MANISKILL3: GPU PARALLELIZED ROBOTICS SIMULATION AND RENDERING FOR GENERALIZABLE EMBODIED AI

Stone Tao[1,3], **Fanbo Xiang**[1,3], **Arth Shukla**[1,3], **Yuzhe Qin**[1], **Xander Hinrichsen**[1]
Xiaodi Yuan[1,3], **Chen Bao**[2], **Xinsong Lin**[1], **Yulin Liu** [1,3], **Tse-kai Chan**[1], **Yuan Gao**[1]
Xuanlin Li[1], **Tongzhou Mu**[1], **Nan Xiao**[1], **Arnav Gurha**[1], **Zhiao Huang**[1,3]
Roberto Calandra[4], **Rui Chen**[5], **Shan Luo**[6], **Hao Su**[1,3]
[1]University of California San Diego, [2] Carnegie Mellon University, [3] Hillbot, [4]TU Dresden
[5]Tsinghua University, [6] King's College London

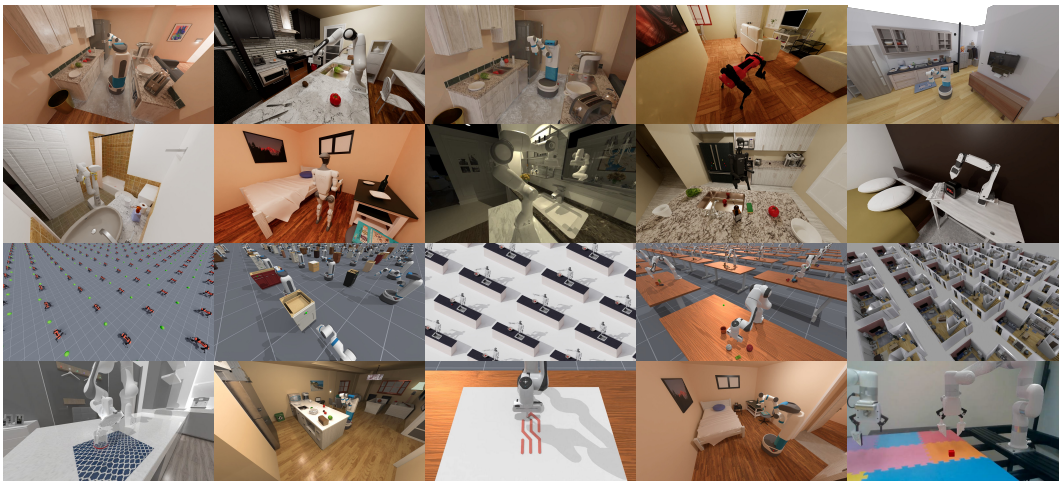

Figure 1: Multiple distinct task categories are displayed, ranging from room-scale tasks to humanoid interactions and drawing tasks. Every task shown is GPU-parallelized, simulating + rendering at state-of-the-art speeds and GPU memory efficiency. Scenes are from ReplicaCAD and AI2-THOR.

ABSTRACT

Simulation has enabled unprecedented compute-scalable approaches to robot learning. However, many existing simulation frameworks typically support a narrow range of scenes/tasks and lack features critical for scaling generalizable robotics and sim2real. We introduce and open source ManiSkill3, the fastest state-visual GPU parallelized robotics simulator with contact-rich physics targeting generalizable manipulation. ManiSkill3 supports GPU parallelization of many aspects including simulation+rendering, heterogeneous simulation, point-clouds/voxels visual input, and more. Simulation with rendering on ManiSkill3 can run 10-1000x faster with 2-3x less GPU memory usage than other platforms, achieving up to 30,000+ FPS in benchmarked environments due to minimal python/pytorch overhead in the system, simulation on the GPU, and the use of the SAPIEN parallel rendering system. Tasks that used to take hours to train can now take minutes. We further provide the most comprehensive range of GPU parallelized environments/tasks spanning 12 distinct domains including but not limited to mobile manipulation for tasks such as drawing, humanoids, and dextrous manipulation in realistic scenes designed by artists or real-world digital twins. In addition, millions of demonstration frames are provided from motion planning, RL, and teleoperation. ManiSkill3 also provides a comprehensive set of baselines that span popular RL and learning-from-demonstrations algorithms. Website: http://maniskill.ai/

1 INTRODUCTION

One of the grand challenges of robotics is robust and generalized manipulation. However, unlike vision and language research, there are still no good datasets for robotic manipulation that can be trained on. One approach has been to create human-scalable real-world teleoperation tools (Fu et al., 2024; Cheng et al., 2024) to then perform imitation learning. Another is to set up real-world reinforcement learning to fine-tune trained offline policies (Feng et al., 2023). However, real-world imitation learning approaches require enormous amounts of data that are infeasible to collect efficiently at low costs only to achieve relatively low success rates that are otherwise impractical for real-world deployment (Zhao et al., 2024). Real-world reinforcement learning approaches are promising, but require extensive setup in the real world to generate real-world rewards/success and environment resets.

GPU parallelized simulations such as Isaac (Makoviychuk et al., 2021) and Mujoco's MJX (Todorov et al., 2012) have made massive advancements in solving some robotics problems such as robot locomotion by training in large-scale GPU parallelized simulations with reinforcement learning (RL) (Rudin et al., 2021). GPU parallelized simulation makes data incredibly cheap to generate. However, when it comes to manipulation, success is often limited to narrower ranges of manipulation tasks and typically requires strong state estimation (Handa et al., 2023) to replace visual inputs like RGB or pointcloud. Existing GPU simulators have limitations that hinder the generalization and scalability of previous work. These simulators lack support for heterogeneous simulation, where each parallel environment contains different scenes. Additionally, they often don't support fast parallel rendering capabilities. As a result, algorithms like reinforcement learning (RL) that operate on visual input train too slowly to be practical.

The core contributions of ManiSkill3 that set it apart from existing simulators are as follows:

1) State-of-the-art GPU Parallelized Simulation and Rendering: RL algorithms like PPO (Schulman et al., 2017) can now solve visual tasks in 10-1000x less time it would have taken on other simulators due to fast parallel rendering and low overhead in the system design of ManiSkill3, leading to highly efficient use of the GPU. Depending on task the simulation + rendering FPS can reach up to 30,000+, massively accelerating visual data collection by 10-1000x compared to other simulators. Importantly ManiSkill3 maintains extremely low GPU memory usage, typically 2-3x lower than that of other simulators which enables on device visual RL and larger neural networks during training.

2) Most comprehensive range of environments with 12 different categories of environments and 20+ different robots provided out of the box, all GPU parallelized: ManiSkill3 out of the box provides a diverse set of different types of environments including but not limited to mobile manipulation, room-scale scenes, drawing, and humanoid/bi-manual manipulation. We further support 20+ different robot embodiments out of the box such as quadrupeds, floating grippers, humanoids, and dextrous hands. Furthermore we support several sim2real and real2sim setups for manipulation. Importantly, extensive documentation/tutorials are provided to teach users on how to add new environments/robots, as well as how to make open-source contributions to expand the repository of simulated tasks/robots.

3) Heterogeneous Simulation for Generalizable Learning: ManiSkill3 makes it possible to simulate and render completely different objects, articulations, even entire room-scale scenes in each parallel environment. This is done thanks to a data-oriented system design and easy-to-use API to manage GPU memory of objects/articulations even if they may have different degrees of freedom.

4) Simple Unified API to Easily Manage and Build GPU Simulated Tasks: ManiSkill3 distinguishes itself from other GPU-parallelized robotics simulators and benchmarks by offering a user-friendly API for creating diverse robotics environments. Key improvements include object-oriented APIs and the elimination of complex tensor indexing. The platform provides feature-rich tooling to streamline various operations, such as domain randomization (e.g., camera poses, robot controllers), trajectory replay, action space conversion, and more.

5) Scalable Dataset Generation Pipeline from Few Demonstrations: For tasks in ManiSkill3 where reward design is difficult, we provide a pipeline that leverages demonstration efficient, wall-time fast, online imitation learning algorithms, to learn a generalized neural network policy from a few teleoperated/hardcoded demonstrations. The generalized task-specific neural network policy is then used to rollout many more demonstrations to form larger datasets.

2 Related Work

Feature	ManiSkill 3	Isaac Lab	RoboCasa	RLBench	OmniGibson	Habitat	AI2THOR
GPU Parallelized Simulation	✓	✓	✗	✗	✗	✗	✗
GPU Parallelized Rendering	✓	✓	✗	✗	✗	✗	✗
Heterogeneous GPU Parallelized Simulation	✓	✗	✗	✗	✗	✗	✗
Large Scale Demonstrations	✓	✗	✓	✓	✗	✗	✗
Realistic Object Physics	✓	✓	✓	✓	✓	✗	✗
Photorealistic Rendering	✓	✓	✓	✗	✓	✗	✗
Room-Scale Scenes	✓	✓	✓	✗	✓	✓	✓
Visual RL Baselines	✓	✓	✗	✗	✗	✓	✓
Trajectory replay/conversion	✓	✗	✗	✗	✗	✗	✗
Task Categories	12	5	2	1	2	1	1
Interactive GUI	✓	✓	✗	✗	✗	✓	✗

Table 1: Comparison of major features across different open-source robotics frameworks/tools

Robotics Simulation Frameworks: Isaac Lab (previously called Isaac Orbit) (Mittal et al., 2023) and Mujoco (Todorov et al., 2012) are some open-source general-purpose rigid-body GPU parallelized robotics simulators. Isaac Lab and Brax (Freeman et al., 2021) (which supports the Mujoco MJX backend) are the most similar to ManiSkill in that they provides out of the box environments for reinforcement learning/imitation learning, as well as APIs to build environments. There are robotics frameworks like Robocasa (Nasiriany et al., 2024), Habitat, (Szot et al., 2021), AI2THOR (Kolve et al., 2017), OmniGibson (Li et al., 2022), RLBench (James et al., 2020) that only have CPU simulation backends and thus run magnitudes slower than Isaac Lab and ManiSkill, limiting researchers to often only explore imitation learning/motion planning approaches instead of reinforcement learning/online learning from demonstrations methods. Isaac Lab relies on the closed source Isaac Sim framework for GPU parallelized simulation and rendering whereas ManiSkill relies on the open-source SAPIEN (Xiang et al., 2020) for the same features. Brax/Mujoco uses the MJX backend and currently does not have parallel rendering. Both Isaac Lab and ManiSkill use PhysX for GPU simulation.

Robotics Datasets: Amongst existing datasets there are typically two kinds, real-world and simulated datasets. Open-X (Collaboration et al., 2023) is one of the largest real-world robotics datasets but suffers from issues with inconsistent data labels and overall poor data quality. DROID (Khazatsky et al., 2024) addresses some of Open-X's problems by using a consistant data collection platform. However, both Open-X and DROID require immense amounts of human labor to collect data and are inherently difficult to scale up to the sizes of typical vision/language datasets. Among simulated datasets, frameworks like AI2-THOR (Kolve et al., 2017), and OmniGibson (Li et al., 2022) have complex room-scale scenes but do not readily provide demonstrations or ways to generate large-scale demonstrations for use in robot learning. Robocasa has a myriad of tasks and realistic room-scale scenes, but further leverages MimicGen (Mandlekar et al., 2023) to scale human teleoperated demonstrations by generating new demonstrations.

ManiSkill sources large-scale demonstrations through a combination of different methods. For easier tasks, motion planning and rewards for RL are used to generate demonstrations. For more com-

plex tasks without easily defined motion planning scripts or reward functions, ManiSkill3 relies on online learning from demonstrations algorithms like RLPD (Ball et al., 2023) and RFCL (Tao et al., 2024), which are more flexible compared to MimicGen used in Robocasa as MimicGen makes a number of assumptions about the task (end-effector action spaces, little to no geometric variations, engineered task stage indicators).

3 Core Features of ManiSkill3

ManiSkill3 is the most feature-rich GPU simulation framework compared to popular alternatives as shown in Table 1. For the largest features, we detail them in subsequent subsections below.

3.1 Unified GPU Parallelized Tasks Supported Out Of The Box

The engineering of ManiSkill3 enables it to easily support many different kinds of task categories via a flexible task-building API, contributing to a unified interface for GPU parallelized robotics simulation. Of the existing popular robotics simulators ManiSkill3 supports the most categories of different tasks. Concretely we categorize the 12 distinct categories as follows: Table top manipulation, mobile manipulation, room-scale scenes for manipulation, quadruped/humanoid locomotion, humanoid/bi-manual manipulation, multi-agent robotics, drawing/cleaning, dextrous manipulation, vision-tactile manipulation, classic control, digital twins, and soft body manipulation environments. All of these tasks are GPU parallelized and can be rendered fast in parallel as well, with examples of the tasks shown in Fig. 1. Each of these task categories has various optimizations done to run more accurately and faster to achieve state-of-the-art simulation speeds. Other simulators typically support a smaller subset of the type of tasks ManiSkill3 supports easily. Additional details on the exact optimizations/implementations and available robots are detailed in Appendix A.

3.2 GPU Parallelized Simulation and Rendering

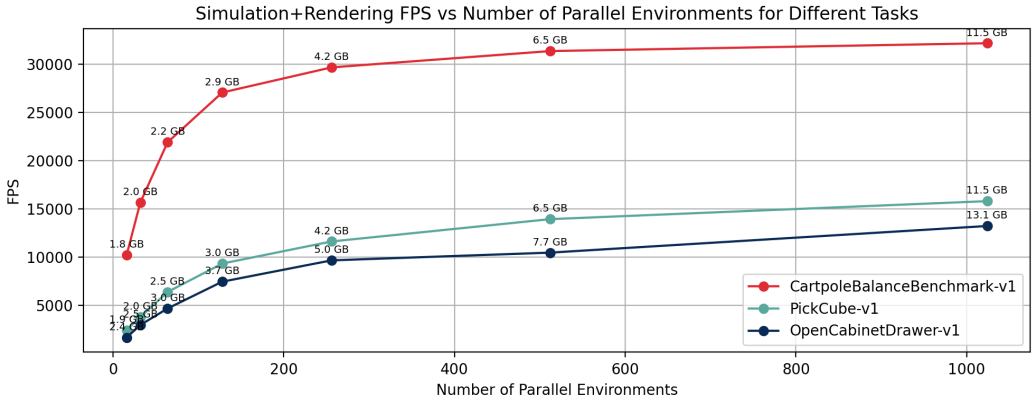

Figure 2: GPU Simulation+Rendering speeds of various tasks with a single 128x128 resolution camera that renders every 2 physics steps in the simulation. RGB, depth, and segmentation data are all simultaneously being rendered. The only big variations the between environments of the three curves are the objects and robots being simulated.

ManiSkill3 distinguishes itself from its predecessors and other robotics simulators by offering robust support for GPU-parallelized simulation and rendering. ManiSkill3 is the first benchmark to enable fast RL from visual inputs on complex robot manipulation tasks, with Isaac Lab recently adding a similar feature. Tasks such as picking up a cube or controlling a quadruped to reach a goal from pixel inputs are now solved on the order of minutes instead of hours. RL training results/speed are detailed in Section 4.3.

The performance results shown in Figure 2 are the results after simulating + rendering RGB, depth, and segmentation data simultaneously for various tasks. In terms of speed and GPU memory use,

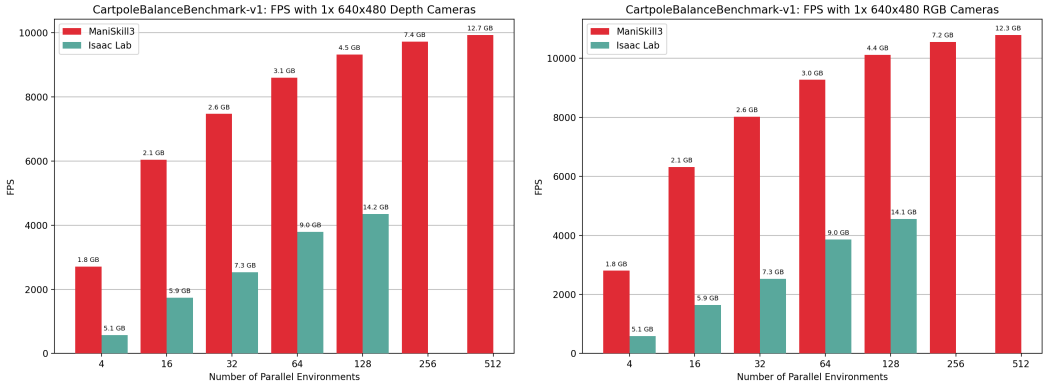

Figure 3: GPU Simulation+Rendering of RGB/Depth speeds of the Cartpole environment with a 640x480 resolution camera in ManiSkill3 and Isaac Lab. Annotated numbers indicate GPU memory usage, with no data points beyond 128 environments for Isaac Lab due to running out of GPU memory. Note that this rendering setting mimics that of real world datasets collected in Open-X. Speed is dependent on a few factors, primarily the number of objects, geometry complexity of each object, as well as simulation/rendering configurations which can be tuned for speed or accuracy. As a result, it is possible the numbers/trends here may not hold for every environment.

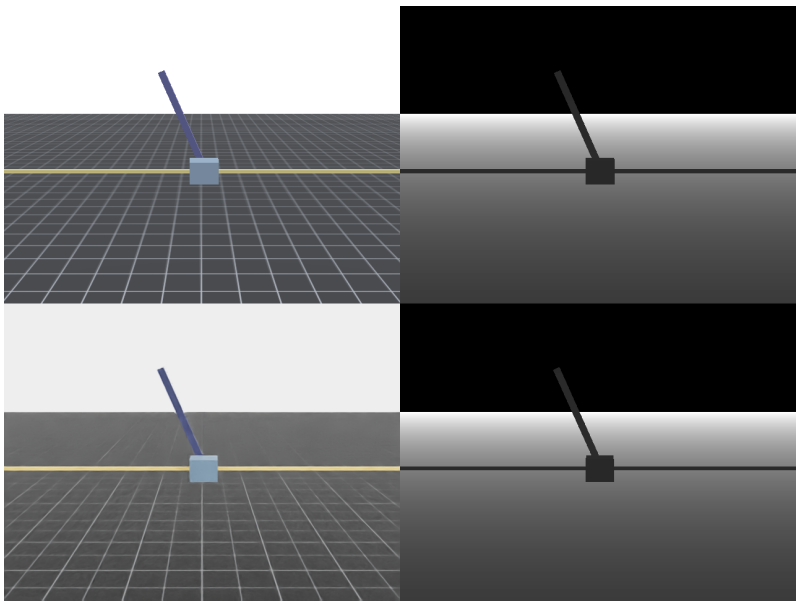

Figure 4: Comparison of ManiSkill3 (Top row) and Isaac Lab (Bottom row) parallel rendering 640x480 RGB and depth image outputs of the Cartpole benchmark task.

Figure 3 shows ManiSkill3 outperforms Isaac Lab, particularly when it comes to rendering common real-world camera resolutions which are important for sim2real and real2sim transfer. We compare against the latest Isaac Lab v1.2.0 released on Sept. 20, 2024. In particular, with 128 parallel environments for the benchmarked task, ManiSkill3 uses just 4.4GB of GPU memory whereas Isaac Lab uses 14.1GB. The memory efficiency of the ManiSkill3 platform allows for more room for e.g. RL replay buffers or larger neural network models. Training and inference can be kept extremely optimized on a single GPU as a result without needing to store any data on the CPU. From experimentation with visual RL, we find that GPU memory efficiency becomes much more important as the FPS gains from more parallel environments become marginal. GPU memory efficiency is especially important for off-policy algorithms like TD-MPC2 (Hansen et al., 2024) and SAC (Haarnoja et al., 2018) that typically maintain replay buffer sizes on the order of $10^5 \sim 10^6$ frames. For more

in-depth performance benchmarking results and comparisons of rendered outputs, see Appendix B. We acknowledge that this comparison is not strictly apples-to-apples due to differences in rendering techniques. Isaac Lab employs ray-tracing for parallel rendering, while the ManiSkill3 results are generated using SAPIEN's rasterization renderer (see Figure 4 for a visual comparison), although ManiSkill3 also supports a ray-tracing mode without parallelization. Ray-tracing generally offers greater flexibility in balancing rendering speed and quality through the adjustment of parameters such as samples per pixel. It's worth noting that the Isaac Lab data presented here uses the fastest rendering settings, although it can be easily tuned to achieve better rendering quality that may be helpful for sim2real. Despite the use of different rendering techniques, we believe this experiment provides a meaningful basis for comparison.

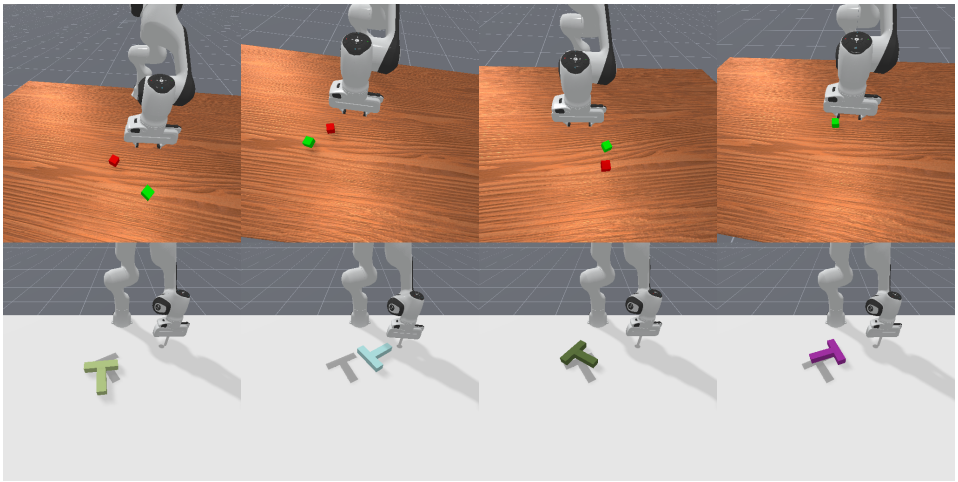

Figure 5: Parallel rendering outputs of 1024 parallel environments for the StackCube and PushT tasks with a subset of 4 them visualized here. Original renders are size 128x128, images shown are up-scaled for clarity. Top-row shows camera pose randomization and bottom row shows texture randomization, renderered depth/segmentation data is not shown here.

GPU parallelized simulation and rendering further enable an entirely new regime of possible domain randomizations. For example you can now quickly render 1000s of different cameras, each with different extrinsics/intrinsics, mounted/fixed, as well as randomize object textures in each of the parallel environments. A subset of 4 out of 1024 environments is shown rendered by our parallel renderer in Figure 5 with different settings. This type of visual diversity in simulation data generation enables much faster training of more visually robust policies and is critical for sim2real applications. Furthermore, ManiSkill3 supports parallelized rendering of voxel and pointcloud formats to support 3D robot learning approaches.

Finally, ManiSkill3 enables extremely fast visual digital twins. For example, ManiSkill3 implements 4 of the environments in SIMPLER (Li et al., 2024) which are evaluation digital twins that enable the evaluation of generalist robotic policies trained on real-world data like Octo (Octo Model Team et al., 2024) and Robotics Transformers (RT) (Brohan et al., 2022). Currently, ManiSkill3 digital twins can evaluate models like Octo at 60x to 100x the speed of the real world without human supervision and runs approximately 10x faster than the original digital twin implementations in SIMPLER. The speed increase is due to fast and efficient parallel rendering of large camera resolutions (640x480) and flexible GPU parallelized controllers to match most real-world robot arms/manipulators. More details on sim2real and real2sim support are covered in Section 3.4.

3.3 Heterogeneous GPU Simulation

ManiSkill3 is so far the only simulation framework that supports heterogeneous GPU simulation. This is the feature of being able to simulate different object geometries, different numbers of objects, and different articulations with different DOFs across different parallel environments. For example, in the Open Cabinet Drawer task, for each parallel environment, we build a different cabinet (all with different DOFs) and sample a random drawer link that needs to be opened to succeed. In the Pick

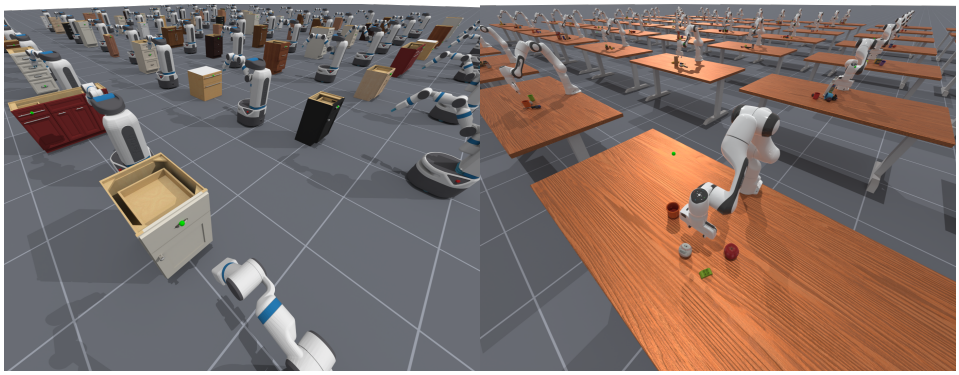

Figure 6: Example tasks in ManiSkill3 showing heterogeneous GPU simulation with different DoF articulations and/or different numbers of objects being simulated in each parallel environment.

Clutter YCB task, we sample a different number of YCB objects in each parallel environment and sample one random object out of the clutter that is meant to be picked up. ManiSkill3 easily supports this kind of simulation and further supports rendering these different scenes in parallel all at once with an example 3rd view rendering, as shown in Figure 6. Heterogeneous GPU simulation enables more generalizable learning for manipulation as algorithms like those in RL can now simultaneously train on every single object from the YCB object dataset (Çalli et al., 2015) or the PartNet Mobility Dataset of cabinets (Mo et al., 2019).

3.4 SIM2REAL AND REAL2SIM FOR ROBOT MANIPULATION

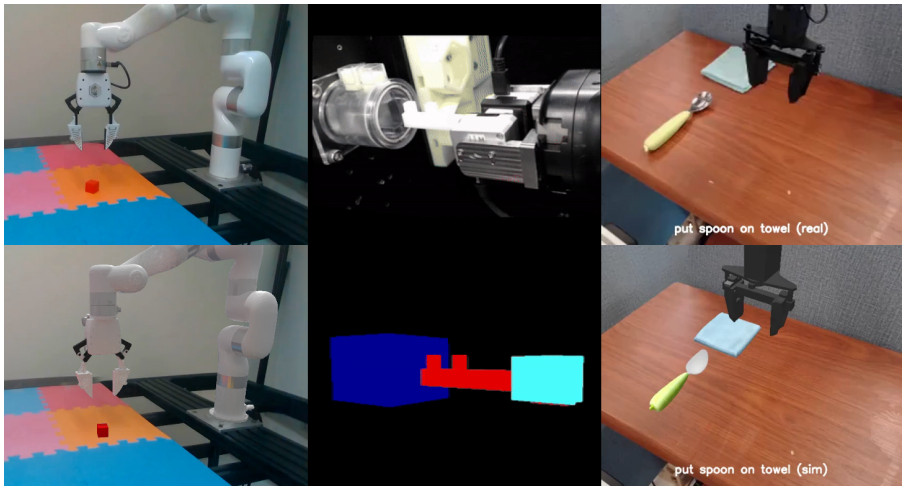

Figure 7: Three different kinds of digital twins in ManiSkill3. Top row shows the real-world setup and bottom row shows the digital twin. Left: Digital twin of a cube picking task. Middle: Digital twin of the vision-tactile simulation of a key insertion task. Right: Real2sim digital twin of spoon placing task.

Towards the goal of robust real-world robotics beyond simulation, we verify sim2real and real2sim are both possible using ManiSkill3 via digital twins on some tasks. Figure 7 showcases several digital twins supported with real world counterparts. For the cube picking task, we segment out relevant objects to feed into a PointNet model and train with PPO on only simulated data, ultimately achieving a real world success rate of 95% over a 50cm x 50cm workspace. For the vision-tactile peg insertion task, we simulate the tactile sensor made of silicone as a softbody and refer readers to the results showcased in the original work by Chen et al. (2024) for those environments, which achieved a 95.08% success rate in the real world. Finally, for the real2sim digital twins we evaluate Octo and RT-1X on the ManiSkill3 GPU parallelized version of 4 tasks in SIMPLER (Li et al.,

2024). We achieve a correlation between real-world success rates and simulation success rates of 0.9284 and a Mean Maximum Rank Violation (MMRV) value 0.0147, which is close to the original values reported in SIMPLER. See Appendix A.11 for more details on real2sim evaluations.

3.5 Simple Unified API for Building GPU Simulated Robotics Tasks

A core reason behind the flexibility of the ManiSkill3 system to support so many different distinct task categories is the clean and simple API for task-building. The simple API also enables users to easily build and customize their own robotics tasks.

3.5.1 Object-Oriented API for Articulations, Links, Joints, and Actors

Figure 8: Code comparison for computing a grasp position on a cabinet handle and the joint angle of the cabinet drawer in 3 different GPU simulation frameworks.

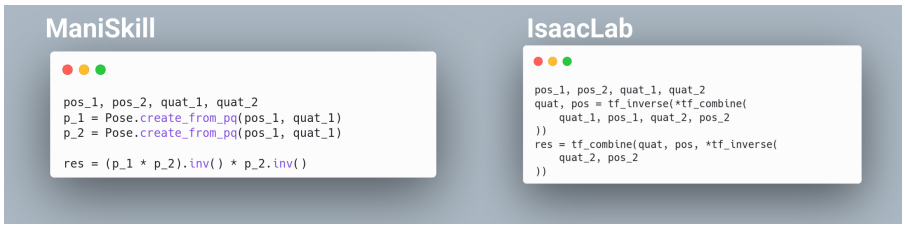

Figure 9: Code comparison for manipulating batched poses

ManiSkill3 is the only framework with a complete object-oriented API around the high-level articulations/actors down to individual links/joints and meshes. In contrast, IsaacGymEnvs requires users to instantiates relevant GPU buffers for holding articulation state such as root pose and joint angles. Isaac Lab improves on this with a partially object-oriented articulation API that allows one to create an articulation object (e.g., for a cabinet). However, one still has to often play around with index values to get the relevant articulation data they need. We use a cabinet opening task as a case study. In a cabinet drawer opening task, to write good reward functions you need to access the drawer link's handle mesh's pose, as well as the joint angle between the drawer and the cabinet. A visual comparison of the 3 APIs (simplified from the actual code) is shown in Figure 8.

Furthermore, pose information in ManiSkill3 is object-oriented and stored as batched Pose objects, enabling an easy to read, method chaining pattern of programming for working with poses. For

the sake of an example, suppose that we have 2 poses P_1, P_2 and want to compute $(P_1P_2)^{-1}P_1^{-1}$, ManiSkill3 provides a much simpler and method chainable API to do this compared to Isaac Lab, as shown in Figure 9.

3.5.2 Robots and Controllers

ManiSkill3 supports both URDF and Mujoco MJCF definition formats natively and builds articulated robots based on the URDF/MJCF directly. For each robot, ManiSkill3 further provides a number of configurable controller options for both GPU parallelized joint position control and inverse-kinematic (IK) control, modified from ManiSkill2 for GPU simulation. ManiSkill3 builds upon the PyTorch Kinematics package (Zhong et al., 2024) to support inverse-kinematic based controllers parallelized on the GPU. These options are easily configured at runtime with either preset configurations or user-supplied controller configurations. Currently, there are 20+ different robots supported out of the box in ManiSkill3, a subset of which are visualized in Figure 11 in the Appendix. Finally, ManiSkill3 comes with extensive tutorials and examples of how to tune and optimize robots for fast simulation, which has often proven a stumbling block for those new to robot simulation importing complex robots for the first time. The tutorials highlight details from how to simulate mobile-bases to simulation hyperparameter tuning for improving simulation speed and/or accuracy.

3.6 Demonstration Datasets

We leverage a variety of approaches to collect/generate our demonstration datasets infinitely at scale. For the simplest tasks, we write and open source some motion planning-based solutions to generate demonstration data. Some tasks with easy-to-define reward functions have dense reward functions defined and converged RL policies are used to generate demonstration data. For more difficult tasks, we collect demonstration data (typically about 10 demonstrations) via teleoperation tools. Then, we use RFCL (Tao et al., 2024) or RLPD (Ball et al., 2023) to run fast online imitation learning and generate data from converged policies.

We further adapt the trajectory replay tool from ManiSkill2 to work with both CPU and GPU simulated demonstration data. The replay tool enables users to change the observations stored (e.g., state or rgbd), as well as modify the actions stored (e.g. joint position or delta end effector pose). The tool makes working with demonstration data more flexible in ManiSkill3, such as converting joint position actions generated by motion planning to other control modes like end-effector control. We are currently still in the process of generating demonstrations for more tasks.

4 Baselines and Results

ManiSkill3 provides several popular robot learning baselines that span 4 different categories:

Wall-time Efficient Reinforcement Learning: We include a torch based vectorized implementation of model-free RL algorithms PPO and SAC (Haarnoja et al., 2018), as well as the state-of-the-art model-based RL algorithm TD-MPC2 (Hansen et al., 2024). Configurations for baselines are tuned to minimize training wall time with no regard to sample efficiency.

Sample Efficient Reinforcement Learning: All of the RL baselines in the wall-time efficient setting are included here with configurations tuned towards more gradient updates and fewer environment steps to maximize sample efficiency.

Offline Imitation Learning: These are supervised learning and offline RL style baselines. We currently provide Behavior Cloning, offline variants of SAC, and the current state-of-the-art Diffusion Policy (Chi et al., 2023) as baselines.

Online Imitation Learning: Online imitation learning generally refers to algorithms that learn from demonstrations in addition to collecting online environment transitions. We currently provide the two state-of-the-art baselines in this area, Reinforcement Learning from Prior Data (RLPD) (Ball et al., 2023), and Reverse Forward Curriculum Learning (RFCL) (Tao et al., 2024). Note that RFCL leverages simulation state resets.

Furthermore, ManiSkill3 provides tools that establish clear specifications for how to evaluate policies that address common issues in other simulators that can often cause confusion.

4.1 Reinforcement Learning Training/Evaluation Setup

As part of an effort to unify baselines and robot simulation to allow easy comparison and research/testing, we ensure that all baselines report the same metrics and run the same evaluation environment setups. With shared metrics, this enables us to form a massive Weights and Biases (Wandb) online dashboard of every RL (and IL) algorithm's results and allows for easy cross-comparison.

We observe that in past simulators/research, there is often inconsistency in how episode truncations/terminations are handled and how to report environment success/fail. For example, Isaac Lab by default terminates episodes early during training before reaching the timelimit if the agent succeeds or fails, but many RL algorithms (especially off-policy ones) like TD-MPC2 and SAC model environments as having infinite horizon and do not terminate environments early due to success/fail. Moreover, Isaac Lab is inconsistent with whether termination means success or failure in different environments, as shown in their Franka Open Cabinet code and Anymal C Locomotion code.

To ensure better consistency in metrics reported not only by baselines provided in ManiSkill3, but also those reported by users, we provide an environment wrapper for evaluation environments that auto-records these defined metrics. We provide clear instructions on how to set it up, record results, and exactly what each metric means. Concretely, the wrapper ensures evaluation of policies never terminate environments early (it is permitted during training as early termination can accelerate some RL algorithms), and the wrapper records episode return, length, success/fail once, and success/fail at end metrics.

Success once is equivalent to early termination due to success and is set true if the environment achieves success once at any point during an episode before truncation. Success at the end is equivalent to recording whether the environment is in a success state at the final episode step before truncation. Similarly, fail once/at the end operate the same way but for failure conditions. Both "once" and "at the end" metrics are recorded to indicate a clear distinction in the types of success/failure a policy might achieve to avoid confusion.

Finally, we make a key distinction in sample-efficient and wall-time-efficient training results by explicitly tagging which training runs are using sample/wall-time-efficient configurations. Research into both sample efficiency and wall-time efficiency remains active in the robot learning community so we ensure this distinction is clearly labelled. The final results on the Wandb dashboard are still being uploaded.

4.2 Imitation Learning Training/Evaluation Setup

Mandlekar et al. (2021) show that imitation learning algorithm performance heavily depends on how the demonstrations were collected, particularly on how "multimodal" the data is. For example, many behavior cloning algorithms regress actions and perform poorly when trained on motion planning or human teleoperated data, but can perform very well if trained on data generated by a deterministic neural network. With this caveat in mind, we explicitly track in all our imitation learning (online and offline) baselines how many demonstrations are used (an integer), what type of demonstrations are used (neural net, motion planning, or human), and where the demonstrations are sourced from exactly (a longer description e.g. neural net trained via TD-MPC2, teleoperation via Meta-Quest VR). On the ManiSkill3 Weights and Biases online dashboard, these key configurations over the dataset can be searched/filtered, enabling users to easily compare algorithm performance on any demonstration dataset ablation they wish.

Furthermore, we provide shared code/evaluation harnesses that enable users to easily use a vectorized CPU backend or the GPU simulation backend for fast evaluation of imitation learning policies during training. Success/fail once/at the end metrics are similarly also required and reported automatically following the same evaluation setup as RL algorithms. The final results on the Wandb dashboard are still being uploaded.

4.3 Training Speed

We run experiments using the Proximal Policy Optimization (PPO) (Schulman et al., 2017) RL algorithm on the ManiSkill3 GPU simulation and the ManiSkill2 CPU simulation. ManiSkill2 was previously the fastest robotics simulation+rendering framework until ManiSkill3. The experiments

were run on an RTX-4090 GPU on the PickCube task, where a Franka robot arm must grasp a randomly initialized cube and hold it still at a random goal location. For the vision-based experiment no ground truth data like cube pose is provided and the RL policy must solve from proprioceptive information and a single 128x128 resolution RGB image rendered by the environment's 3rd-person camera. RL hyperparameters are tuned to achieve the fastest training time in both settings. Results in Figure 10 show that state and vision based training are massively accelerated with GPU simulation and rendering.

We note that while the speedups are around 5x in the results, if more scalable RL algorithms are used the speedups can be greater as PPO has worse sample-efficiency when using a high number of parallel environments compared to typical CPU simulation setups with less parallel environments.

Figure 10: Training time and sample-efficiency of PPO on GPU/CPU simulation.

5 Conclusion

ManiSkill3 introduces a state-of-the-art framework/benchmark for generalizable robotics simulation and rendering. ManiSkill3 runs faster, uses less GPU memory, and supports the most diverse range of robotics tasks compared to alternative simulators. In particular, ManiSkill3 offers unparalleled support for both sim2real and real2sim environments in manipulation tasks. We believe that this comprehensive approach will encourage the research community to tackle manipulation challenges more extensively through compute-scalable simulation. Furthermore, ManiSkill3 provides an easy-to-use object-oriented API for building GPU simulated heterogeneous tasks, democratizing access to scalable robot learning. Finally, demonstrations and RL/IL baselines with clearly defined metrics are open sourced for users to use.

Acknowledgments

The authors sincerely thank members of the Hao Su lab and Hillbot for great discussion and feedback, including but not limited to Nicklas Hansen, Rokas Bendikas, Jiayuan Gu, James Hou. The authors also thank members of the open source community for spotting early bugs, various contributions, and providing feedback on the framework. We especially acknowledge the generous support from Qualcomm Embodied AI Fund and Hillbot Inc. Stone Tao is supported in part by the NSF Graduate Research Fellowship Program grant under grant No. DGE2038238.

References

Philip J. Ball, Laura M. Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. 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. 1577–1594. PMLR, 2023. URL `https://proceedings.mlr.press/v202/ball23a.html`.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian

Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. *CoRR*, abs/2212.06817, 2022. doi: 10.48550/ARXIV.2212.06817. URL https://doi.org/10.48550/arXiv.2212.06817.

Berk Çalli, Aaron Walsman, Arjun Singh, Siddhartha S. Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. *CoRR*, abs/1502.03143, 2015. URL http://arxiv.org/abs/1502.03143.

Weihang Chen, Jing Xu, Fanbo Xiang, Xiaodi Yuan, Hao Su, and Rui Chen. General-purpose sim2real protocol for learning contact-rich manipulation with marker-based visuotactile sensors. *IEEE Transactions on Robotics*, 40:1509–1526, 2024. doi: 10.1109/TRO.2024.3352969.

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

Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://doi.org/10.15607/RSS.2023.XIX.026.

Open X-Embodiment Collaboration, 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, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, 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 Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu 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 Suenderhauf, 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 Mart'in-Mart'in, 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 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, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, 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, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

Yunhai Feng, Nicklas Hansen, Ziyan Xiong, Chandramouli Rajagopalan, and Xiaolong Wang. Finetuning offline world models in the real world. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, 2023.

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.

Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. Humanplus: Humanoid shadowing and imitation from humans. *CoRR*, abs/2406.10454, 2024. doi: 10.48550/ARXIV.2406.10454. URL https://doi.org/10.48550/arXiv.2406.10454.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, and Yashraj S. Narang. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pp. 5977–5984. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10160216. URL https://doi.org/10.1109/ICRA48891.2023.10160216.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. In *Conference on Robot Learning*, 2024.

Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X. Chang, and Manolis Savva. Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation. *arXiv preprint*, 2023.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1k: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=_8DoIe8G3t.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

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. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/28dd2c7955ce926456240b2ff0100bde-Abstract-round2.html.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. 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. 1678–1690. PMLR, 2021. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

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.

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.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

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 (RSS)*, 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.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. 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. 91–100. PMLR, 2021. URL https://proceedings.mlr.press/v164/rudin22a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Stone Tao, Arth Shukla, Tse-kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demo efficiency. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=w4rODxXsmM.

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. doi: 10.1109/IROS.2012.6386109.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy P. Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Softw. Impacts*, 6:100022, 2020. doi: 10.1016/J.SIMPA.2020.100022. URL https://doi.org/10.1016/j.simpa.2020.100022.

Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Tony Z. Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Seyed Kamyar Seyed Ghasemipour, Chelsea Finn, and Ayzaan Wahid. ALOHA unleashed: A simple recipe for robot dexterity. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=gvdXE7ikHI.

Sheng Zhong, Thomas Power, Ashwin Gupta, and Peter Mitrano. PyTorch Kinematics, February 2024.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *CoRR*, abs/2009.12293, 2020. URL https://arxiv.org/abs/2009.12293.

A Environments and Robots List

Figure 11: A sample of 16 of the robots supported in ManiSkill3 on both CPU and GPU simulation.

This section covers key implementation and optimization details of the general supported task categories. Moreover, comparisons are made when possible with other robot simulation frameworks that have similar tasks/categories. For a complete list with videos of every task, see ManiSkill3 Tasks Documentation. A sample of 16 of the supported robots, which include mobile manipulators, floating grippers, quadrupeds, etc. is displayed in Figure 11. For a complete list with details/pictures of every robot, see ManiSkill3 Robots Documentation.

A.1 Table Top Manipulation

Table-top manipulation is primarily related to controlling one or more robot arms to manipulate an object on a table. Robots like Franka Emika Panda and Universal Robots 5 fall under this category. Typical tasks may include picking up objects, inserting a peg, assembling a structure, pushing objects, etc.

Robosuite (Zhu et al., 2020) is an existing robotics framework with CPU simulation+rendering implementations of a few table top tasks. Isaac Lab (Mittal et al., 2023) has only one table-top manipulation task supported out of the box. ManiSkill3 has 10+ different table-top manipulation tasks,

some of which leverage heterogeneous parallel simulation where each parallel simulated+rendered environment has completely different numbers of objects with different geometries.

Implementation Details: All robot arms are modified to have certain impossible self-collisions disabled and some have their collision meshes modified for faster simulation.

A.2 Mobile Manipulation

Mobile manipulation here refers to tasks in which a robot arm has a mobile base. Robots like Fetch and Stretch fall into this category. Typical tasks may include placing objects on surfaces, opening cabinet doors/drawers, picking up objects off the ground, etc.

Implementation Details: The default robot supported is Fetch. The mobile base in particular is not simulated by driving the wheels, and is modeled similarly to AI2-THOR (Kolve et al., 2017) and Habitat (Szot et al., 2021) with one joint controlling forward/backward movement and another controlling rotation of the base. The Fetch robot definition and collision meshes have further been tuned to be simpler for faster simulation. Several impossible self-collisions between some links have been explicitly ignored to speed up simulation.

A.3 Room Scale Environments

ManiSkill3 provides out-of-the-box code to build the ReplicaCAD environment from Habitat (Szot et al., 2021) and all AI2-THOR environments (Kolve et al., 2017) using assets compiled by the authors of the Habitat Synthetic Scenes Dataset (Khanna et al., 2023). Photo-realism is also possible by turning on the ray-tracing shader options when creating the environment.

AI2-THOR (Kolve et al., 2017) and Habitat (Szot et al., 2021) have long-horizon mobile manipulation tasks but rely on slow CPU simulation / slow rendering systems, and do not support contact-rich physics for manipulation (only magical grasp) and do not look photorealistic. Robocasa (Nasiriany et al., 2024) has contact-rich long-horizon mobile manipulation tasks in photorealistic room-scale environments. However, Robocasa simulates and renders these scenes at around 25FPS as it does not use GPU parallelized simulation and rendering. In contrast ManiSkill3 can simulate the complex ReplicaCAD environment up towards 2000+ FPS with rendering.

Implementation Details: We further make several modifications to ReplicaCAD to make it completely interactive as some of the collision meshes for articulations were modelled incorrectly and thus did not support low-level grasping. Via CoACD (Wei et al., 2022) we run convex decomposition on objects in AI2-THOR scenes to generate simulatable non-convex collision meshes so those objects can e.g. be grasped and moved around correctly. Via manual annotation by ManiSkill3 authors, certain categories of objects in AI2-THOR are made to be kinematic so they cannot be moved around (e.g. tables, TVs, clocks, paintings) with the rest allowed to be dynamic to be fully simulated (e.g. apples, baseball bat, cups).

A.4 Locomotion

Locomotion here refers to controlling robot joints to move a robot from one location to another. Quadrupeds such as AnyMAL-C and humanoids such as Unitree-H1 fall into this category. Typical tasks may include standing still, moving to a target location, running as fast as possible, etc.

This type of task can be found in Isaac Lab which has several predefined locomotion tasks with tuned rewards and sim2real results.

Implementation Details: Similar to Isaac Lab, quadruped robots in locomotion tasks are modified such that the large majority of collision meshes are removed, leaving behind just the feet, ankles, and the body. Moreover following Isaac Lab, joint limits are significantly constrained such that random actions do not easily cause the robot to fall over.

A.5 Humanoid / Bi-manual Manipulation

Tasks here refer to the use of a humanoid embodiment such as the Unitree H1 robot or bi-manual robot embodiments for manipulation tasks.

This type of task can be found in benchmarks like RoboCasa and BiGym, although we note that RoboCasa and BiGym do not reliably support working RL setups and are not GPU parallelized. Isaac Lab currently does not have these types of tasks out of the box.

Implementation Details: For some tasks where the robot has legs, to simplify the task the legs are fixed in place so that robot learning methods can focus on manipulation only and train faster. Tasks still have the option to swap a version of the robot where all joints are controlled although they are much harder.

A.6 Multi-Agent Robots

Multi-Agent robots refer to support for controlling multiple different robots in the same simulation to perform a task. Setups such as multiple quadrupeds or robot arms fall into this category. A common task is the handover of objects.

This type of task can be found in Robosuite. Isaac Lab supports this type of task, but not out of the box. Past environments based on older versions of Isaac have example tasks with multiple robot arms/hands running on GPU simulation (Chen et al., 2022).

Implementation Details: By default the action space is a dictionary action space in multi-agent environments with a dictionary key for each controllable agent. This action space can easily be flattened into a single vector if necessary via a environment wrapper.

A.7 Drawing/Cleaning

Drawing/Cleaning refers to tasks for dynamically "adding/removing" objects to simulate the effect of drawing or cleaning. A task could be to draw the outline of a shape on a canvas or clean dirty spots on a table surface.

ManiSkill3 is the only framework that supports this kind of task with GPU parallelization.

Implementation Details: The drawing/cleaning effect is achieved by building ahead of time 1000s of small thin cylinders that represent "ink" or dirty spots. For a drawing task, all of these cylinders are hidden away from the camera view. When a robot moves a drawing tool close to a surface/canvas, the cylinders have their pose set to be on top of the surface right under where the drawing tool is. For a cleaning task, all the cylinders/dirty spots are visible and removed once the cleaning tool moves over the dirty spot. Currently the drawing/cleaning environments in ManiSkill3 do not require intricate grasping (nor is it the focus) of the drawing/cleaning tool, so the solver position/velocity iteration values are tuned down.

A.8 Dextrous Manipulation

Dextrous Manipulation refers to tasks often involving multi-fingered hands and dense/rich contacts occurring during manipulation. An example task is in-hand rotation.

This type of task has been heavily explored and exists in Isaac Lab and ManiSkill3 with GPU parallelization.

Implementation Details: Not many special optimizations are made here. Tactile sensing is further provided in environments via touch sensors placed on the dextrous hand at various points.

A.9 Vision Tactile Manipulation

Vision Tactile Manipulation refers to tasks that rely on processing tactile information like images to solve manipulation tasks. Tasks could include key insertion which require tactile inputs to solve due to visual occlusions.

ManiSkill3 is the only simulator with vision tactile manipulation tasks that also have sim2real capabilities.

Implementation Details: The vision-tactile, sim2real, manipulation environments are ported over from the 2024 ManiSkill Vision-based-Tactile Manipulation Skill Learning Challenge (Chen et al., 2024).

A.10 Classic Control

Classic control is quite broad but in this context refers to fake robots tasked with achieving some stable pose or moving in a direction at desired speeds. Tasks include cartpole balancing, humanoid walking, hopper etc.

DM-Control (Tunyasuvunakool et al., 2020) has the most implemented control tasks and Isaac Lab has a few GPU parallelized variants. ManiSkill3 has GPU parallelized simulation+rendering variants of the most control tasks.

Implementation Details: ManiSkill3 uses its MJCF loader to load the MJCF robot definitions from the original DM-Control repository and tunes robot pd joint delta position controllers to align as closely as possible to the behavior seen in DM-Control.

A.11 Digital Twins

Digital twins have two variants included, environments for real2sim and environments for sim2real. The distinction here is real2sim environments simply need to be designed so that a model trained on a real world equivalent of the simulation environment achieves similar success rates when evaluated in simulation. Sim2real digital twins are environments designed so that models trained on simulation data can be more easily used for real world deployment.

For real2sim environments, ManiSkill3 ports over and GPU parallelizes some environments from SIMPLER (Li et al., 2024), which enables efficient evaluation of policies trained on real world data like the generalist RT-1 and Octo models. The primary tricks include green-screening a real world image and texture matching which have been copied over and parallelized. In depth documentation/tutorials are provided on how to build custom digital twins for real2sim. We ensured the GPU parallelized port of SIMPLER achieves similar results/behaviors as the original CPU simulated environments as shown in Figure 12.

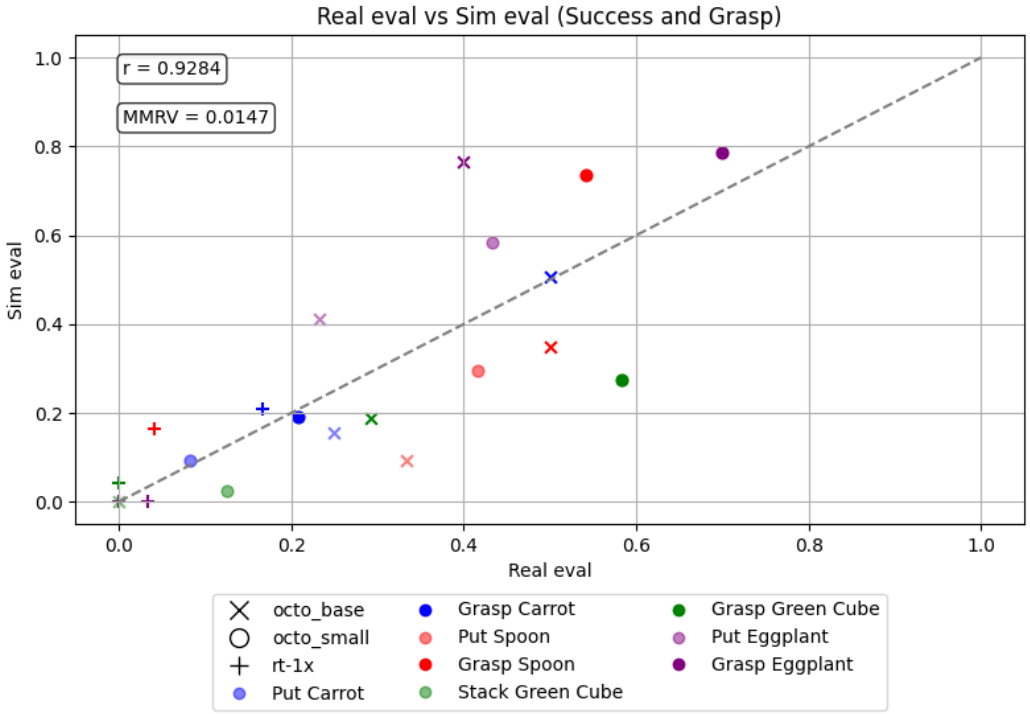

Figure 12: Evaluated success rates of generalist robotics models like Octo and RT-1X on 4 different tasks. The correlation and MMRV metrics are close to that of the original paper.

TRANSIC-Envs (Jiang et al., 2024) provide reproducible sim2real digital twin setups but rely on state estimation via ArUco markers for sim2real transfer and does not support visual feedback. Dextreme (Handa et al., 2023) provides a realistic in hand cube rotation environment but also does not support efficient visual feedback and relies on accurate state estimation. ManiSkill3 on the other hand has fast visual data generation that enable mimicing real-world cameras for training at scale.

A.12 Soft Body Manipulation Environments

Soft body manipulation refers to the manipulation of soft body objects that can deform and morph in shape. Tasks include excavating sand particles, pouring water etc.

ManiSkill3 soft body manipulation environments are the same as ManiSkill2 which uses 2-way coupled rigid-MPM simulation that enables rigid body objects to interact with soft body objects. We point readers to the ManiSkill2 paper for more details on soft body simulation.

B Simulation and Rendering Benchmarking

We carefully analyze the performance of ManiSkill3's parallel simulation and simulation+rendering performance compared to Isaac Lab. Currently we only have accurate results for a simple cartpole environment. We compare against the latest Isaac Lab v1.2.0 released on Sept. 20, 2024.

B.1 Setup

To keep things as fair as possible as both Isaac Lab and ManiSkill3 use PhysX, we ensure the following simulation configurations are the same:

- Simulation frequency: 120
- Control frequency: 60
- Solver Position Iterations: 4
- Solver Velocity Iterations: 0

We further try to ensure the objects in the scene are as similar as possible. Finally, reward/termination computations are explicitly removed to not factor in benchmark timings. Tests were conducted on a RTX 4090 GPU.

B.2 Simulation Only Benchmark Results

Testing on the cart pole task, we see that ManiSkill3 always runs a little faster and uses about 1.5x to 2x less GPU memory in Figure 13. We note that while ManiSkill3 appears faster here, one could make a many simulation specific optimizations (simplified collisions meshes, tuned physics solver configurations etc.) to increase simulation speed and trade off simulation accuracy. We specifically create documentation/tutorials on the various tricks/optimizations a user can perform to improve simulation speed and/or fidelity from the environment object/robot models to simulation configurations.

B.3 Simulation+Rendering Benchmark Results

We ablate on a number of aspects of visual data collection where we simulate and render an environment. We record the FPS of taking 1000 random actions in the environment and fetching the 1000 visual observations. Generally in the ablations we scale up one of environments, camera sizes, or number of cameras until the benchmark runs out of GPU memory.

Overall Isaac Lab shows about 2-3x higher GPU memory usage compared to ManiSkill3, with the minimum amount of GPU memory taken up by just environments with cameras enabled being 4.8GB compared to 1.7GB in ManiSkill3.

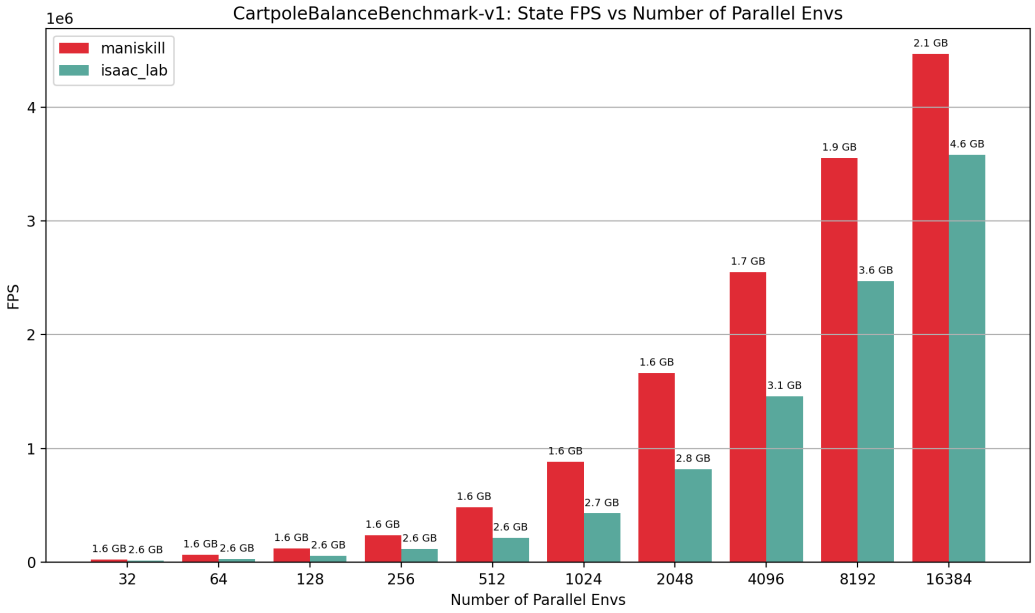

Figure 13: Simulation FPS against number of parallel environments without rendering. Annotated numbers on top of data points indicate the GPU memory usage.

B.3.1 Ablation on Realistic Camera Settings

We test on two realistic camera resolutions/setups based on the setup of the Open-X dataset (Collaboration et al., 2023) and the Droid dataset (Khazatsky et al., 2024). Many datasets in Open-X like the Google RT datasets have a single 640x480 resolution RGB camera, and the Droid dataset uses three 320x180 resolution RGB cameras. The results are shown in Figures 14 and 15. In both cases ManiSkill3 has about 2-3x better GPU memory efficiency and runs about 2x faster. Note that Isaac Lab's rendering output is a bit different from ManiSkill3, we share some qualitative examples in Appendix B.4. We further note that both ManiSkill3 and Isaac Lab support more photorealistic, high quality renders with full ray-tracing that is not parallelized. Example ManiSkill3 high-quality ray-traced renders can be seen in Figure 1.

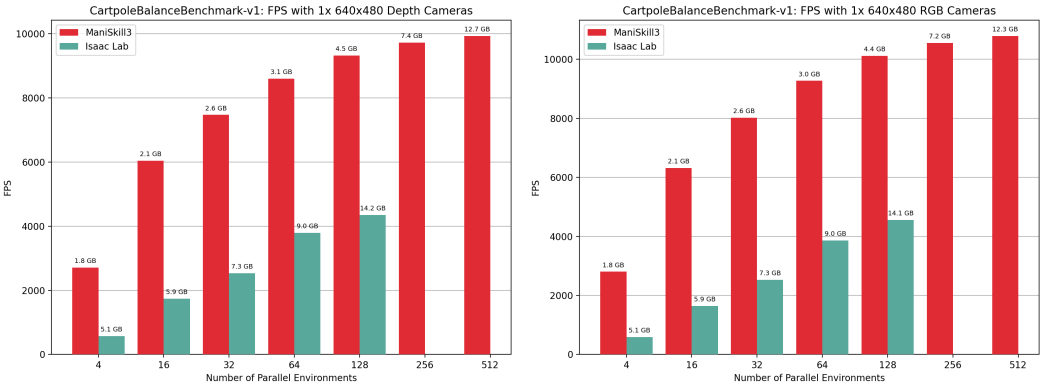

Figure 14: Simulation+Rendering of RGB or Depth FPS against number of parallel environments with 1x640x480 camera. Annotated numbers on top of data points indicate the GPU memory usage.

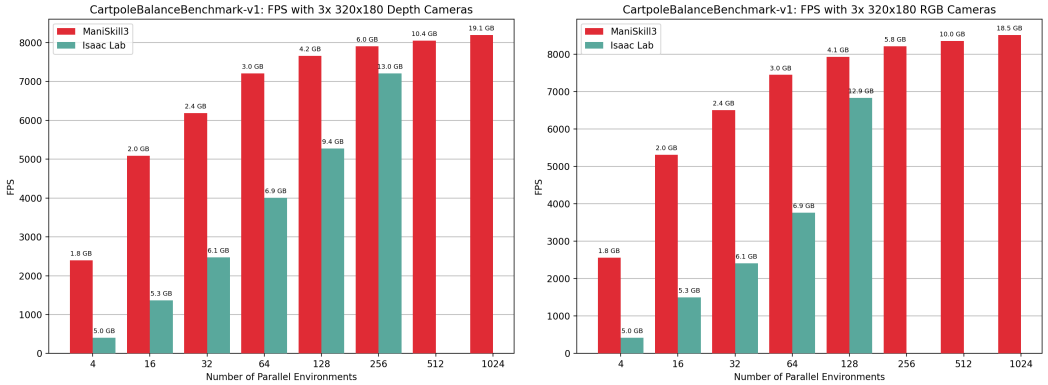

Figure 15: Simulation+Rendering of RGB or Depth FPS against number of parallel environments with 3x320x180 cameras. Annotated numbers on top of data points indicate the GPU memory usage.

B.3.2 Ablation on Camera Size

We ablate on square camera sizes from a large 512x512 resolution to a small 128x128 resolution shown in Figures 16 17 18. We observe that Isaac Lab runs at most about 1.2x faster when there are a high number of parallel environments with small camera resolutions compared to ManiSkill3. At larger camera resolutions ManiSkill3 outperforms up to 2x in speed and 3x in GPU memory usage. Notably, ManiSkill3 always outperforms about 2-4x in speed and 2-3x in GPU memory usage for smaller number of parallel environments. While Isaac Lab is fast at smaller resolutions, we note that some manipulations tasks are fairly impractical and impossible to solve at small resolutions. Moreover memory efficiency is absolutely critical for RL applications that can remain fast by keeping large replay buffers on the GPU instead of using CPU memory. For this reason all visual RL baselines in ManiSkill3 typically do not use more than 256 parallel environments as RL rollouts only run marginally faster with more environments beyond that point (for both Isaac Lab and ManiSkill3) but GPU memory use worsens a lot more.

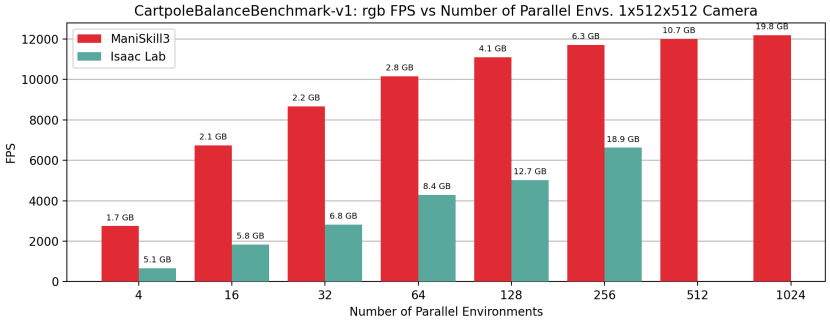

Figure 16: Simulation+Rendering (RGB) FPS against number of parallel environments 1x512x512 camera. Annotated numbers on top of data points indicate the GPU memory usage.

B.4 Qualitative Rendering Results

To the best of our capabilities we try to make the benchmarked environments look as similar as possible in terms of visuals. It is not possible to keep everything the same given fundamental differences in the parallel rendering system of Isaac Lab compared to ManiSkill3 or easily quantify the rendering differences. Section 3.2 details a little bit about the differences. Figure 4 in the main paper shows the cartpole RGB+Depth rendering results. Figures 19 and 20 shows RGB renders of a environment with a Franka Panda arm with different resolutions.

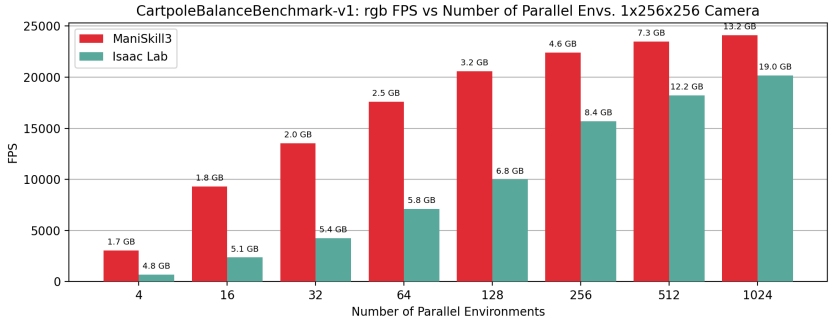

Figure 17: Simulation+Rendering (RGB) FPS against number of parallel environments 1x256x256 camera. Annotated numbers on top of data points indicate the GPU memory usage.

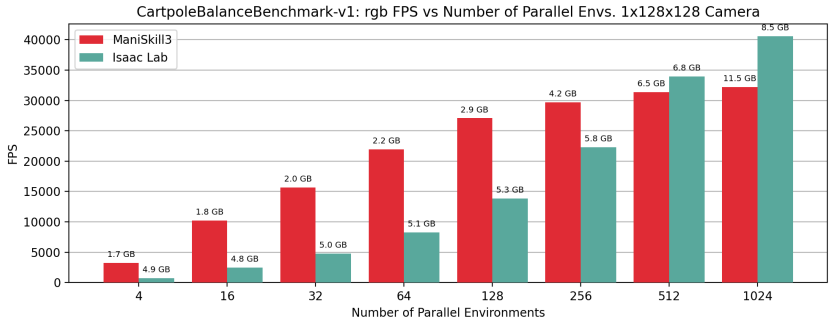

Figure 18: Simulation+Rendering (RGB) FPS against number of parallel environments 1x128x128 camera. Annotated numbers on top of data points indicate the GPU memory usage.

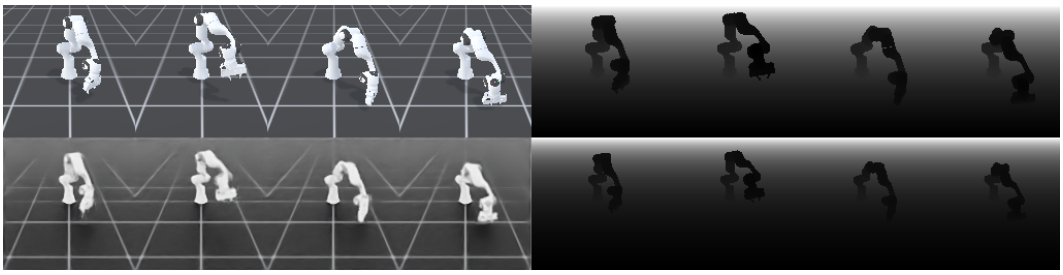

Figure 19: Comparison of ManiSkill3 (Top row) and Isaac Lab (Bottom Row) parallel rendering 128x128 RGB+Depth image outputs of the Franka Panda arm.

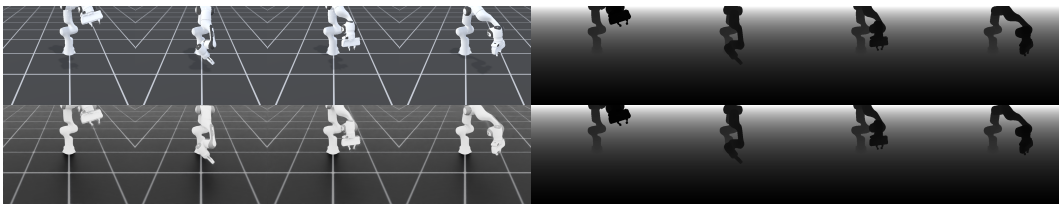

Figure 20: Comparison of ManiSkill3 (Top row) and Isaac Lab (Bottom row) parallel rendering 640x480 RGB+Depth image outputs of the Franka Panda arm.