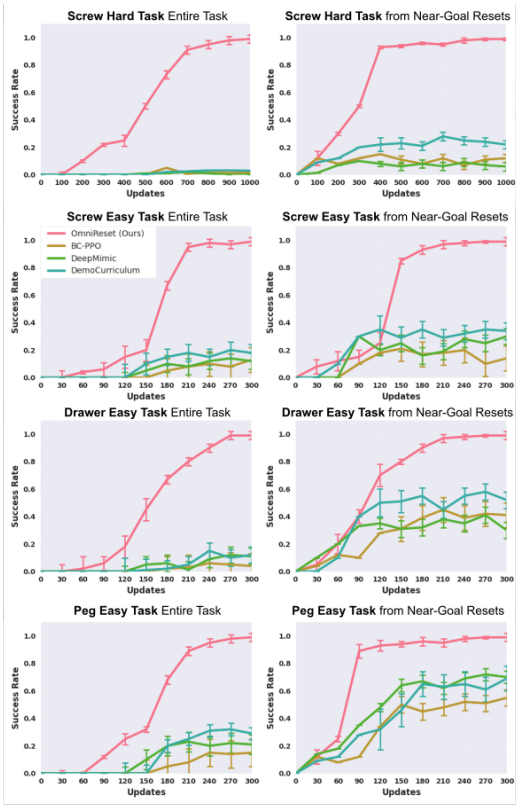


Figure 5: **Success rates during RL training.** We plot success rates over learning process for tasks described in Sec.5. We see that `OmniReset` yields significant improvements in success.

To give a more thorough picture of the performance of `OmniReset`, we analyze the robustness of learned policies in simulation via a series of scatter plots (Fig 6). These plots show success rates over various initial conditions for both `OmniReset` and Demo Curriculum (the most successful baseline) on the `Screw` tasks. For each plot we show the success rate over 1000 sampled initial conditions from the full task distribution. This plot demonstrates how the baseline struggles to achieve achieve consistent success across the distribution of initial conditions it was trained on, while `OmniReset` achieves high degrees of coverage. Finally, we conduct a robustness analysis on the learned policies (Fig 7). We sample an initial condition from one of the demonstrations, perturb the initial condition with forces of different magnitudes and report policy success from these perturbed initial conditions. We find that baseline performance quickly degrades under small perturbations, while the performance of `OmniReset` is barely affected, even under large perturbations.

We further ablate the key design decisions behind `OmniReset`. While we defer the detailed description to the Appendix, we find having sufficient coverage over two factors matters significantly for performance - 1) the number of parallel environments (and correspondingly RL batch size) and 2) the range of reset randomization used by `OmniReset`.

5.4 Real-World Transfer

We deploy our policies on a 6 DoF UR5e arm with a 2F-85 Robotiq Gripper mounted. We mount one Intel Realsense D435 and one Intel Realsense D455 camera rigidly on the table. We also mount one Intel Realsense D415 camera on the wrist of the arm. The real robot setup is shown in 8. The UR5e low-level impedance control runs at 500Hz and 2F-85 binary gripper control runs at 200 Hz.

We evaluate our method on `Screw` task from a narrow range of initial leg poses next to the tabletop.

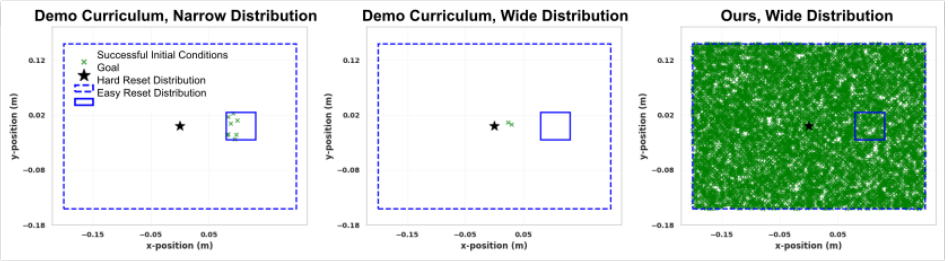


Figure 6: **Success states of RL policy.** For the screwing task, we plot the xy configurations from which RL polices trained with Demo Curriculum and `OmniReset`succeed when trained on the full reset distribution and on a narrow reset distribution. We find that `OmniReset`succeeds from a much broader range of initial conditions.

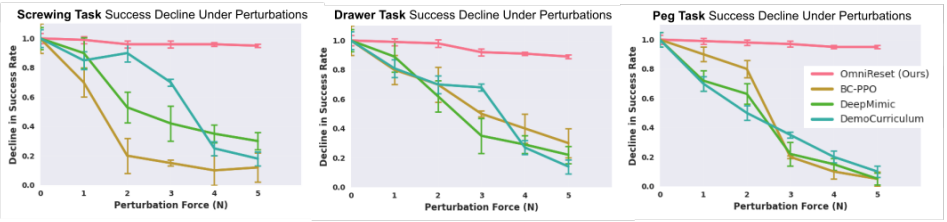


Figure 7: **Success rate over perturbations.** We plot the decline in success rate (measured by ratio of success rate between no perturbations and current level of perturbations). We find that `OmniReset`is robust to perturbations while performance of baselines drops significantly.

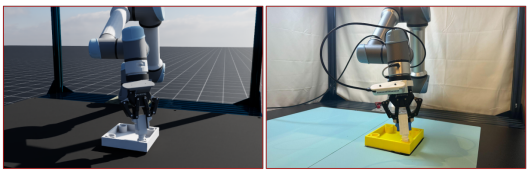


Figure 8: **Sim-to-Real Setup.** Real-world setup for evaluating transfer on the leg screwing task.

Evaluating our distilled RGB policy on 10,000 simulation trajectories zero-shot in real has a success rate of zero. Qualitatively, the robot can grasp leg but cannot align it with the tabletop hole successfully. We collect 100 real demonstrations of this task and finetune our distilled RGB policy on this via sim-real cotraining. The co-trained policy achieves 30% success in all areas of reaching, grasping, inserted, and twisting. Interestingly, if we train an RGB policy on just the 100 real demonstrations, the policy gets a 0% success rate, not even succeeding at grasping. Full results are shown in Table 1.

Table 1: Columns compare success rates of (i) a distilled RGB policy trained in simulation, (ii) a policy trained only on 100 real expert demonstrations, and (iii) a distilled simulation policy fine-tuned with sim+real co-training data.

	Distilled Sim Policy	Real-Only Policy	Sim+Real Co-Training
Success Rate (Sim)	98%	0%	N/A
Success Rate (Real)	0%	0%	30%

6 Conclusion

In this work we presented `OmniReset`, a simple and scalable system for data generation in simulation for complex, dexterous tasks. The primary insight in `OmniReset`is showing that a diverse, minimally structured set of reset states paired with large-batch on-policy reinforcement learning in simulation can lead to the emergence of surprisingly complex dexterous behavior. We provide a general purpose recipe to instantiate data generators across a variety of manipulation tasks, and demonstrate both the efficacy of this paradigm in simulation and it's ability to bootstrap real world policy learning. The presented work is currently still single-task and object, making a multi-task extension in future work a promising direction of investigation. Furthermore, `OmniReset`opens up the possibility of more systematically studying scaling laws for policy learning.

7 Reproducibility Statement

We have made efforts to ensure reproducibility of our results by describing the steps of our data generation and training pipeline (Sec.3), our distillation and transfer pipeline (Sec.4), and our experimental results (Sec.5). Additional ablation studies are provided in the Appendix.

References

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 1, 2

Lars Ankile, Anthony Simeonov, Idan Shenfeld, and Pulkit Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5096–5103. IEEE, 2024. 3

Maria Bauza, Jose Enriaue Chen, Valentin Dalibard, Nimrod Gileadi, Roland Hafner, Murilo F Martins, Joss Moore, Rugile Pevceviciute, Antoine Laurens, Dushyant Rao, et al. Demostart: Demonstration-led auto-curriculum applied to sim-to-real with multi-fingered robots. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6756–6763. IEEE, 2025. 2, 3, 8

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016. 3

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018. 3

Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pp. 297–307. PMLR, 2021. URL https://proceedings.mlr.press/v164/chen22a.html. 6

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020. 3

Ashley D Edwards, Laura Downs, and James C Davidson. Forward-backward reinforcement learning. *arXiv preprint arXiv:1803.10227*, 2018. 3

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2019. 3

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017. 2, 3

Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5977–5984. IEEE, 2023. 2

Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *The International Journal of Robotics Research*, pp. 02783649241304789, 2023. 2, 7

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 3, 7

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. 1

Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 15–21. IEEE, 2019. 3

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the nineteenth international conference on machine learning*, pp. 267–274, 2002. 3

Liyiming Ke, Yunchu Zhang, Abhay Deshpande, Siddhartha Srinivasa, and Abhishek Gupta. Ccil: Continuity-based data augmentation for corrective imitation learning. *arXiv preprint arXiv:2310.12972*, 2023. 3

Gagan Khandate, Siqi Shang, Eric T Chang, Tristan Luca Saidi, Yang Liu, Seth Matthew Dennis, Johnson Adams, and Matei Ciocarlie. Sampling-based exploration for reinforcement learning of dexterous manipulation. *arXiv preprint arXiv:2303.03486*, 2023. 2

Cevahir Koprulu, Po-han Li, Tianyu Qiu, Ruihan Zhao, Tyler Westenbroek, David Fridovich-Keil, Sandeep Chinchali, and Ufuk Topcu. Dense dynamics-aware reward synthesis: Integrating prior experience with demonstrations. *arXiv preprint arXiv:2412.01114*, 2024. 3

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023. 3

Jarryd Martin, Suraj Narayanan Sasikumar, Tom Everitt, and Marcus Hutter. Count-based exploration in feature space for reinforcement learning. *arXiv preprint arXiv:1706.08090*, 2017. 3

Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034. 1, 5, 6

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6292–6299. IEEE, 2018. 2, 3

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 3

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *J. Mach. Learn. Res.*, 20:124:1–124:62, 2019. URL https://jmlr.org/papers/v20/18-339.html. 3

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017. 3

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017. 3

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018. 2, 3, 7

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017. 6

Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In *Conference on robot learning*, pp. 1634–1644. PMLR, 2022. 6

Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron C. Courville, and Alexandre Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28598–28617. PMLR, 2023. URL https://proceedings.mlr.press/v202/rajeswar23a.html. 3

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017. 7

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2018. URL https://arxiv.org/abs/1709.10087. 3

Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019. 3

Cinjon Resnick, Roberta Raileanu, Sanyam Kapoor, Alexander Peysakhovich, Kyunghyun Cho, and Joan Bruna. Backplay:" man muss immer umkehren". *arXiv preprint arXiv:1807.06919*, 2018. 3

Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *Found. Trends Mach. Learn.*, 11(1):1–96, 2018. doi: 10.1561/2200000070. URL https://doi.org/10.1561/2200000070. 3

Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018. 3

Cansu Sancaktar, Sebastian Blaes, and Georg Martius. Curious exploration via structured world models yields zero-shot object manipulation. *Advances in Neural Information Processing Systems*, 35:24170–24183, 2022. 3

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2

Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands, 2025. URL https://arxiv.org/abs/2412.01791. 7

Jayesh Singla, Ananye Agarwal, and Deepak Pathak. Sapg: split and aggregate policy gradients. *arXiv preprint arXiv:2407.20230*, 2024. 2

Bingjie Tang, Michael A Lin, Iretiayo Akinola, Ankur Handa, Gaurav S Sukhatme, Fabio Ramos, Dieter Fox, and Yashraj Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. *arXiv preprint arXiv:2305.17110*, 2023. 3

Bingjie Tang, Iretiayo Akinola, Jie Xu, Bowen Wen, Ankur Handa, Karl Van Wyk, Dieter Fox, Gaurav S Sukhatme, Fabio Ramos, and Yashraj Narang. Automate: Specialist and generalist assembly policies over diverse geometries. *arXiv preprint arXiv:2407.08028*, 1(2), 2024. 2, 3, 5

Stone Tao, Arth Shukla, Tse-kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demonstration efficiency in reinforcement learning. *arXiv preprint arXiv:2405.03379*, 2024. 3

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. 1

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017. 3

A Appendix

A.1 Ablations

We ablate the key design decisions that allow our approach to reliably scale to the `Hard` versions of the tasks we consider. We ablate 1) the number of parallel environments (and PPO batch size) and 2) the range of reset randomization used by `OmniReset`. Due to the high-compute cost needed to solve the full `Hard` tasks, we restrict this study to `Screw Hard`. We find that more environments during PPO training and broader grasp samples in the reset distribution dramatically improve sample efficiency and converged final success rate. Moreover, we plot success rates from each of the sub-reset distributions (D^R, D^NO, D^G and D^{NG}) as they provide an indication of how `OmniReset` works backwards from the end of the problem without requiring any explicit curricula, first achieving a high success rate on the near goal states D^{NG} then making progress on reset distributions which correspond to earlier portions of the task. Refer to Figures 9 and 10 in the appendix for details.

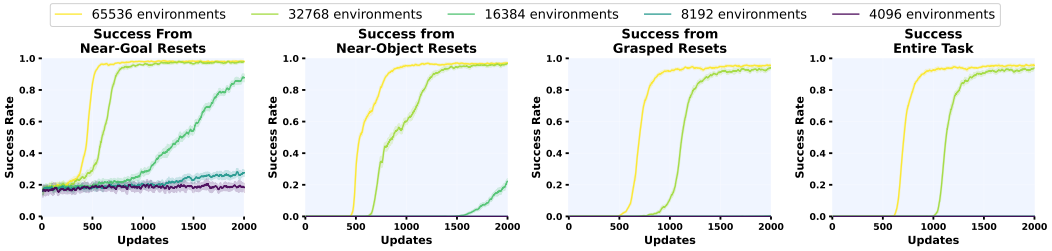


Figure 9: **Ablation on number of environments.** We plot the success rates over course of RL training using different number of environments. We find that the number of environments significantly impacts training performance.

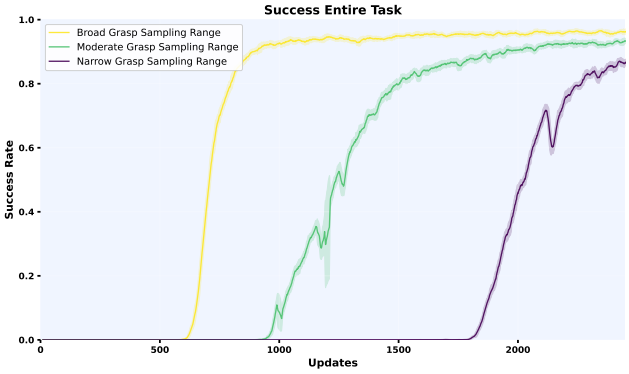


Figure 10: **Ablation on grasp sampling range.** For this ablation on the screwing task, we find that training RL on narrower grasp sampling ranges leads to worse sample efficiency and lower converged success rate.

A.2 The Use of Large Language Models (LLMs)

We used the help of LLMs for formatting some of the figures in LaTeX and spell check for the paper. We also used it as a code assist tool during the research process for writing some of the code for our experiments.