Assistax: A Hardware-Accelerated Reinforcement Learning Benchmark for Assistive Robotics

Anonymous authors
Paper under double-blind review

Abstract

The development of reinforcement learning (RL) algorithms has been largely driven by ambitious challenge tasks and benchmarks. Games have dominated RL benchmarks because they present relevant challenges, are inexpensive to run and easy to understand. While games such as Go and Atari have led to many breakthroughs, they often do not directly translate to real-world embodied applications. In recognising the need to diversify RL benchmarks and addressing complexities that arise in embodied interaction scenarios, we introduce Assistax: an open-source benchmark designed to address challenges arising in assistive robotics tasks. Assistax uses JAX's hardware acceleration for significant speed-ups for learning in physics-based simulations. In terms of open-loop wall-clock time Assistax runs up to $370\times$ faster, compared to CPU-based alternatives, when vectorising training runs. Assistax conceptualises the interaction between an assistive robot and an active human patient using multi-agent RL to train a population of diverse partner agents against which an embodied robotic agent's zero-shot coordination capabilities can be tested. Extensive evaluation and hyperparameter tuning for popular continuous control RL and MARL algorithms provide reliable baselines and establish Assistax as a practical benchmark for advancing RL research for assistive robotics.

1 Introduction

Assistive robotics (Chen et al., 2013; Savage, 2022) aims to develop autonomous systems that aid human users in performing various daily activities. For example, a healthcare application could be bed bathing where a robot is tasked with washing a human user who has a mobility impairment and cannot complete the task by themselves. The robot must account for both the user's behaviour and preferences like the duration and intensity of contact. Such a robot should also be capable of completing this it's tasks across a wide range of different human's, for example, if the robot is deployed in a care home the robot will be attending to multiple different users. As it is infeasible to individually train robots for each human they may encounter over their life-cycle, we wish to design a robot that is capable of interacting with any other agent it may encounter. This is challenging because the robot must act and coordinate with a user (another agent) in the context of limited or no prior experience and knowledge.

Benchmarks play a crucial role in advancing reinforcement learning (RL) by providing structured challenges and enabling comparative evaluation; in particular, a series of game benchmarks has been instrumental in driving innovation in the field (Bellemare et al., 2013; Mnih et al., 2013; Silver et al., 2016; Vinyals et al., 2019; Berner et al., 2019). We have also seen great success in robot simulation based learning for manipulation and locomotion tasks (Rajeswaran et al., 2018; Tan et al., 2018). Physics engines like MuJoCo (Todorov et al., 2012; Coumans, 2015; Makoviychuk et al., 2021) provide essential tools for exploring continuous control tasks in 3D environments, forming the backbone of numerous simulated tasks in robotics and reinforcement learning benchmarks. While continuos control benchmarks have established themselves in a single-agent setting, they have

Table 1: Assistax comparison with related benchmarks. Assistax supports algorithms for single-agent reinforcement learning (SARL), multi-agent reinforcement learning (MARL), and zero-shot coordination (ZSC) (a special case of ad-hoc teamwork (AHT)) in a 3D environment with continuous actions and hardware acceleration.

Benchmarks	**Algorithms**			**Environment**		
	SARL	MARL	ZSC	Continuous actions	3D	Hardware acceleration
DM Control (Tunyasuvunakool et al., 2020)	✓	✗	✗	✓	✗	✗
Bi-DexHands (Chen et al., 2022)	✓	✓	✗	✓	✓	✓
RLBench (James et al., 2020)	✓	✗	✗	✓	✓	✗
Robotsuite (Zhu et al., 2020b)	✓	✗	✗	✓	✓	✗
Gymnax (Lange, 2022)	✓	✗	✗	✓	✗	✓
SMAC (Samvelyan et al., 2019)	✗	✓	✗	✗	✗	✗
JaxMARL (Rutherford et al., 2024a)	✗	✓	✗	✓	✓	✓
Hanabi (Bard et al., 2019)	✗	✓	✓	✗	✗	✗
Assistive Gym (Erickson et al., 2019b)	✓	✓	✗	✓	✓	✗
Assistax (ours)	✓	✓	✓	✓	✓	✓

yet to make a similar impact in multi-agent settings and in particular in modelling human-robot interaction. For the assistive tasks we are interested in we require interaction policies that are robust when interacting with previously unseen other agents, a problem widely studies as ad-hoc teamwork (AHT) (Mirsky et al., 2022).

A key consideration when developing a benchmark for assistive robotics is the ability to rapidly prototype and evaluate various algorithms. This is particularly important for RL algorithms as they require a large number of interactions with the environment and multiple runs for rigorous evaluation and hyperparameter tuning (Colas et al., 2019; Agarwal et al., 2021). Because of this, various RL benchmarks have focused on designing environments that can utilise hardware acceleration allowing for agent environment GPU/TPU collocation and greater parallelisation, resulting in faster training and evaluation by several orders of magnitude when compared to using CPU-based alternatives (Rutherford et al., 2024b; Lange, 2022; Chen et al., 2022).

In light of these advancements and to foster research in assistive robotics using RL this paper introduces *Assistax* - a hardware-accelerated RL benchmark for assistive robotics. This benchmark offers unique combinations of features from the environment and algorithm points of view compared to similar benchmarks (see Table 1). Assistax utilizes the JAX (Bradbury et al., 2018) Python library and MuJoCo's MJX physics engine taking full advantage of hardware acceleration for research-friendly RL training pipelines. This benchmark focuses on real-world assistive robotics scenarios (Erickson et al., 2019a) and provides baselines for training and evaluating algorithms for agents trained using popular single-agent RL (SARL) and multi-agent RL (MARL). We also provide baselines for zero-shot coordination (ZSC) a special case of AHT by testing SARL algorithms when trained and evaluated against disjoint pre-trained partner-agent populations. We also provide the parameters for our partner-agent population for future use. To the best of our knowledge, Assistax is the first hardware-accelerated benchmark targeting assistive robotics and AHT. This benchmark contributes to the development of RL algorithms in assistive robotics by providing an effective hardware-accelerated environment with accompanying tasks and baselines.

2 Background and Related Work

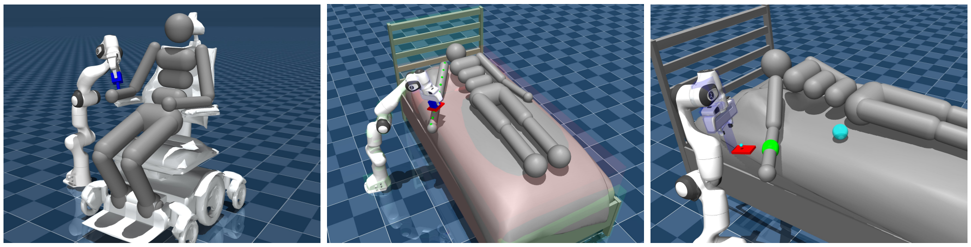
Hardware-Accelerated RL Benchmarks. Many environments leverage hardware acceleration (Bettini et al., 2022; Richmond et al., 2023; Mittal et al., 2023), but often restrict themselves to 2D domains or discrete state and action spaces. Assistax uses the JAX Python library (Bradbury et al.,

2018) for easy parallelization via `vmap` (vectorization) and `pmap` (multi-device distribution). Additionally, MuJoCo's MJX enables collocating agents and environments on GPUs/TPUs, removing CPU-GPU memory transfers and enhancing performance through JAX's JIT compilation. Existing JAX-based RL environments include Gymnax (Lange, 2022), Pgx (Koyamada et al., 2023), and MuJoCo-based benchmarks (Freeman et al., 2021; Zakka et al., 2025). JaxMARL (Rutherford et al., 2024b) specifically targets MARL, achieving significant speedups.

Multi-Agent Reinforcement Learning. MARL considers multiple RL agents interacting and learning in an environment simultaneously (Albrecht et al., 2024). The Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek & Amato, 2016) is the canonical representation of the decision-making problem in MARL. It is a tuple $\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, T, p_0, \{\Omega^i\}_{i\in\mathcal{N}}, O, R, \gamma\rangle$ where: $\mathcal{N}$ is the set of agents; $\mathcal{S}$ is the state space; $\mathcal{A}^i$ is the action space for agent i; $T(s_{t+1} \mid s_t, a_t)$ is the state transition probability function; $p_0(s_0)$ is the initial state distribution; Ω^i is the observation space for agent i; $O(o_t \mid s_t, a_{t-1})$ is the observation probability function; $R(s_t, a_t)$ is the team reward function; and $\gamma \in [0, 1]$ is the discount factor. In Dec-POMDP, at every time-step t all agents $i \in \mathcal{N}$ take an action a^i_t, each of these actions forms the joint action $a_t = (a^1_t, a^2_t, \dots, a^{|\mathcal{N}|}_t)$. After the action a_t is taken, the state transitions to the next state s_{t+1} given by the transition probability function $s_{t+1} \sim T(\cdot \mid s_t, a_t)$. Each agent receives an observation o^i_{t+1} given by the observation probability function $o_t \sim O(\cdot \mid s_t, a_{t-1})$ and receives the reward r_t given by the team reward function $R(s_t, a_t)$. Joint actions are sampled from the joint policy $\pi = (\pi^1, ..., \pi^{|\mathcal{N}|})$ i.e., $a_t \sim \pi(\cdot|h_t)$, where $h_t = (h^1_t, ..., h^{|\mathcal{N}|}_t)$ is the joint history. The history of each agent $i \in \mathcal{N}$ consists of all observations and actions up until the current timestep $h^i = (o^i_0, a^i_0, o^i_1, a^i_1, \dots o^i_t)$. In Dec-POMDP, the objective is to maximize the discounted cumulative rewards over a finite/infinite horizon.

Ad-hoc Teamwork. AHT is the problem of controlling a single agent or a subset of agents within a team, to maximize the team's returns across a broad distribution of previously unknown teammate types (Mirsky et al., 2022). A teammate type refers to the distinct set of behavioural policies, capabilities, and preferences that define how a teammate operates and collaborates within a shared task (Stone et al., 2010). Zero-Shot Coordination (ZSC), a special case of AHT, requires agents to coordinate with unseen partners without adaptation, typically generalizing from training with a limited set of partner policies (Hu et al., 2021). Existing benchmarks (e.g., Hanabi (Bard et al., 2019)) often lack the complexity needed for assistive robotics, which involves continuous actions and multi-agent interactions in 3D environments. Assistax specifically addresses these complexities.

3 Assistax



(a) *Scratch* (b) *Bed Bath* (c) *Arm Assist*

Figure 1: Suite of hardware-accelerated simulated environments and tasks provided by Assistax.

Assistax is a Python library that provides hardware-accelerated environments in the domain of assistive robotics together with accompanying baseline algorithm implementation.

3.1 Environments

Tasks The visualization of the Assistax task suite is given in Figure 1. Inspired by Assistive Gym (Erickson et al., 2019b), Assistax implements *Scratch*, *Bed Bath*, and *Arm Assist* tasks, representing real-world assistive robotic scenarios between a human and a robot as well. Although the task goals are the same as in Assistive Gym, our implementations differ in terms of the observation space and the reward function (see Appendix A). Currently, MJX scales poorly in scenarios involving many collisions, resulting in significant slowdowns when simulating deformable bodies such as cloth, liquids, or detailed mesh interactions. However, we anticipate that future advances in MJX will enable the simulation of more complex tasks, such as robotic dressing. Each task is modelled as Dec-POMDP with two agents. Assistax strives to reuse all relevant components between tasks, to clearly distinguish which individual complexities each task offers. The more detail description of each task is as follows:

- *Scratch* (Figure 1(a)): A scratching target is randomly sampled on the surface of the human's right arm. The robot must move its end-effector to this position and apply a specified force. The human can move its arm to make the target more accessible to the robot.
- *Bed Bath* (Figure 1(b)): We provide target bath points distributed along the surface of the human's arm. The robot must reach each point and apply a certain force to activate the next point. The aim is to reach ('wipe') all points before the end of an episode.
- *Arm Assist* (Figure 1(c)): The robot must help the human lift its right arm back into a comfortable position on the bed. In this task the human is too weak to complete the task on their own and thus requires the robot. The robot has to learn to align its end-effector with a target section of the arm (shown in green on Figure 1(c)), and then move the human arm until the green and blue targets overlap.

Agents The Assistax environment supports robot and human agents. The robot agent takes the form of a Franka Emika Panda robot arm for all environments with its base model taken from MuJuCo menagerie (Zakka et al., 2022). For each task, we implement custom end-effectors, which do not have controllable joints. The robot agent is torque-controlled with 7 joints $\mathcal{A}^{robot} := [-1, 1]^7$. The human model is taken from the Brax humanoid tasks (Freeman et al., 2021) and is also torque-controlled. Note that when referring to the "human" we mean the agent controlling the human model. For the human we restrict the action space to the shoulder and elbow joints of the right arm $\mathcal{A}^{human} := [-1, 1]^3$, which are most relevant for the tasks. Following Assistive Gym, Assistax simulates tremors, joint weakness and limited range of motion in an attempt to replicate real-world challenges for assistive healthcare tasks.

3.2 Algorithms and Baselines

Overview Assistax takes inspiration from JaxMARL (Rutherford et al., 2024b) for straightforward single-file implementations of continuous RL algorithms. We conduct extensive hyperparameter tuning for each of these implementations to ensure a reliable benchmark (see Appendix B for details).

SARL and MARL Algorithms Assistax includes implementations of PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018) as baseline SARL algorithms for continuous control tasks. For both SARL algorithms, we provide MARL variants, including Independent (IPPO, ISAC) and Multi-Agent (MAPPO, MASAC) approaches. The RL agent training pipeline is optimized for JAX, leveraging its `vmap` functionality to vectorize across multiple environments, as well as parallelize training across multiple seeds and hyperparameter settings.

ZSC Baseline Assistax is also designed for benchmarking of ZSC. We use our MARL algorithms to train a population of potential team-mate human agents Π with varying disability parameters a summary of this can be seen in Table 6. We train our SARL algorithms on a random half of this partner agent population Π^{train} and evaluate against the unseen half denoted as Π^{test}. More formally we train for the following objective (Rahman et al., 2024):

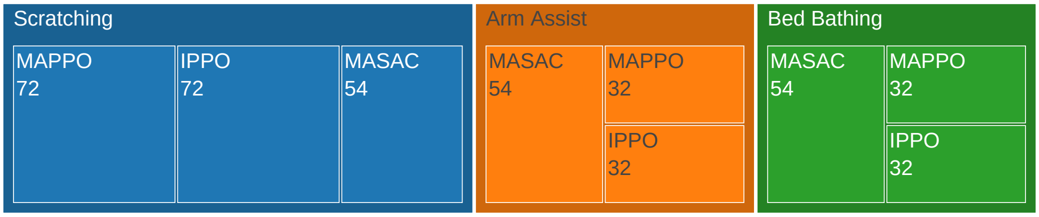


Figure 2: The treemap visualization of the partner agent population for different tasks trained using different MARL algorithms.

$$\pi^{i^*}(\Pi^{\text{train}}) = \arg\max_{\pi^i} \mathbb{E}_{\pi^{-i}\sim\mathbb{U}(\Pi^{\text{train}}), a_t^i\sim\pi_i, a_t^{-i}\sim\pi^{-i}, T, O}\left[\sum_{t=0}^{\infty}\gamma^t R(a_t, s_t, a_{t+1})\right].$$

We then evaluate the measure $M_{\Pi^{\text{test}}}$ which measures the robustness of the π^{i^*} when paired with unseen agents uniformly sampled from Π^{test}. Which we define as:

$$M_{\Pi^{\text{test}}} = \mathbb{E}_{\pi^{-i}\sim\mathbb{U}(\Pi^{\text{test}}), a_t^i\sim\pi^{i^*}(\Pi^{\text{train}}), a_t^{-i}\sim\pi^{-i}, T, O}\left[\sum_{t=0}^{\infty}\gamma^t R(a_t, s_t, a_{t+1})\right].$$

Assistax provides the parameters of pre-trained partner policies, allowing the benchmark users to train SARL algorithms in a multi-agent setting against an active pre-trained teammate where pre-trained refers to an already trained teammate policy using MARL.

3.3 Optimized computation

Assistax prioritises simulation efficiency over high fidelity, a trade-off made to speed up RL training pipelines. This trade-off is currently necessary as MuJoCo's MJX has some limitations, most notable for our use-case is the poor scaling with the number of collisions making mesh collisions unfeasible. As RL typically requires extensive interaction with the environment, it makes sense to focus on simulation speed, which enables researchers to train policies faster, do rigorous hyperparameter tuning, and run more experiments. Although this trade-off sacrifices some physical accuracy, it improves the library's utility for RL research by making these more complex environments much more attractive from a computation-cost perspective. Key trade-offs include[1]:

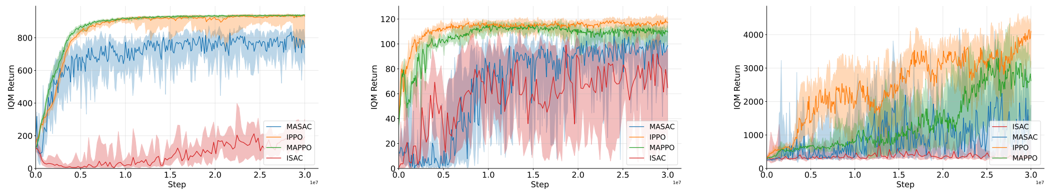
1. *Primitive geometries*: We simplify objects by fitting them with primitive geometries (e.g., capsules for the Franka arm, boxes for wheelchairs and beds). These shapes reduce computational overhead while maintaining task relevance.
2. *Collision Optimization*: We selectively disable collisions between geometries that are unlikely to interact during an episode, further improving simulation efficiency.

See Figure 8 in the Appendix A.1 for more information on collisions and primitive geometries.

4 Experiments

In this section we benchmark our algorithms. We evaluate how well our MARL algorithms perform when they are co-trained. We also evaluate the diversity of co trained policies to see whether MARL

[1] While this is set by default these can be adjusted to increase the physical fidelity of the tasks by leveraging the MuJoCo XML API, providing flexibility for higher-fidelity requirements.



(a) *Scratching* (b) *Bed Bathing* (c) *Arm Assist*

Figure 3: Learning curves for training MARL baselines. All curves show inter-quartile means ± 95% stratified-bootstrap CIs over 16 seeds.

training leads to different conventions or stratagies between the two agents, by analysing the cross-play returns for a population of trained agents for each task. Lastly we also benchmark SARL algorithms for Zero-Shot Co-ordination in line with Section 3.2.

4.1 MARL Experiments

We provide evaluation for four baseline MARL algorithms across all three tasks, where we constrain the robot and the human with shared rewards. We evaluate MASAC, ISAC, MAPPO and IPPO, each with no parameter sharing and simple feed-forward architectures. While we do not evaluate other architectures we do provide implementations for each algorithms with a RNN architecture as well as parameter sharing options.

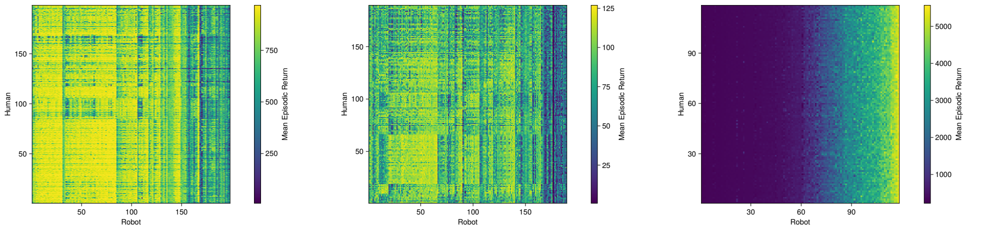
We conduct extensive hyperparameter tuning, testing at least 168 different hyperparameter settings for each algorithm-environment pair. We perform a random sweep for continuous hyperparameters and select a few reasonable settings for discrete hyperparameters. The final hyperparameters used, along with the results of the sweeps, can be found in the appendix. Below, we present the results for algorithms using the optimized hyperparameters from the sweep. We show learning curves and final returns for baseline algorithms Figure 3. PPO variants typically outperform SAC algorithms which present much higher variance in their performance. This discrepency is likely a symptom of challenges that arise for off-policy algorithms due to the non-stationarity introduced by other learning agents (Lowe et al., 2017).

Table 2: MARL baseline evaluation using final interquartile mean (IQM) returns with 95% confidence interval. Values in bold denote best-performing algorithms for each task. Upper Bound shows the theoretical upper bound of returns for each task i.e. by obtaining the maximum possible reward at each time-step.

Algorithm \ Task	*Scratching*	*Bed Bath*	*Arm Assist*
MAPPO	**945.80**[804.52, 949.37]	109.52[106.84, 111.73]	2621.95[1310.16, 3204.55]
IPPO	**939.51**[933.97, 957.97]	**117.16**[112.44, 120.24]	**4101.31**[3027.48, 4347.11]
MASAC	708.87[502.33, 831.42]	98.71[79.64, 116.76]	910.21[370.89, 1890.83]
ISAC	84.12[29.61, 305.65]	64.68[22.06, 96.35]	325.68[297.88, 616.79]
Upper Bound	1,135	1,052	11,346

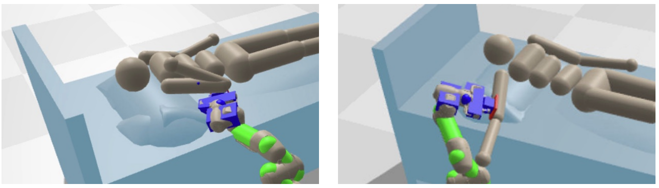
4.2 ZSC Experiments

Assistax also provides a baseline for the ZSC capabilities of single-agent versions of PPO and SAC. We take inspiration from RL domain randomization benchmarks like ProcGen (Cobbe et al., 2019) and the domain randomization techniques used for sim-to-real transfer in robotics applications (Tobin et al., 2017). We randomize the 'human' policy that the single-agent robot sees during training.



(a) *Scratching* (b) *Bed Bathing* (c) *Arm Assist*

Figure 4: Cross-play matrices obtained by computing the returns (averaged over 16 seeds) when robot agents trained in one team are paired with human agents trained in another team. The order of agents in each graph has been permuted using a hierarchical clustering algorithm to show the strategy clusters more clearly.



(a) *Strategy 1* (b) *Strategy 2*

Figure 5: Two emerging strategies for bedbathing, which are mutually incompatible. Strategy 5(a) and 5(b) have a mean episode return of 122.6 and 118.5 respectively this however drops to below 60 when the different strategies are matched with eachother.

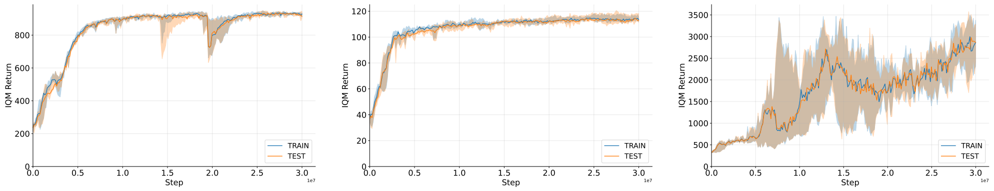
At each reset of the environment, a random partner policy is sampled from the train set of partner policies.

Using Assistax's hardware acceleration we are able to pre-train 434 active 'human' policies using our MARL baselines and by varying disability parameters (See Table 2 for an overview of the agent population and see Table 6 in the appendix for more details). For each algorithm and task, we train partner agents with 9 different disability settings which limit the joint strength and range of motion of the elbow.

Through co-trained MARL we obtain cross-play matrices which show that for the scratching (Figure 4(a)) and bed-bathing tasks (Figure 4(b)) distinct strategies emerge, such that there are certain groups of agents whose policies are incompatible with each other. Figure 5 showcases one such example from the bed-bathinig task. Nevertheless, we note that few distinct clusters emerge, suggesting that the tasks do not require a high degree of coordination. For example, the scratching task requires the robot to navigate to a scratching target, while the human makes this target more accessible to the robot's end-effector. The simplicity of this task means that there are few distinct optimal strategies for solving the task. The arm-assist task in particular does not require complex coordination between the human and robot, as Figure 4(c) shows that the human contributes little to task success. As a consequence, the returns depend mainly on the performance of the robot.

4.3 Runtime Experiments

A key benefit of implementing the environment in JAX is the significant speed-up in training pipelines, as it allows reinforcement learning to leverage hardware acceleration. In terms of wall-clock time a typical IPPO training run of 30 million environment time-steps takes roughly 20 minutes when using 512 vectorized environments; in comparison the equivalent training run for Assistive gym (with RLlib multiprocessing Liang et al. (2018)) take 8.3 hours, resulting in an approximate speed up of



(a) *Scratching*

(b) *Bed Bathing*

(c) *Arm Assist*

Figure 6: ZSC Performance of PPO on all three tasks. Showing IQM returns and stratified 95% bootstrapped CI across 16 seeds.

this translates into an approximate speed up of 25× over the original assistive-gym. Figure 7 shows how the scaling of the steps per second scale with the number of vectorized environment for an IPPO training run. Table 3 shows the open-loop speeds compared with assistive gym, in which we run the environments with no RL training.

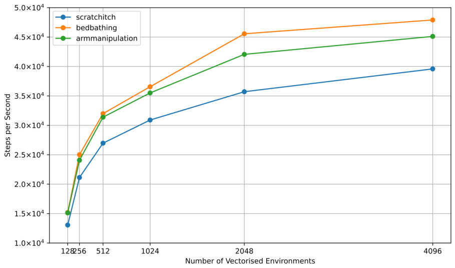


Figure 7: Steps per second and number of vectorized environments for IPPO training pipeline using a single A100 (40GB).

Table 3: Task speed when taking random actions for 10 million timesteps with 512 vectorized environments using an Nvidia A100 (40GB) GPU. Relative speedup is against Assistive Gym with a single environment.

Task	SPS	Relative Speedup
Scratch	26,953	116.6×
Bed bath	34,218	370.8×
Arm assist	34,097	238.19×

5 Conclusion

This paper introduced a hardware-accelerated RL benchmark for assistive robotics. The presented task suite and experiments demonstrate that continuous, physics-based 3D environments can, from a computational-cost perspective, compete with simpler, game-like settings commonly used in SARL and MARL, thereby enabling faster research iteration and more thorough evaluations. Specifically for AHT, we focused on the case of ZSC, providing a baseline for future advancements in assistive robotics. The benefits of hardware acceleration enable efficient training of large numbers of embodied agents; to the best of our knowledge, Assistax is the only benchmark facilitating investigation of ZSC for distinct embodied agents. This positions Assistax as the benchmark of choice not only for RL research in assistive robotics but also more broadly within RL and AHT.

Limitations and Future Work There is a trade-off between the algorithm's runtime and the environment's fidelity which may come at the cost of replicating real-world scenarios. Off-policy algorithms like SAC are require careful consideration of the sampling/replay ratio when parallelising across many simulations(Rutherford et al., 2024b). Further, our benchmark tasks are not long-horizon and are often extensions of reach tasks with additional complexities introduced by the multi-agent interaction. Designing reward functions for more complex longer horizon tasks remains very challenging.

A Benchmark Details

Code is available at: https://github.com/anonym-nips/assistax

We provide details about the observation space and the reward function for different Assistax tasks.

Observation Space.

A summary of the observations for each task is shown in Table 4. In our benchmark, we consider 3 types of observations: proprioception, tactile, and ground-truth information from the simulator. Proprioception is information relating to robot configuration. It is computed from the robot's internal sensors. Assistax considers tactile observation of the net contact forces between the end-effector and the human arm, expressed as a force vector in the MuJoCo contact frame. Observations about the other body forces are not included. Ground truth refers to the privileged information available in simulation but requires estimation in the real world (e.g. end-effector to object distance, human joint angles). Subscripts R and H denote robot and human respectively. The end-effector is chosen to be an imaginary frame at the end of the robotic arm chosen. Note that *Arm Assist* has a larger observation space to take into account the increased task complexity. For this task, we provide the rotation matrix between the end-effector and the target on the human arm (green in Figure 1(c)) to influence the robot to lift the arm in a particular way. We also provide an additional distance target for the second phase of bringing the arm back to the waist target position (blue in Figure 1(c)).

Table 4: Observations space overview. Assistax uses three types of observations: proprioception (prop.), tactile, and ground-truth (gt.) information from the simulator.

Type	Symbol	Description	Dimension		Task	
				Scratch	*Bed Bath*	*Arm Assist*
prop.	θ_R	robot joint angles	7	✓	✓	✓
	$\dot{\theta}_R$	robot joint velocities	7	✓	✓	✓
	x_{ee}	end-effector position	3	✓	✓	✓
	q_{ee}	end-effector quaternion	4	✓	✓	✓
tactile	f_{ee}	end-effector force	3	✓	✓	✓
gt.	θ_H	human joint angles	9	✓	✓	✓
	$\dot{\theta}_H$	human joint velocities	9	✓	✓	✓
	$x_{H_{lower}}$	human lower arm position	3	✓	✓	✓
	$x_{H_{upper}}$	human upper arm position	3	✓	✓	✓
	x_{ee_t}	end-effector to target distance	3	✓	✓	✓
	d_{ee_t}	end-effector to target euclidean distance	1	✓	✓	✓
	$\mathbf{R}_{ee\_t}$	end-effector to target angular distance	9	✗	✗	✓
	$x_{H\_t'}$	human arm to waist target distance	3	✗	✗	✓
	$d_{H\_t'}$	human arm to waist target euclidean distance	1	✗	✗	✓

Reward function. The reward for each task at every timestep is given as a linear combination of different components involving numerical and indicator functions (e.g., the end-effector being close to the target). A constant numeric value scales each of the components. The summary of the reward components used is given in Table 5.

Table 5: Reward component overview. Each component is evaluated using the equation and scaled when computing the reward. In equations σ is a scaling factor we set to 0.1, v_{ee} is the end-effector velocity, $[\cdot]$ is the indicator function and f^* and v^* are the target forces and velocities, respectively.

Component	**Symbol**	**Equation**	**Scale**	**Task**		
				Scratch	*Bed Bath*	*Arm Assist*
Reach target	r_t	$\exp\left(-\frac{d_{ee}^2}{\sigma}\right)$	1	✓	✓	✓
Scratch	r_s	$[d_{ee\_t} < 0.1] \cdot \left(\frac{v_{ee}}{v^*} \exp\left(-\frac{v_{ee}}{v^*}\right)\right) \cdot \left(\frac{f_{ee}}{f^*} \exp\left(-\frac{f_{ee}}{f^*}\right)\right)$	1	✓	✗	✗
Wipe	r_w	$[d_{ee\_t} < 0.1] \cdot [f_{ee} > 0]$	1	✗	✓	✗
Reach waist	$r_{t'}$	$1 - \tanh\left(\frac{d_{H\_t'}}{\sigma}\right)$	10	✗	✗	✓
Rotation	r_R	$\text{norm}(\mathbf{R}_{ee\_t})$	0.1	✗	✗	✓

Table 6: Composition of Partner Agent Population

Algorithm / Tasks		Scratching		Bed Bathing		Arm Assistance		Total	
		n = 198	%	n = 118	%	n = 118	%	n = 434	%
IPPO									
	Disability Setting 1	8	4.0	8	6.8	8	6.8	24	5.5
	Disability Setting 2	8	4.0	8	6.8	8	6.8	24	5.5
	Disability Setting 3	8	4.0	8	6.8	8	6.8	24	5.5
	Disability Setting 4	8	4.0	8	6.8	8	6.8	24	5.5
	Disability Setting 5	8	4.0	0	0	0	0	8	4.0
	Disability Setting 6	8	4.0	0	0	0	0	8	4.0
	Disability Setting 7	8	4.0	0	0	0	0	8	4.0
	Disability Setting 8	8	4.0	0	0	0	0	8	4.0
	Disability Setting 9	8	4.0	0	0	0	0	8	4.0
	IPPO Total	**72**	**36.4**	**32**	**27.1**	**32**	**27.1**	**136**	**31.3**
MAPPO									
	Disability Setting 1	8	4.0	8	6.8	8	6.8	24	5.5
	Disability Setting 2	8	4.0	8	6.8	8	6.8	24	5.5
	Disability Setting 3	8	4.0	8	6.8	8	6.8	24	5.5
	Disability Setting 4	8	4.0	8	6.8	8	6.8	24	5.5
	Disability Setting 5	8	4.0	0	0	0	0	8	4.0
	Disability Setting 6	8	4.0	0	0	0	0	8	4.0
	Disability Setting 7	8	4.0	0	0	0	0	8	4.0
	Disability Setting 8	8	4.0	0	0	0	0	8	4.0
	Disability Setting 9	8	4.0	0	0	0	0	8	4.0
	MAPPO Total	**72**	**36.4**	**32**	**27.1**	**32**	**27.1**	**136**	**31.3**
MASAC									
	Disability Setting 1	6	5.1	6	5.1	6	5.1	18	4.1
	Disability Setting 2	6	5.1	6	5.1	6	5.1	18	4.1
	Disability Setting 3	6	5.1	6	5.1	6	5.1	18	4.1
	Disability Setting 4	6	5.1	6	5.1	6	5.1	18	4.1
	Disability Setting 5	6	5.1	6	5.1	6	5.1	18	4.1
	Disability Setting 6	6	5.1	6	5.1	6	5.1	18	4.1
	Disability Setting 7	6	5.1	6	5.1	6	5.1	18	4.1
	Disability Setting 8	6	5.1	6	5.1	6	5.1	18	4.1
	Disability Setting 9	6	5.1	6	5.1	6	5.1	18	4.1
	MASAC Total	**54**	**45.8**	**54**	**45.8**	**54**	**27.3**	**162**	**37.3**

Composition of Partner Population for ZSC The Table 6 shows the overview of the partner agent population provided by Assitax. The agents are trained using different MARL algorithms with varying disability settings.

A.1 Simulation Fidelity

See Figure 8 for an example of how we set collisions in Assistax and how we use primitive geometries instead of mesh collisions.

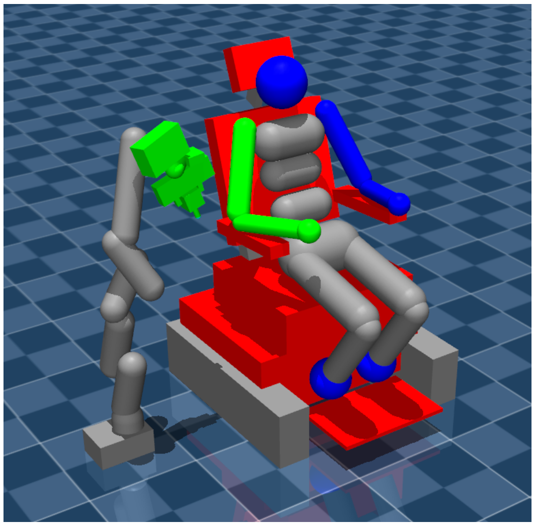


Figure 8: Shows the scratch-itch task with primitive geometries. Green geometries collide with both green and red geometries while blue geometries collide with blue and red geometries. Gray geometries have collision disabled.

B Hyperparameters

We run hyperparameter sweeps for all our algorithms task pairs providing confidence in our baseline results. For each algorithm and each task we test at least 168 different configuration, each configuration is tested on 3 seeds. Below we provide plots and the chosen hyperparameters setting for the baselines. We use the same hyperparameter settings for each task as there was no meaningful difference between tasks.

B.1 Hyperparameter Sweep Results

In Figures 9, 10 11, 12, and 13 we show the results chosen hyper-parameter sweeps.

For MASAC we run some additional ablations to show some intuitions on why the Q-learning rate is more indicative of performance than the policy learning rate although these results lack statistical significance, and are not directly relevant to the benchmark.

While we do not show additional plots our the hyper-parameters chosen for our algorithms in Tables 7, 10, 9, and 10 were chosen by sweeping 168 different hyper-parameter settings across 3 continuous and 1 discrete hyper-parameter for PPO based algorithms and 2 continuous and three discrete hyper-parameters for the SAC based algorithms.

Table 7: Hyperparameter settings for IPPO algorithm

Hyperparameter	**Value**
Number of steps per rollout	64
Number of parallel environments	1024
PPO epochs per update	4
Number of minibatches	4
Learning rate	1×10^{-3}
Learning rate annealing	False
Entropy coefficient	1×10^{-4}
PPO clipping epsilon (ϵ)	0.31
Scale clipping epsilon	False
Ratio clipping epsilon	False
Discount factor (γ)	0.99
GAE lambda (λ)	0.95
Value function coefficient	1.0
Max gradient norm	0.5
Adam optimizer epsilon	1×10^{-8}

Table 8: Hyperparameter settings for MAPPO algorithm

Hyperparameter	**Value**
Number of steps per rollout	128
Number of parallel environments	1024
PPO epochs per update	4
Number of minibatches	4
Learning rate	4.4×10^{-3}
Learning rate annealing	False
Entropy coefficient	2.7×10^{-4}
PPO clipping epsilon (ϵ)	0.11
Scale clipping epsilon	False
Ratio clipping epsilon	False
Discount factor (γ)	0.99
GAE lambda (λ)	0.95
Value function coefficient	1.0
Max gradient norm	0.5
Adam optimizer epsilon	1×10^{-8}

Table 9: Hyperparameter settings for MASAC algorithm

Hyperparameter	**Value**
Number of steps per rollout	256
Number of parallel environments	64
Exploration steps	5000
Policy update delay	4
Replay buffer size	10^6
Batch size	128
Policy learning rate	3×3^{-4}
Q-function learning rate	1×2^{-4}
Alpha learning rate	3×3^{-4}
Max gradient norm	10
Target smoothing coefficient (τ)	0.005
Discount factor (γ)	0.99
SAC updates per iteration	32
Rollout length	8
Automatic entropy tuning (Autotune)	True
Target entropy scale	5.0
Initial alpha	0.1

Table 10: Hyperparameter settings for ISAC algorithm

Hyperparameter	**Value**
Exploration steps	5000
Policy update delay	4
Replay buffer size	10^6
Batch size	128
Policy learning rate	3×10^{-4}
Q-function learning rate	1×10^{-3}
Alpha learning rate	3×10^{-4}
Max gradient norm	10
Target smoothing coefficient (τ)	0.005
Discount factor (γ)	0.99
SAC updates per iteration	32
Rollout length	8
Automatic entropy tuning (Autotune)	True
Target entropy scale	5.0
Initial alpha	0.1

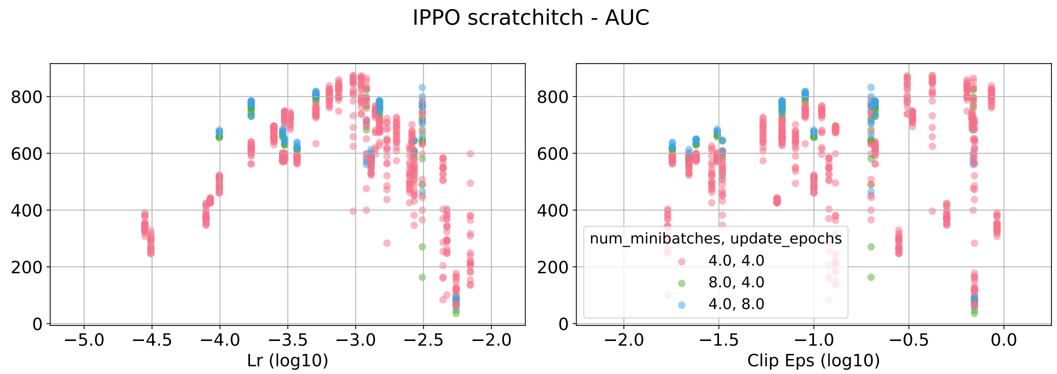


Figure 9: Area under curve returns (mean training returns) for IPPO in the scratching task. We show two plots for continuous hyper-parameters and group by discrete hyper-parameters

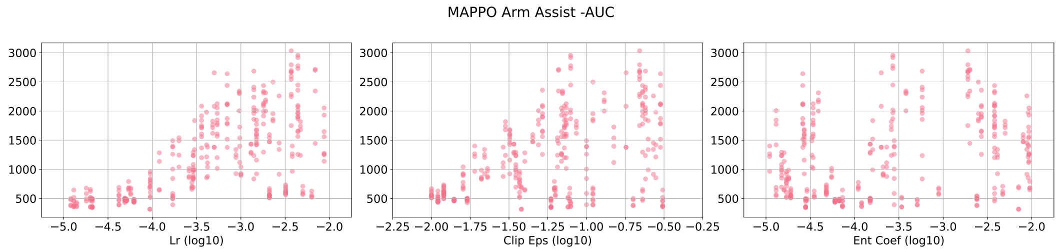


Figure 10: Area under curve returns (mean training returns) for MAPPO in the arm assist task. We show three plots for continuous hyper-parameters.

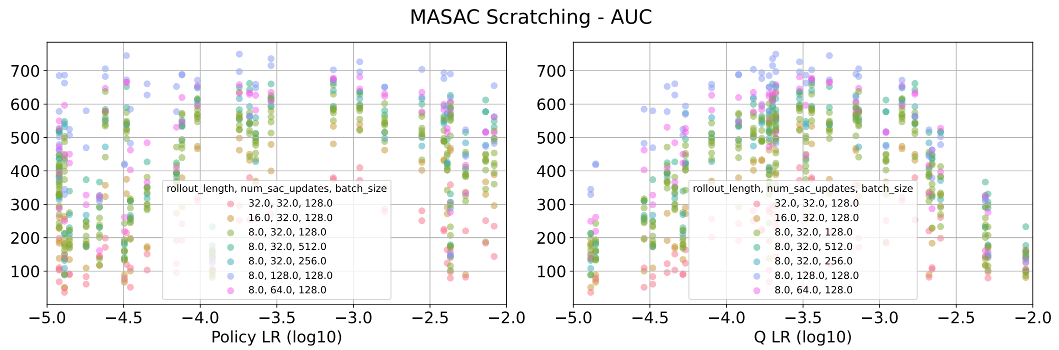


Figure 11: Area under curve (mean training returns) for MASAC on the Scratchitch task. We show two plots for the policy and critic learning rates. The hyper-parameter values are on the X-axis while AUC returns on the Y axis. Points are grouped by further hyper-parameter settings

C Additional ZSC Results

In Figure 14 we show the ZSC results for SAC in all three environments. We note similar trends to PPO in that the ZSC performance is very strong and there is no real discrepancy between the train and test sets.

D Additional Related Work (Robot Learning)

Robot Learning There is a significant interest in using learning as a means of designing generalist robot policies (Octo Model Team et al., 2024; Kim et al., 2024; Black et al., 2024) that enable

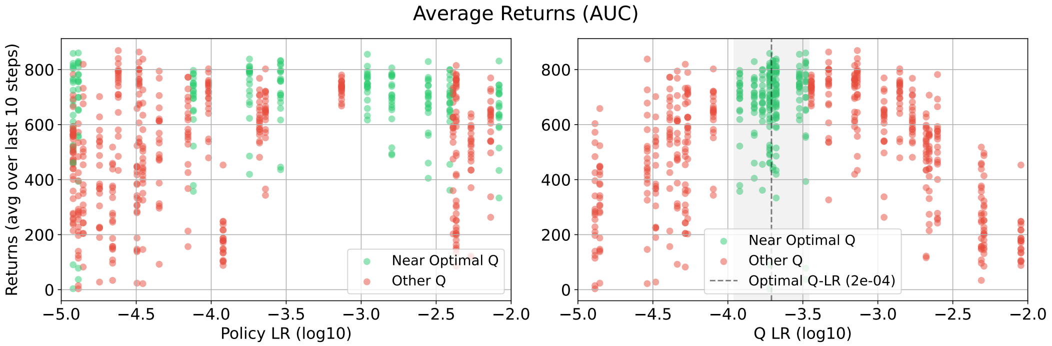


Figure 12: Ablation to check whether the Q-learning rate is more important for than the Policy learning rate for MASAC. We group points based on the proximity to the best Q value we find from our sweeps.

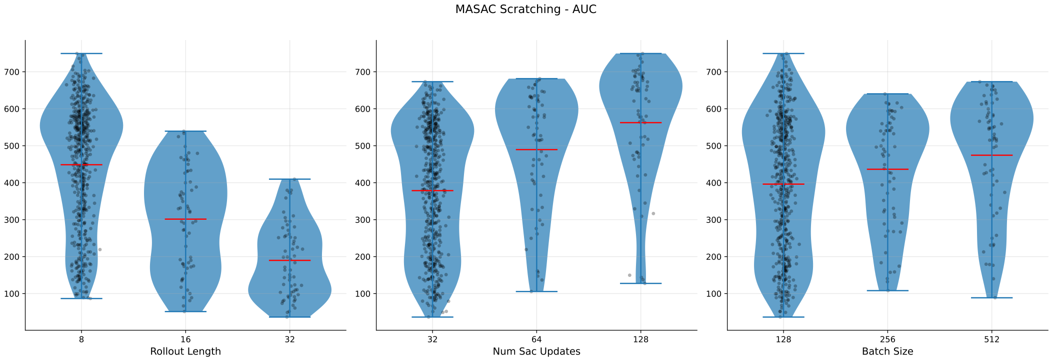
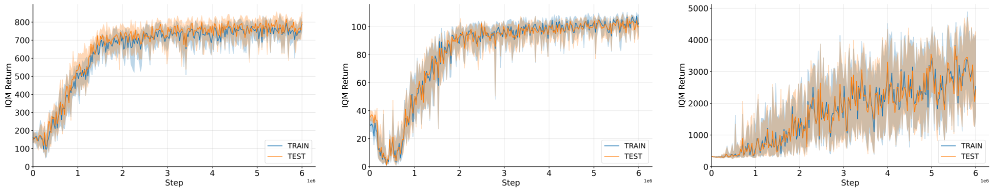


Figure 13: Violin plots showing flattened hyperparameter returns and grouped by different hyperparameter settings. In general this shows small improvements when increasing the number of epochs and batch size and worse performance for larger rollout lengths.



(a) *Scratching*

(b) *Bed Bathing*

(c) *Arm Assist*

Figure 14: ZSC Performance of SAC on all three tasks. Showing IQM returns and stratified 95% bootstrapped CI across 16 seeds.

robots to act in the real world. These policies are learned using RL (Kober et al., 2013) and imitation learning (Osa et al., 2018) by utilizing data collected in the real world (O'Neill et al., 2024), simulation (Mandlekar et al., 2023; Maddukuri et al., 2025), and the internet (McCarthy et al., 2024). There exist various robot learning frameworks and benchmarks focusing on tasks in table-top manipulation and mobile robotics (Chen et al., 2022; James et al., 2020; Zhu et al., 2020a; Sferrazza et al., 2024) with increased interest in everyday household environments (Li et al., 2021; Szot et al., 2021; Gu et al., 2023; Nasiriany et al., 2024). Assistax focuses on human-robot interaction scenarios requiring close contact and coordination between human and the robot (Chao et al., 2022; Thumm et al., 2024). Our assistive tasks are directly inspired by Assistive Gym (Erickson et al., 2019b) but our design philosophy is different driven by a need to design RL-first benchmark that can leverage hardware acceleration. Applying contemporary robot learning in human-robot interaction scenarios is still challenging partially due to a lack of efficient simulators that capture physical interaction with an active human (Tang et al., 2025). Assistax aims to fill this gap by providing a benchmark which allows for evaluating and developing novel robot learning techniques within the context of physical human-robot interaction.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. URL https://www.marl-book.com.

Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. The Hanabi Challenge: A New Frontier for AI Research. https://arxiv.org/abs/1902.00506v2, February 2019.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning, December 2019.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning, September 2022.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *CoRR*, abs/2410.24164, 2024. DOI: 10.48550/ARXIV.2410.24164. URL https://doi.org/10.48550/arXiv.2410.24164.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Yu-Wei Chao, Chris Paxton, Yu Xiang, Wei Yang, Balakumar Sundaralingam, Tao Chen, Adithyavairavan Murali, Maya Cakmak, and Dieter Fox. HandoverSim: A simulation framework and benchmark for human-to-robot object handovers. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

Tiffany L. Chen, Matei Ciocarlie, Steve Cousins, Phillip M. Grice, Kelsey Hawkins, Kaijen Hsiao, Charles C. Kemp, Chih-Hung King, Daniel A. Lazewatsky, Adam E. Leeper, Hai Nguyen, Andreas Paepcke, Caroline Pantofaru, William D. Smart, and Leila Takayama. Robots for humanity: using assistive robotics to empower people with disabilities. *IEEE Robotics & Automation Magazine*, 20 (1):30–39, 2013. DOI: 10.1109/MRA.2012.2229950.

Yuanpei Chen, Yaodong Yang, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen Marcus McAleer, Hao Dong, and Song-Chun Zhu. Towards human-level bimanual dexterous manipulation with reinforcement learning. In *Thirty-sixth Confer-*

ence on Neural Information Processing Systems Datasets and Benchmarks Track, 2022. URL https://openreview.net/forum?id=D29JbExncTP.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms. In *Reproducibility in Machine Learning, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryx0N3IaIV.

Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336345. DOI: 10.1145/2776880.2792704. URL https://doi.org/10.1145/2776880.2792704.

Zackory Erickson, Henry M. Clever, Vamsee Gangaram, Greg Turk, C. Karen Liu, and Charles C. Kemp. Multidimensional capacitive sensing for robot-assisted dressing and bathing. In *16th IEEE International Conference on Rehabilitation Robotics, ICORR 2019, Toronto, ON, Canada, June 24-28, 2019*, pp. 224–231. IEEE, 2019a. DOI: 10.1109/ICORR.2019.8779542. URL https://doi.org/10.1109/ICORR.2019.8779542.

Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C. Karen Liu, and Charles C. Kemp. Assistive Gym: A Physics Simulation Framework for Assistive Robotics, October 2019b.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, August 2018.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "Other-Play" for Zero-Shot Coordination, May 2021.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2024. URL https://proceedings.mlr.press/v270/kim25c.html.

Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *Int. J. Robotics Res.*, 32(11):1238–1274, 2013. DOI: 10.1177/0278364913495721. URL https://doi.org/10.1177/0278364913495721.

Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45716–45743, 2023.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL http://github.com/RobertTLange/gymnax.

Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, C. Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. 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. 455–465. PMLR, 2021. URL https://proceedings.mlr.press/v164/li22b.html.

Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning, 2018. URL https://arxiv.org/abs/1712.09381.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, Scott Reed, Ken Goldberg, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, Los Angeles, CA, USA, 2025.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021. URL https://arxiv.org/abs/2108.10470.

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. In *7th Annual Conference on Robot Learning*, 2023.

Robert McCarthy, Daniel Chee Hian Tan, Dominik Schmidt, Fernando Acero, Nathan Herr, Yilun Du, Thomas George Thuruthel, and Zhibin Li. Towards generalist robot learning from internet video: A survey. *CoRR*, abs/2404.19664, 2024. DOI: 10.48550/ARXIV.2404.19664. URL https://doi.org/10.48550/arXiv.2404.19664.

Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V. Albrecht. A Survey of Ad Hoc Teamwork Research, August 2022.

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.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang

Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Frans A. Oliehoek and Christopher Amato. The Decentralized POMDP Framework. In Frans A. Oliehoek and Christopher Amato (eds.), *A Concise Introduction to Decentralized POMDPs*, pp. 11–32. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28929-8. DOI: 10.1007/978-3-319-28929-8_2.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alexander Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew E. Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Paul Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Haoshu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I. Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J. Joshi, Niko Sünderhauf, Ning Liu, Norman Di Palo, Nur Muhammad (Mahi) Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R. Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham D. Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Liangwei Xu, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and RT-X models : Open x-embodiment collaboration.

In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, pp. 6892–6903. IEEE, 2024. DOI: 10.1109/ICRA57147.2024.10611477. URL https://doi.org/10.1109/ICRA57147.2024.10611477.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Found. Trends Robotics*, 7(1-2):1–179, 2018. DOI: 10.1561/2300000053. URL https://doi.org/10.1561/2300000053.

Arrasy Rahman, Jiaxun Cui, and Peter Stone. Minimum coverage sets for training robust ad hoc teamwork agents, 2024. URL https://arxiv.org/abs/2308.09595.

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, June 2018.

Paul Richmond, Robert Chisholm, Peter Heywood, Mozhgan Kabiri Chimeh, and Matthew Leach. FLAME GPU 2: A framework for flexible and performant agent based simulation on gpus. *Softw. Pract. Exp.*, 53(8):1659–1680, 2023. DOI: 10.1002/SPE.3207. URL https://doi.org/10.1002/spe.3207.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schröder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert T. Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu, and Jakob N. Foerster. Jaxmarl: Multi-agent RL environments and algorithms in JAX. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/5aee125f052c90e326dcf6f380df94f6-Abstract-Datasets_and_Benchmarks_Track.html.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Tjarko Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktaschel, Chris Lu, and Jakob Nicolaus Foerster. JaxMARL: Multi-Agent RL Environments and Algorithms in JAX, November 2024b.

Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *CoRR*, abs/1902.04043, 2019. URL http://arxiv.org/abs/1902.04043.

Neil Savage. Robots rise to meet the challenge of caring for old people. *Nature*, 601:S8 – S10, 2022. URL https://api.semanticscholar.org/CorpusID:246064292.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. HumanoidBench: Simulated Humanoid Benchmark for Whole-Body Locomotion and Manipulation, June 2024.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. ISSN 1476-4687. DOI: 10.1038/nature16961.

Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):1504–1509, July 2010. ISSN 2374-3468, 2159-5399. DOI: 10.1609/aaai.v24i1.7529.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John M. Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel X. Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 251–266, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/021bbc7ee20b71134d53e20206bd6feb-Abstract.html.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018. ISBN 978-0-9923747-4-7. DOI: 10.15607/RSS.2018.XIV.010.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 28698–28699. AAAI Press, 2025. DOI: 10.1609/AAAI.V39I27.35095. URL https://doi.org/10.1609/aaai.v39i27.35095.

Jakob Thumm, Felix Trost, and Matthias Althoff. Human-Robot Gym: Benchmarking Reinforcement Learning in Human-Robot Collaboration, June 2024.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017. DOI: 10.1109/IROS.2017.8202133.

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, October 2012. DOI: 10.1109/IROS.2012.6386109.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. DOI: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, November 2019. ISSN 1476-4687. DOI: 10.1038/s41586-019-1724-z.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL http://github.com/google-deepmind/mujoco_menagerie.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carlo Sferrazza, Yuval Tassa, and Pieter Abbeel. Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer., 2025. URL https://github.com/google-deepmind/mujoco_playground.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020a.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, Yifeng Zhu, and Kevin Lin. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020b.