ROBOVERSE: Towards a Unified Platform, Dataset and Benchmark for Scalable and Generalizable Robot Learning

Haoran Geng[1*], Feishi Wang[1,2,3*], Songlin Wei[2*], Yuyang Li[2*], Bangjun Wang[3*], Boshi An[2*], Charlie Tianyue Cheng[1*], Haozhe Lou[3], Peihao Li[1,4], Yen-Jen Wang[1], Yutong Liang[2], Dylan Goetting[1], Chaoyi Xu[2], Haozhe Chen[5], Yuxi Qian[6], Yiran Geng[2], Jiageng Mao[3], Weikang Wan[7], Mingtong Zhang[3], Jiangran Lyu[2], Siheng Zhao[3], Jiazhao Zhang[2], Jialiang Zhang[1,2], Chengyang Zhao[8], Haoran Lu[2], Yufei Ding[1,2], Ran Gong[9], Yuran Wang[2], Yuxuan Kuang[2,3], Ruihai Wu[2], Baoxiong Jia[9], Carlo Sferrazza[1], Hao Dong[2], Siyuan Huang[9], Koushil Sreenath[1], Yue Wang[3†], Jitendra Malik[1†], Pieter Abbeel[1†]

[1]UC Berkeley [2]PKU [3]USC [4]UMich [5]UIUC [6]Stanford [7]UCSD [8]CMU [9]BIGAI
* equal contribution † equal advising Correspondence to: Haoran Geng <ghr@berkeley.edu>

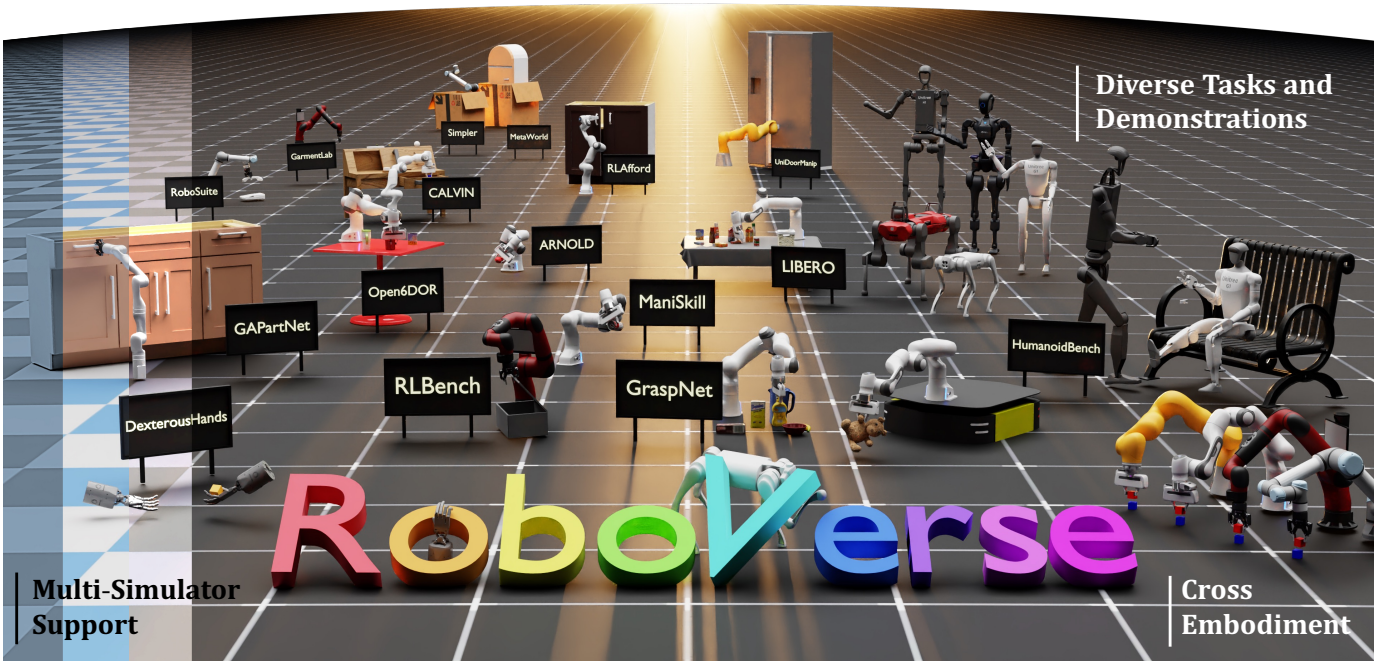


Fig. 1: ROBOVERSE comprises a scalable simulation platform, a large-scale synthetic dataset, and unified benchmarks. The simulation platform supports seamless integration of new tasks and demonstrations through unified protocols, ensuring flexibility and extensibility. The dataset includes over 1,000 diverse tasks and more than 10 million transitions, constructed through large-scale data migration, cross-embodiment transfer, and robust augmentation and randomization.

***Abstract*—Data scaling and standardized evaluation benchmarks have driven significant advances in natural language processing and computer vision. However, robotics faces unique challenges in scaling data and establishing reliable evaluation protocols. Collecting real-world robotic data is resource-intensive and inefficient, while benchmarking in real-world scenarios remains highly complex. Synthetic data and simulation offer promising alternatives, yet existing efforts often fall short in data quality, diversity, and benchmark standardization. To address these challenges, we introduce ROBOVERSE, a comprehensive framework comprising a *simulation platform*, a *synthetic dataset*, and *unified benchmarks*. Our simulation platform supports multiple simulators and robotic embodiments, enabling seamless transitions between different environments. The synthetic dataset, featuring high-fidelity physics and photorealistic rendering, is constructed through multiple approaches including migration from public datasets, policy rollout, and motion planning, *etc*. enhanced by data augmentation. Additionally, we propose unified benchmarks for imitation learning and reinforcement learning, enabling consistent evaluation across different levels of generalization. At the core of the *simulation platform* is METASIM, an infrastructure that abstracts diverse simulation environments into a universal interface. It restructures existing simulation environments into a simulator-agnostic configuration system, as well as an API aligning different simulator functionalities, such as launching simulation environments, loading assets with initial states, stepping the physics engine, *etc*. This abstraction ensures interoperability and extensibility. Comprehensive experiments demonstrate that ROBOVERSE enhances the performance of imitation learning, reinforcement learning, and world model**

learning, improving sim-to-real transfer. These results validate the reliability of our dataset and benchmarks, establishing RoboVerse as a robust solution for advancing simulation-assisted robot learning.

I. Introduction

Large-scale datasets, combined with well-established benchmarks, have fueled rapid advancements in natural language processing (NLP) [92, 5] and computer vision (CV) [23, 59, 57, 94, 66, 43] . Specifically, large-scale data provides ample training examples that bolster learning, while uniform benchmarks enable standardized evaluation and fair comparison across different methods. However, replicating these successes in robotics remains challenging due to the difficulty of collecting high-quality, diverse data and the lack of widely recognized evaluation protocols.

Real-world approaches [15, 54] to constructing datasets and benchmarks, though authentically reflecting the complexities of operational environments, face significant practical constraints. First, collecting demonstrations is time-consuming and resource-intensive, and the resulting data is often hardware-dependent or modality-specific, limiting its adaptability to new scenarios. Additionally, establishing standardized and widely applicable benchmarks is inherently challenging since reproducing identical conditions for fair comparisons is nearly impossible. For instance, object placements can vary across rollouts, ambient lighting fluctuates under natural sunlight, and background environments may change. Consequently, scaling real-world datasets, evaluating policies, and iterating development in real-world scenarios remain cost-prohibitive and difficult to standardize.

Simulators, on the other hand, present a promising alternative for large-scale dataset and benchmark construction. By providing efficient computation, synthetic assets, and omniscient information in reproducible settings, they enable cost-effective dataset construction and consistent performance evaluation. Recent works, exemplified by [134, 50, 10, 33, 97, 123, 69], have demonstrated the potential of simulation-based methods in various robotic tasks. Despite these advantages, several challenges impede the broader adoption of synthetic datasets and benchmarks. First, utilizing simulators often demands considerable expertise due to both the complexity of simulator design and the relative immaturity of many platforms, which complicates the data construction process. Second, simulators vary widely in their internal architectures and external interfaces, making it laborious to transfer data and models or adapt workflows from one to another. Consequently, reusing existing synthetic datasets and benchmarks is difficult, resulting in a fragmented ecosystem that further hinders convenient construction and effective use of large-scale data in simulation environments.

To fully harness the potential of simulation in robotics, we introduce RoboVerse, a scalable simulation platform that unifies existing simulators under a standardized format and a single infrastructure, a large-scale synthetic dataset, and unified benchmarks. To achieve this, we first propose MetaSim, the core infrastructure of the RoboVerse. Through careful design, MetaSim establishes a universal configuration system for agents, objects, sensors, tasks, and physics parameters while exposing a simulator-agnostic interface for simulation setup and control. This architecture enables seamless integration of tasks, assets and robot trajectories from diverse simulation environments with minimal adaptation effort. MetaSim provides three key capabilities: (1) *Cross-Simulator Integration*: Enables seamless switching between different simulators, fostering unified benchmarking and facilitating the transfer of environments and demonstrations across platforms. (2) *Hybrid Simulation*: Combines the strengths of multiple simulators—such as pairing advanced physics engines with superior renderers—to generate scalable and high-quality synthetic data. (3) *Cross-Embodiment Transfer*: Allows the retargeting of trajectories across various robot arms with parallel grippers, maximizing dataset reuse from heterogeneous sources.

MetaSim enables RoboVerse to systematically enhance the workflow for building and scaling simulation environments and datasets. Our method features:

- *Scalable and Diverse Data Generation*: By aligning multiple benchmarks and task trajectories and leveraging a robust multi-source integration and data filtering pipeline, we generate large-scale, high-quality datasets. Additionally, our data randomization and augmentation pipeline enhances data diversity and volume, further enriching the dataset for comprehensive model training;
- *Realistic Simulation and Rendering*: With MetaSim's hybrid simulation capability, we enable the fusion of advanced physics engines and rendering systems across multiple simulators and renderers. Combined with carefully curated scenes, materials, and lighting assets, RoboVerse enhances realism in physical interactions and sensory observations;
- *Unified Benchmarking and Evaluation*: We unify widely used benchmarks into a cohesive system, streamlining algorithm development and performance comparison within a structured evaluation framework. Additionally, we introduce a standardized benchmarking protocol to assess varying levels of generalization and sim-to-real transferability.
- *Highly Extensibility and Scalability:* The aligned APIs and infrastructure streamline development and enable efficient algorithm integration, testing, and deployment across diverse simulation environments. Additionally, we develop real-to-sim frameworks, multiple teleoperation methods, and AI-generative systems for scalable task and data creation.

Leveraging these workflows in RoboVerse, we construct the largest and most diverse high-quality synthetic dataset and benchmark to date, all in a unified format. This dataset includes $\sim$500k unique, high-fidelity trajectories covering 276 task categories and $\sim$5.5k assets. Additionally, we generate over 50 million high-quality state transitions to support policy learning.

Beyond dataset and benchmark construction, we explore the potential of ROBOVERSE through extensive experiments on imitation learning (Sec. VI-B), reinforcement learning (Sec. VI-C), and world model learning (Sec. VI-E). Our results demonstrate that ROBOVERSE enables reliable policy learning and evaluation, supports strong sim-to-sim and (Sec. VI-G) sim-to-real transfer (Sec. VI-F) via high-fidelity physics and rendering, and facilitates efficient data expansion through teleoperation (Sec. **??**), trajectory augmentation (Sec. IV-D1), domain randomization (Sec. IV-D2) and generative models (Sec. IV-C). These findings highlight the framework's robustness, scalability, and real-world applicability.

II. RELATED WORK

A. Robotics Simulators

Advancements in computer graphics have contributed to the development of high-fidelity simulators, which are widely used in robotics research and development. CoppeliaSim [96], Bullet [16], and MuJoCo [110] provide accurate physics simulations and are extensively utilized in applications such as reinforcement learning and robotic benchmarking [3, 125, 86, 14]. More simulators have been developed to fully exploit parallelism for better efficiency. IsaacGym [71], IsaacSim [84], SAPIEN [37, 108], MuJoCo MJX [110, 131], and Genesis [2] utilize GPU power for enhanced performance, enabling large-scale reinforcement learning and efficient data collection, significantly improving training speed and scalability. Some simulators focus on bridging the simulation-reality gap (Sim-to-Real Gap), incorporating technologies including ray-tracing and customized renderers for photo-realistic rendering [84, 108]. Furthermore, IsaacSim [84] and Genesis [2] offer high-fidelity soft-body and liquid simulation, expanding the scope of realistic robotic interactions. ROBOVERSE proposes a unified platform that supports multiple simulators, facilitating seamless transitions between them and enabling hybrid integration to utilize the strengths of each simulator.

B. Large-Scale Robotics Dataset

The scarcity of large-scale, high-quality, and diverse datasets in the robotics community has long been recognized. Several works have shown the possibility of collecting demonstration data directly on real robots. RoboNet [20] is a large-scale manipulation dataset containing roughly 162k trajectories from multiple robot platforms. DROID [54] has collected over 76k contact-rich robotic manipulation demonstrations across 86 tasks. RH20T [28] proposed a dataset with over 100k demonstrations and 147 tasks. At the same time, RT-1 [4] set the record further to 130k demonstrations on over 700 tasks. Recently, Open X-embodiment [15] has demonstrated a promising approach to unite the community's efforts, collecting over 1M trajectories on 160,266 tasks with 22 different embodiments. At this stage, real-world datasets became difficult to scale up due to the proportional effort and cost required to collect more demonstrative trajectories.

Simulation-based data collection provides a promising solution to the high cost and inefficiencies of real-world datasets.

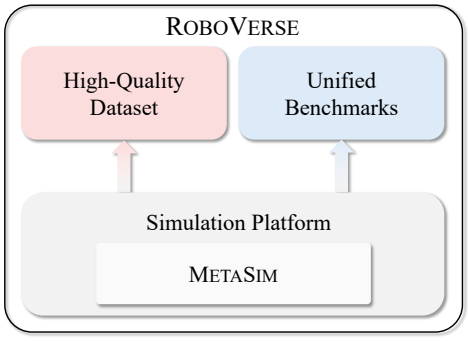


Fig. 2: ROBOVERSE consists of a simulation platform, a large-scale, high-quality dataset, and unified benchmarks. At the core of the simulation platform is METASIM, the infrastructure of ROBOVERSE. Powered by METASIM, the simulation platform facilitates dataset creation and benchmark construction.

Hussing et al. [46] proposed a dataset containing 256M transitions on 256 tasks for offline compositional reinforcement learning. RoboCasa [81] introduced a dataset of 100 tasks and over 100k trajectories for generalist robots. DexGraspNet-2.0 [133] has collected over 400M demonstrations for dexterous grasping. Despite these efforts, synthetic datasets often exist in disparate simulators, leading to a fragmented ecosystem with limited diversity and quality. Moreover, simulation-based data often fails to capture complex physics and diverse task variations found in the real world [62, 26], potentially causing overfitting to specific simulators and hampering generalization to real-world scenarios.

ROBOVERSE provides a unified solution for large-scale, high-quality, and diverse synthetic data. It enables agents to train on a large set of environments and simulators to reduce overfitting, thereby improving the robustness of the learned policies.

C. Benchmarking in Robotics

Benchmarking remains a critical yet highly challenging problem in the robotics community. Compared to supervised learning tasks, it is relatively difficult to evaluate the performance of a robotics model. MetaWorld [130] is an early attempt in multi-task benchmarking. This is followed by RLBench [48], Behavior-1k [61], Habitat [107], and ManiSkill [80, 37, 108, 102], covering a large variety of robotic tasks. Grutopia [115] and InfiniteWorld [95] make a leap toward general-purpose robot benchmarking.

Despite significant efforts dedicated to these benchmarks, it is not guaranteed that the results are reproducible across different benchmarks. The uncertainty comes from multiple aspects including simulation accuracy, rendering style and asset properties [62, 26]. To address these challenges, ROBOVERSE enables researchers to evaluate their policies across multiple benchmarks and simulators seamlessly, without familiarizing themselves with each one individually.

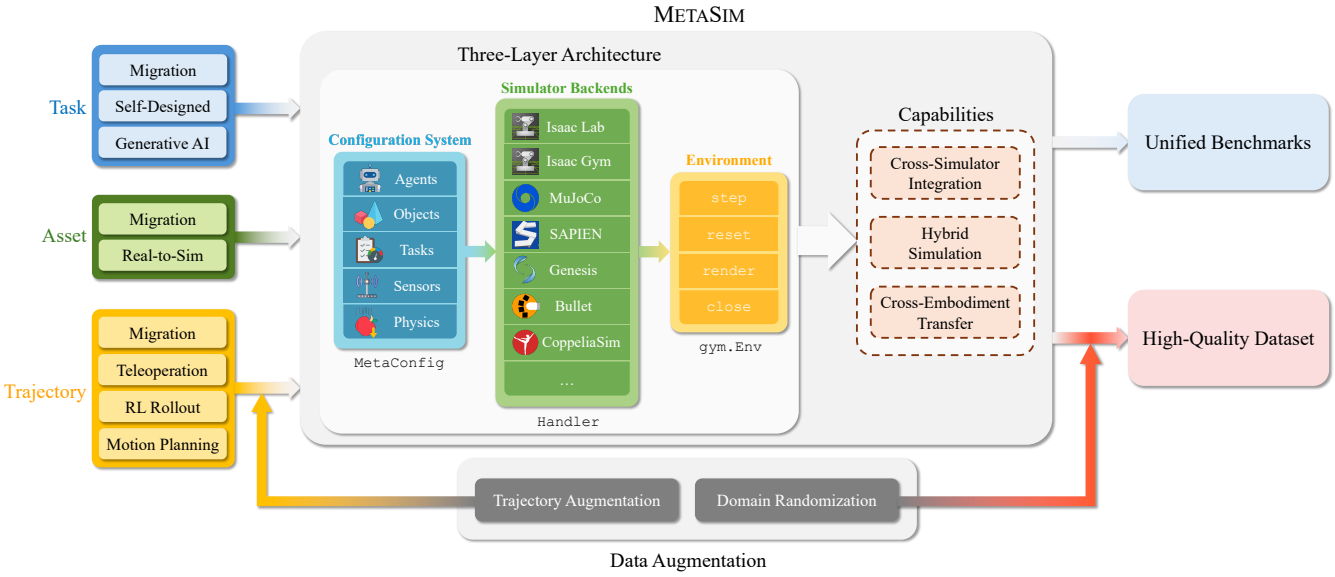


Fig. 3: METASIM provides a universal configuration system, aligned simulator backends, and a Gym [111] environment wrapper. This three-layer architecture abstracts simulation environments into simulator-agnostic specifications and aligns simulator backends, enabling three key capabilities: cross-simulator integration, hybrid simulation and cross-embodiment transfer. Based on METASIM, we build a pipeline to collect tasks, assets and trajectories from diverse public sources in a unified format, employ data augmentation methods, and ultimately generate a large-scale high-quality dataset along with unified benchmarks. This data pipeline forms the foundation of ROBOVERSE, facilitating the generation of large-scale datasets and construction of unified benchmarks.

III. INFRASTRUCTURE: METASIM

A. METASIM *Overview*

We present METASIM, a high-level interface above specific simulation environment implementations. It is also the core infrastructure of ROBOVERSE. As illustrated in Fig. 2, METASIM empowers the ROBOVERSE simulation platform, allowing for the generation of a large-scale high-quality dataset, as well as the construction of a unified benchmark.

B. METASIM *Implementation*

As illustrated in Fig. 3, METASIM employs a three-layer architecture including a universal configuration system, a simulator-agnostic interface, and a user-friendly environment wrapper. The universal configuration system unifies specifications for a simulation scenario and ensures consistent format across simulators. The simulator-agnostic interface interprets these specifications, translates them into simulator-specific commands, and therefore aligns different simulator backends. In addition, the environment wrappers encapsulate the simulator-agnostic interface into a standarized learning environment, such as a Gym [111] environment. We describe each layer with more details in the following sections.

1) Universal Configuration System: A typical simulation environment comprises agents, objects, tasks, sensors, and physics parameters. They collectively define who performs the actions (agents), what the environment looks like (objects), what the agents should do (tasks, including instructions, success metrics, and rewards), how the environment is perceived and measured (sensors), and the governing physical laws (physics parameters). Ideally, these components should be simulator-agnostic, requiring a unified standard of simulation scenarios. Such a standard would enable researchers to work across different simulators seamlessly and integrate existing efforts from the community through cross-simulation.

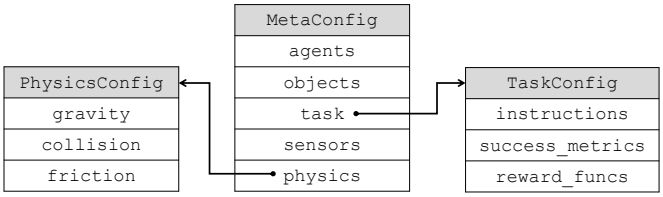


Fig. 4: The `MetaConfig` is a nested dataclass that abstracts the core components in any simulation environment in a simulator-agnostic way.

Based on such a principle, we design a configuration system, `MetaConfig`, to abstract simulation scenarios in a simulator-agnostic way. As illustrated in Fig. 4, `MetaConfig` is a nested class that contains the above-mentioned core components. It can be interpreted by different simulator backends to build the corresponding simulation. Additionally, `MetaConfig` supports optional simulator-specific hyperparameters (*e.g.*, solver type), allowing fully leveraging the unique features of different simulators through customization.

```
class Env:
    def __init__(self, handler):
        self.handler = handler
        handler.launch()

    def reset(self):
        handler.set_states()
        states = handler.get_states()
        return get_observation(states), \
               handler.get_extra()

    def step(self, action):
        handler.set_states(action=action)
        handler.step()
        states = handler.get_states()
        return get_observation(states), \
               get_reward(states), \
               get_success(states) \
               get_termination(states), \
               get_time_out(states), \
               handler.get_extra()

    def render(self):
        return handler.render()

    def close(self):
        handler.close()
```

Code 1: Pseudocode for `gym.Env` implementation. Each method of `gym.Env` is implemented by calling the corresponding methods of the `Handler` class.

2) Aligned Simulator Backends: Different simulators have their own implementations and specializations. However, routine operations – such as initializing a scene, loading objects, stepping the physics engine, retrieving observations, time management, and determining success states – tend to follow similar patterns. To standardize these shared operations, we create a unified interface through a `Handler` class. Each simulator has its own handler instance implementing this interface. The handler class implements the common methods including `launch()`, `get_states()`, and `set_states()`, *etc*., spanning the whole lifecycle of simulating a task. The usage of the APIs is illustrated in Code 1. More information is provided in the supplementary materials.

3) User-Friendly Environment Wrapper: The Gym API [111] is a widely adopted paradigm in reinforcement learning and robotics, in which the `gym.Env` class is fundamental to building learning environments. We define a wrapper to easily transform a `Handler` into an environment equipped with Gym APIs (`step()`, `reset()`, `render()`, and `close()`). As shown in Code 1, these methods are implemented by leveraging the underlying `Handler` methods.

C. MetaSim Capabilities

MetaSim offers the following three key capabilities.

1) Cross-Simulator Integration: Seamlessly switching between different simulators, allowing tasks and trajectories from one simulator to be utilized in other simulators. This capability enables efficient task and trajectory integration, unified benchmark construction, and sim-to-sim transfer for reinforcement learning training. For example, tasks from MetaWorld [130] can be used by Isaacgym [71] for fast parallel training, after which the generated trajectories can be deployed in IsaacSim [84] for rendering.

2) Hybrid Simulation: MetaSim supports combining the physics engine of one simulator and the renderer of another simulator at the same time, allowing users to benefit from advantages owned by different simulators. Specifically, using a single command, one could launch a simulator with a powerful renderer (*e.g*., IsaacSim [84]) with a simulator that has an accurate physics engine (*e.g*., MuJoCo [110]) to form an even more powerful simulation, enabling high-quality data generation.

3) Cross-Embodiment Transfer: Reusing the trajectories across different gripper-based robot morphologies by retargeting the end-effector pose, which allows the integration of data collected from diverse robots into a unified format.

IV. RoboVerse Dataset

A. Dataset Overview

On top of MetaSim, we generate large-scale high quality dataset by incorporating multiple data collection methods. Overall, there are three key data types to collect: tasks, assets, and robot trajectories. The main source of these data is migration from existing simulation environments. Beyond migration, we explore various methods to collect these data, such as using large language models to generate new tasks, leveraging the RealSsim toolset [67] to reconstruct assets from the real world, using teleoperation to collect new trajectories, *etc*. Additionally, we leverage data augmentation methods for both trajectories and visual observations. Finally, we report the statistics for current progress of data migration in RoboVerse.

B. Tasks, Assets and Trajectories Collection: Migration

Leveraging the RoboVerse format and infrastructure, we seamlessly integrate a wide range of benchmarks and datasets into our system with a unified format and clean codebase. We apply the following approaches to collect tasks and demonstrations.

- **Direct Migration from Other Simulation Environments** Some benchmarks provide essential components integration into RoboVerse. We define environment configurations for task initialization and evaluation, then convert trajectory data and asset formats for seamless compatibility. Notably, RoboVerse streamlines this migration process by first aligning formats in the original simulator and automatically ensuring compatibility across all simulators.
- **Motion Planning and RL Rollout** When benchmarks provide only partial manipulation data, such as keypoint trajectories or grasping poses, we use motion planning to generate complete trajectories. If no explicit manipulation data is available but pre-existing policies or reinforcement learning frameworks exist, we either utilize these policies or train new ones to collect demonstration data through rollouts. To ensure high data quality and consistency with

our system standards, we carefully adapt the success checker and rigorously filter both planned and collected trajectories.

With the techniques mentioned above, we migrated multiple existing manipulation datasets into ROBOVERSE. Currently, we support ManiSkill [80, 37, 108], RLBench [48], CALVIN [78], MetaWorld [130], RoboSuite [141], MimicGen [75], GAPartNet [34], Open6DOR [24], ARNOLD [36], LIBERO [64], Simpler [62], GraspNet [27], GarmentLab [68], and UniDoorManip [63].

We also integrated datasets from a wider range of embodiments, including dexterous hands, quadrupeds, and humanoids, covering tasks such as dexterous manipulation, locomotion, navigation, and whole-body control. Currently, we have migrated VLN-CE R2R [58] and RxR [60] for navigation, as well as HumanoidBench [101] and Humanoid-X [76] for locomotion and whole-body control.

RoboVerse simplifies and standardizes the migration process, and we will continue to maintain and expand it.

C. Tasks, Assets and Trajectories Collection: Teleoperation and Generation

- **Teleoperation System for Trajectory Collection** . As shown in Fig. 5, ROBOVERSE integrates teleoperation systems within the METASIM infrastructure, offering a flexible and efficient solution for high-quality data collection. It supports various robotic systems, including arms, dexterous hands [87], and bimanual setups, enabling seamless teleoperation across different simulators. To mitigate the high cost and complexity of professional equipment, we introduce an interactive motion control system utilizing accessible devices such as keyboards, joysticks, mobile apps (we developed a new app for Android and iOS to control robotic arms; see supplementary materials for more details.), motion capture (Mocap) [113], and VR systems [12, 91]. These devices' integrated sensors capture motion data, allowing natural, gesture-based control along with real-time, high-frequency communication for precise, low-cost remote operation. Further details are provided in the supplementary materials.
- **AI-Assisted Task Generation**. Leveraging the generalization capability of large generative models, AI-assisted task generation provides a mechanism to diversify task varieties and scenario distribution. By learning from example placements, it acquires a sense of spatial and semantic constraints [1] (*e.g.* by demonstrating specific constraints, it can learn to spread out objects to avoid potential overlap etc.). It can arrange objects originally from different benchmarks into a physically plausible scenes based on METASIM, as shown in Fig. 6. Incorporating randomization in robot and object selection [52] with their initial poses, large generative models can generate various initial states. The system can automatically output all the required configuration files in unified format for instant visualization and user-friendly editing. After task generation, we will process a two-step filtering to avoid errors and hallucinations: (1) *Format Validation*: Tasks that fail to meet ROBOVERSE format standards are discarded. (2) *Feasibility Check*: Since trajectory data is collected via human teleoperation, tasks deemed unreasonable by the teleoperator are removed. By unleashing the extrapolative and few-shot learning abilities of large generative models, we integrate assets under a uniform schema automatically, driving task generation that spans multiple simulators and benchmarks.
- **Real-to-Sim for Asset Construction**. Video-based reconstruction proves to be a valuable source for data and asset creation by leveraging Real-to-Sim techniques. Our approach integrates multiple reconstruction pipelines to extract high-fidelity assets from video data. First, we initialize the structure using Colmap [98, 99] and employ Gaussian Splatting [53] for high-quality rendering. Next, we infer physical properties by feeding both semantic and original images into a Vision-Language Model (VLM) [139]. For geometry reconstruction, we estimate surface normals from video [128], apply surfel splatting [45], and utilize TSDF-based methods with dynamic filtering to reconstruct detailed meshes [127]. By leveraging semantic masks [94], we selectively extract components from both Gaussian and mesh representations. To further enhance realism, we infer and learn object kinematics directly from video [65], ensuring accurate motion representations. Finally, we formulate URDF models by refining key attributes such as coordinate frames, orientation, axis alignment, scale, relative 6-DoF poses, and PD control parameters [67]. This pipeline effectively bridges the gap between real-world video data and simulation-ready assets, enhancing robotic learning and simulation fidelity. We also present comparative experiments in the supplementary materials, demonstrating that our methods significantly enhance real-world policy performance.

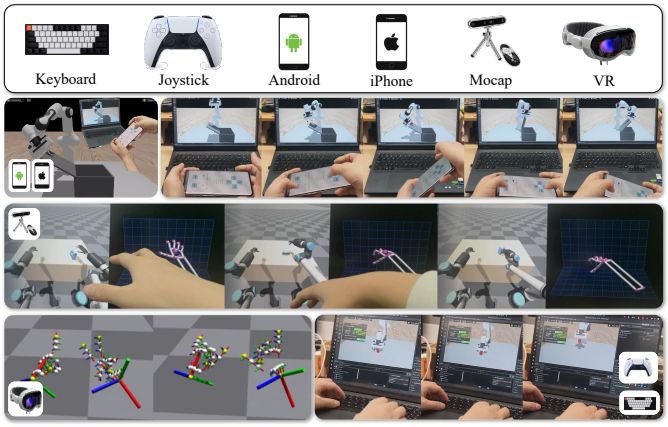


Fig. 5: **Teleoperation System.** ROBOVERSE supports various user-friendly teleoperation approaches. Currently, it enables teleoperation via a phone app (second row), motion capture (middle), VR devices (bottom left), as well as keyboard and joystick (bottom right). These methods allow control of robotic arms, dexterous hands, and bimanual systems across different simulators.

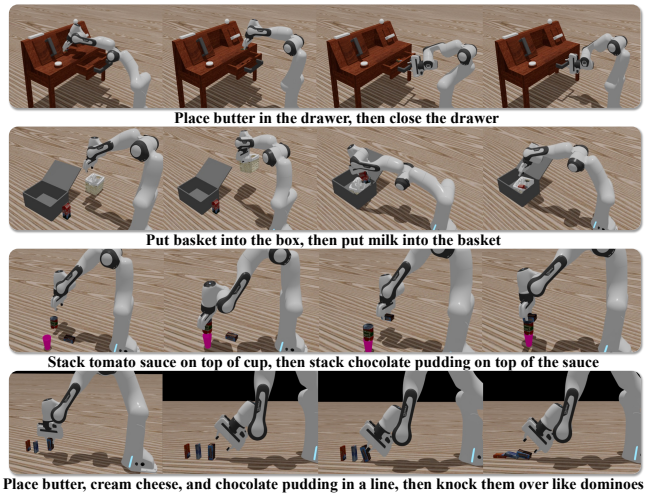


Fig. 6: **AI-Assisted Task Generation.** RoboVerse supports an AI-assisted task generation framework that leverages large generative models' extrapolation capabilities to generate non-trivial and semantically rich tasks. Combined with our teleoperation system, it enables the generation of diverse and high-quality data.

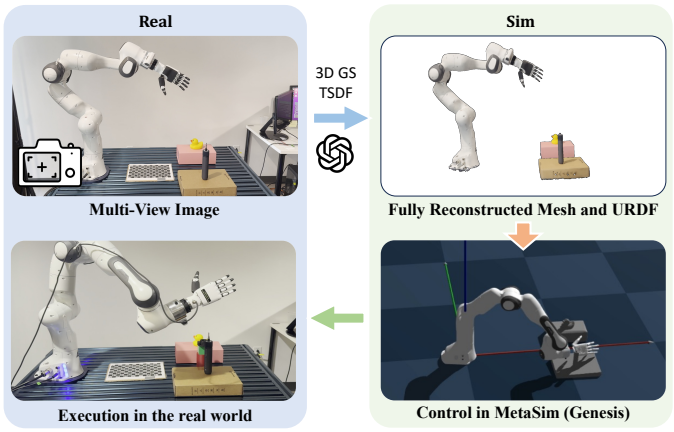


Fig. 7: **Real-to-Sim Tools.** We use a mobile device to capture multi-view images, reconstruct a high-quality mesh, build a URDF using VLM, and then perform actions in both RoboVerse and the real world.

D. Data Augmentation

1) Trajectory Augmentation: With the unified simulation interface and data format, RoboVerse enables significantly more efficient data augmentation and supports advanced augmentation techniques. Beyond the visual randomization detailed in Benchmark Protocol [8], we also provide robust trajectory space augmentation. We offer an API to generate large-scale robot trajectory datasets from a limited number of source demonstrations. Following the MimicGen [75] framework, for most tasks, we can decompose them into a sequence of object-centric subtasks $(S_1(o_{S_1}), S_2(o_{S_2}), \ldots, S_M(o_{S_M}))$, where the robot's trajectory within each subtask $S_i(o_{S_i})$ is relative to a single object's coordinate frame ($o_{S_i} \in \mathcal{O}$, $\mathcal{O}$ is the set of objects in the task $\mathcal{M}$). Additionally, we assume that the sequence of subtasks in each task is predefined. By leveraging this minimal human annotation regarding the order of subtasks, we can efficiently divide each source demo into contiguous object-centric manipulation segments $\{\tau_i\}_{i=1}^{M}$ (each of which corresponds to a subtask $S_i(o_i)$) using a simulator, and then generate extensive trajectory datasets for various task variants (in our case: variations in the initial and goal state distributions of objects (D) and robots (R)) using MimicGen [75]. This approach has been shown to significantly benefit generalization in imitation learning [75, 50, 116, 31, 81], particularly in scenarios where the number of source demonstrations is limited. For further details, please refer to the supplementary materials.

2) Domain Randomization: We implement domain randomization in the IsaacLab [84] handler of `MetaSim`. This involves four types of randomization:

- **Table, Ground, and Wall.** Walls (and ceilings) can be added for tasks that lack a predefined scene. Customizable tables can also be included for tasks that are performed on tabletops. The visual materials for these elements are randomly selected from a curated subset of ARNOLD [36] and vMaterials [83]. The table has ∼300 material options, while the wall and ground each have around ∼150 material options.
- **Lighting Condition.** Two types of lighting scenarios can be specified: distant light and cylinder light arrays. For distant light, the light's polar angles are randomized. For cylinder light, a random $n \times m$ matrix of cylinder lights with random size is added at a fixed height above the agents. In both scenarios, the intensity and color temperature of the lights are randomized within a reasonable range.
- **Camera Poses.** We carefully select 59 candidate camera poses, with the majority positioned to face the robot directly and a smaller subset placed at side-facing angles.
- **Reflection Properties.** The roughness, specular, and metallic properties of each surface are randomized within reasonable ranges.

These randomization options can be freely combined. For example, a scene can include a customized table, walls with a ceiling, and a set of cylinder lights to simulate an indoor environment. For details, please refer to the supplementary materials.

E. RoboVerse Dataset

1) Dataset Statistics:

a) Manipulation Dataset: We migrate diverse manipulation datasets from existing source benchmarks [80, 37, 108, 48, 78, 130, 141, 75, 34, 24, 36, 64, 62, 35, 27, 68, 63, 18] into RoboVerse. The number of task categories, trajectories and assets contributed by each source benchmarks is summarized in Tab. I. In total, this migration results in 276 task categories, 510.5k trajectories, and 5.5k assets. Representitive tasks with rich domain randomization are shown in Fig. 8.

b) Navigation Dataset: We migrate vision-and-language navigation (VLN) tasks into RoboVerse. Note that there exists various VLN tasks with different settings; here, we particularly

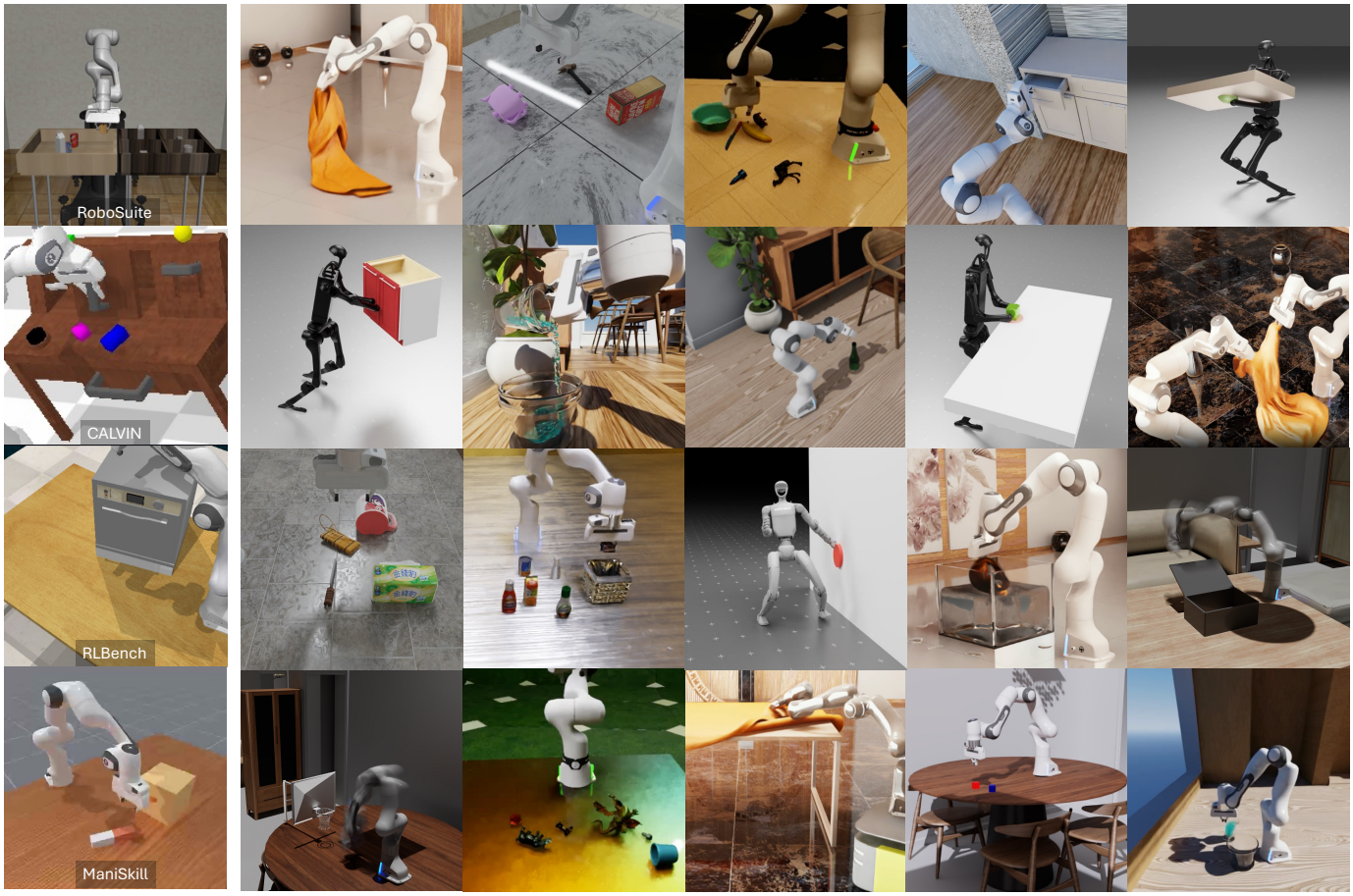


Fig. 8: **Dataset Comparison and Gallery**. Left: other representative synthetic robotics datasets. Right: the RoboVerse dataset.

Source Benchmark	Source Simulator	# Task Categories	# Trajectories	# Assets
ManiSkill [80, 37, 108]	SAPIEN	6	19k	1.7k
RLBench [48]	CoppeliaSim	80	150k	100
CALVIN [78]	Pybullet	7	20k	7
MetaWorld [130]	MuJoCo	5	5k	6
RoboSuite [141]&MimicGen [75]	MuJoCo	6	6k	12
GAPartNet [34]	IsaacGym	4	4k	151
Open6DOR [24]	IsaacGym	69	10k	207
ARNOLD [36]	IsaacSim	6	3k	30
LIBERO [64]	MuJoCo	10	15k	15
Simpler [62]	SAPIEN	6	30k	52
RLAfford [35]	IsaacGym	4	40k	40
GraspNet [27]	-	58	200k	42
GarmentLab [68]	IsaacSim	6	6k	3k
UniDoorManip [63]	IsaacGym	7	1k	140
GAPartManip [18]	IsaacSim	2	1.5k	42
Total	-	276	510.5k	5.5k

TABLE I: Migration progress statistics for manipulation tasks in RoboVerse

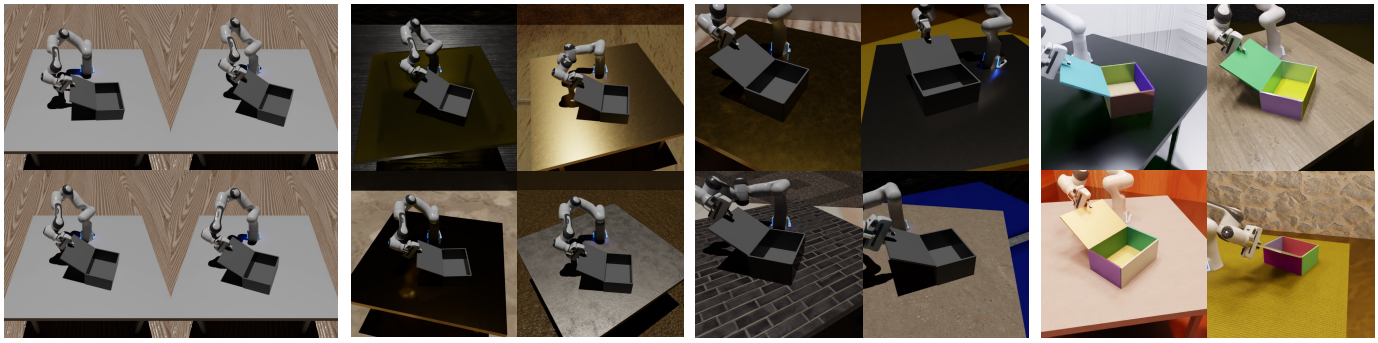
focus on VLN in continuous environments (VLN-CE) [58], as it more closely resembles real-world scenarios [11, 135, 136]. Specifically, we construct our dataset based on RoboVerse by integrating MatterPort 3D scenes [9] (90 scenes) and off-the-shelf instructions from R2R [58] (10k episodes) and RxR [60] (20k episodes). We provide two types of mobile embodiments, including the Unitree Dog (a legged robot) and the JetBot (a wheeled robot), which support different control policies. A detailed elaboration on the navigation dataset is provided in the supplementary materials.

c) Humanoid Dataset: We migrate HumanoidBench [101] tasks for reinforcement learning benchmarks and integrate tasks, policies, and data samples from Humanoid-X [76]. Additionally, we re-implement the UH-1 inference pipeline within our framework. The pretrained policy successfully enables humanoid robots to follow demonstrated poses while maintaining stable locomotion across multiple simulators based on RoboVerse.

V. RoboVerse Benchmark

A. Benchmark Overview

With the collected tasks, assets, and trajectories, RoboVerse establishes standardized benchmarks for robot learning, including both imitation learning and reinforcement learning. We define a unified training and evaluation protocol within the RoboVerse platform and implement standardized baselines and learning frameworks for benchmarking. Specifically, for imitation learning, we introduce different levels of generalization benchmarks to assess the generalization capability of models.



(a) Level 0 (b) Level 1 (c) Level 2 (d) Level 3

Fig. 9: **Benchmark Protocol:** We define a four-level generalization benchmarking protocol, allocating 90% of the data for training and 10% for generalization evaluation. From left to right, Levels 0 to 3 corresponds to task space generalization, environment radomization, camera randomization, lighting and reflection randomization, respectively.

B. Imitation Learning Benchmark

For each imitation learning benchmark, we establish a standardized evaluation framework with a fixed set of demonstrations and a controlled evaluation environment. Policies must be trained exclusively on the provided training data and assessed within this environment to ensure fair comparison. To rigorously test generalization capability, we curate training data from specific domains and evaluate policies on unseen samples, challenging their adaptability to novel scenarios. We systematically categorize visual generalization factors into multiple levels, including task space generalization, environment setup generalization, camera setting generalization, and lighting and reflection generalization. Each level introduces controlled variations to assess a policy's adaptability and robustness in increasingly diverse and challenging conditions.

a) Level 0: Task Space Generalization: We establish a controlled evaluation by standardizing the environment with consistent camera, materials, lighting, and other parameters. The task space, including object initialization and instructions, is split into 90% training and 10% validation to assess generalization within a fixed setting, as shown in Fig. 9 (a).

b) Level 1: Environment Randomization: Building on the standardized setup, we introduce scene randomization while keeping the camera, materials, and lighting fixed [77]. By varying house, table, and ground configurations, we create diverse visual inputs to test robustness against environmental changes [51]. A fixed set of predefined randomized scenes ensures structured evaluation, as shown in Fig. 9 (b).

c) Level 2: Camera Randomization: To assess generalization across camera variations, we introduce different viewing heights and angles using carefully annotated, realistic camera poses. Following the 90/10 training/testing split, we ensure consistent and rigorous evaluation, as illustrated in Fig. 9 (c).

d) Level 3: Lighting and Reflection Randomization: Real-world environments involve diverse materials and lighting conditions [112]. To simulate these challenges, we randomize lighting and reflections, curating realistic object materials and illumination setups [19]. This enhances robustness testing under varying conditions, as shown in Fig. 9 (d).

C. Reinforcement Learning Benchmark

In addition to imitation learning, RoboVerse offers a comprehensive reinforcement learning (RL) benchmark designed to accommodate a diverse range of tasks, robot embodiments, and simulation backends. Specifically, we integrate the PPO [100] algorithm from both STABLE-BASELINES3 [93] and RSL_RL [97] into our METASIM interface, enabling straightforward task definition, seamless environment switching, and standardized performance logging.

Building upon this infrastructure, we have successfully ported multiple humanoid control tasks from the HumanoidBench [101] benchmark into RoboVerse. Through our adapted interface for RSL_RL, we have efficiently extended framework compatibility to support the TD-MPC2 [41, 42] algorithm from the original benchmark while preserving implementation fidelity.

VI. EXPERIMENTAL RESULTS

A. Overview

We conduct extensive experiments to validate the effectiveness and practicality of ROBOVERSE. First, we evaluate baselines on representative tasks from various benchmark sources to ensure the reliability of the collected datasets and established benchmarks. This includes assessments of both imitation learning baselines VI-B and reinforcement learning baselines VI-C.

Then we further demonstrate the strength of the high-quality synthetic dataset. We find that synthetic data could significantly boost world model learning.

B. Results on the Imitation Learning Benchmark

1) Baseline and Task Selection: To genuinely reflect the data quality of the RoboVerse dataset and provide a standard

[1]Due to resource and time constraints, we uniformly sample 20 testing scenarios for the OpenVLA baseline.

Representative Task Benchmark Source		PickCube ManiSkill	StackCube ManiSkill	CloseBox RLBench	MoveSliderLeft CALVIN	PickChocolatePudding LIBERO	NutAssembly RoboSuite	Average -
Diffusion Policy[13]	78M	52.7	53.8	51.5	76.5	50.0	7.1	48.6
ACT[137]	84M	31.7	36.7	68.3	85.0	78.3	0.0	50.0

TABLE II: **Baseline Results on RoboVerse Imitation Learning Benchmark.** We report baseline results on representative tasks from various benchmark sources to validate the effectiveness and reliability of the RoboVerse benchmark.

Task and Generalization Level	MoveSliderLeft				CloseBox				PickCube			
	Level 0	Level 1	Level 2	Level 3	Level 0	Level 1	Level 2	Level 3	Level 0	Level 1	Level 2	Level 3
Diffusion Policy [13]	76.5	81.3	72.0	60.0	51.5	42.8	20.0	10.4	52.7	11.1	0.0	0.0
ACT [137]	85.0	83.3	43.3	16.6	68.3	73.3	0.0	20.0	31.7	30.0	6.7	3.3
OpenVLA[1] [56]	45.0	40.0	35.0	30.0	0.0	0.0	0.0	0.0	40.0	15.0	0.0	0.0

TABLE III: **Generalization Performance on Imitation Learning Benchmark.** This table presents the experimental results for each generalization level in our benchmark across different tasks and methodologies. The tasks are divided into distinct levels (Level 0, Level 1, Level 2, and Level 3) to evaluate performance under progressively challenging scenarios.

Method	Simple		Language-conditioned Grasping		
	PickCube	MoveSliderLeft	Object Set 1	Object Set 2	Object Set 3
OpenVLA	40.0	45.0	46.0	33.3	14.4
Octo	50.0	30.0	42.0	14.4	2.2

TABLE IV: **Vision-Language-Action (VLA) Model Results on RoboVerse Imitation Learning Benchmark.** Constrained with time and resources, we report VLA models' results on two simple tasks from RoboVerse and grasping tasks with diverse and challenging language instructions. We split 58 objects in GraspNet into three sets, each containing progressively more challenging objects based on their geometry.

benchmark for all kinds of imitation learning policy models, we select both prevailing specialist and generalist models as baselines of our RoboVerse benchmark. Specifically, for specialist models, we integrate ACT [137] and Diffusion Policy [13]. For generalist models, We benchmark our approach on OpenVLA [56] and Octo [85], both of which we fine-tuned using our synthetic dataset. ACT is one of the most widely used methods in bi-manual manipulation. Diffusion Policy [13] is the first work that applies the conditional denoising diffusion process as a robot visuomotor policy and achieves great generalization capabilities. OpenVLA is the largest open-source vision-language-action model with 7B parameters.

Leveraging the RoboVerse format and infrastructure design, we are able to evaluate models on different tasks within a unified platform. To fully test policy models' performance under versatile settings, we select one representative task from each of the source benchmarks integrated by the RoboVerse dataset as shown in Tab. II. The experiment subset includes PickCube and StackCube from ManiSkill [80], CloseBox from RLBench [48], MoveSliderLeft from CALVIN [78], PickChocolatePudding from LIBERO [64], and NutAssembly on RoboSuite [141]. These tasks not only demand precise Pick-and-Place skills but also require contact-rich physical interactions with articulated objects. Through these tasks, the benchmark results can provide a comprehensive reflection of each model's performance under different scenarios.

2) Implementation Details: Due to time and resource constraints, we implement specialist and generalist models using different strategies, and all the results are obtained under the single-task setting. The training and evaluation settings follow the $90/10$ RoboVerse benchmark protocol as specified in V-B. During evaluations, we randomly select ten task settings from training sets and another ten from the validation sets. The reported success rates are computed as the averages over three random seeds.

For each step, the inputs are $256 \times 256 \times 3$ RGB images and a short language description depending on the task settings. For specialist models, we train from scratch with action in 9-dim robot joint state space. For generalist models, the action is pre-processed into delta end-effector position space from absolute end-effector position space, and The gripper action is discretized into binary values $\{0, +1\}$. Owing to the lack of time and resources, we are only able to fine-tune the generalist models in the single-task setting. During evaluations, we employ Curobo [105] as the inverse-kinematics solver to transform the action to robot joint state space. Specific model implementation details and hyperparameters are provided in supplementary materials.

3) Experiment Results: We present the imitation learning benchmark results in Tab. II and the generalization evaluation in Tab. III. We further fine-tune large vision-language-action models on both simple and complex language-conditioned tasks, as shown in Tab. VIII.

C. Results on the Reinforcement Learning Benchmark

Using STABLE-BASELINES3 and RSL_RL implementations of PPO, we train policies on tasks from IsaacLab [79] under consistent hyperparameters.

For additional tasks (humanoid, dexterous hand), the same PPO-based workflow applies. We successfully migrate the HumanoidBench [101] from MuJoCo to RoboVerse, enabling training across multiple simulators (IsaacLab and MuJoCo) with consistent interfaces. Experiment results demonstrate stable

policy convergence across simulators, achieving comparable performance to native MuJoCo baselines. Leveraging the generalizability of RSL_RL, we further extend the benchmark to support TD-MPC2 [41, 42] algorithm , which exhibits robust training dynamics in all environments. For implementation details, reward curve, and extended experimental results, please refer to the supplementary materials.

D. Augmentation Experiments

To verify the effectiveness of our trajectory augmentation API, on four representative tasks, we compare the success rates of trained Diffusion Policy on 50 source demonstrations and 200, 1000, and 3000 generated augmentation demonstrations under the imitation learning setting. The results presented in Fig. 10 demonstrate a consistent improvement in model performance as the number of generated data increases, highlighting both the effectiveness and scalability of the trajectory augmentation API.

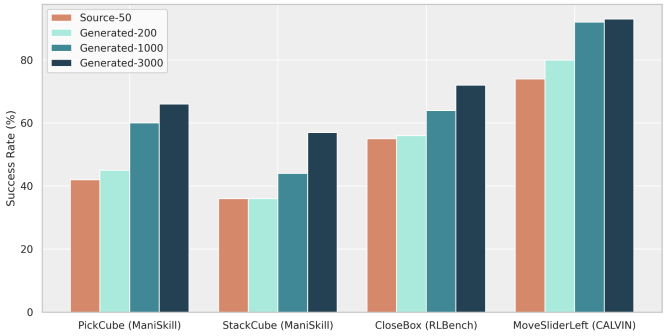


Fig. 10: **Effectiveness of Trajectory Augmentation.** Success rates of policy trained with augmented dataset and source dataset.

E. World Model Learning

Recent advances in general-purpose video generation and interactive world models [109, 6] have shown promising progress. Yet, the scarcity of gigantic-scale robotic datasets still impedes the development of robust world models for a wide range of robotic applications. In this session, we demonstrate how synthetic data from the RoboVerse simulation can augment real-world datasets to train more capable robotics world models.

When a model is trained exclusively on 50,000 episodes from the DROID dataset [54], it generally respects action conditions but struggles to accurately capture physical interactions between the gripper and target objects. Notably, the objects appear "warped" during contact with the gripper, as shown in Fig. 11. By incorporating an additional 50,000 synthetic episodes from RoboVerse to create a combined dataset of 100,000 episodes, the model predictions improve with regard to preserving object geometry. However, merely "watching videos" remains insufficient for learning the intricate physical interactions in DROID.

In contrast, training solely on the RoboVerse-50K or on the DROID-RoboVerse-100K dataset and then validating on

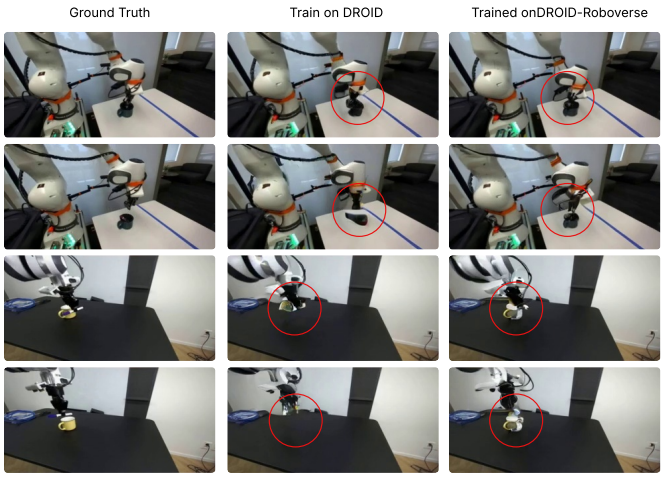


Fig. 11: **Ablation Study of Action-conditioned World Model Learning.** We compare the qualitative results of an action-conditioned world model trained on pure DROID and DROID-RoboVerse datasets, with evaluations sampled from the DROID dataset.

RoboVerse samples, we observe that the generated frames are physically more realistic in most scenes, with details in the supplementary materials. This improvement can be attributed to the extensive randomization and augmentation available in RoboVerse. Conversely, a model trained solely on DROID data fails to transfer effectively to the RoboVerse scene. We hypothesize that this shortcoming stems from limited samples per scene coverage in DROID and incomplete gripper visibility in the camera view.

F. Imitating the RoboVerse Dataset Enables Direct Sim-to-Real Transfer

The RoboVerse system seamlessly integrates a powerful physics engine with a high-quality renderer, ensuring the generation of realistic, high-fidelity data. To demonstrate its potential, we conduct experiments validating its effectiveness in direct sim-to-real transfer. As shown in Fig. 14, we fine-tune OpenVLA[56] on the RoboVerse dataset and transfer the learned policy to real-world scenarios without additional fine-tuning. The model successfully manipulates unseen objects in previously unseen real-world environments, showcasing the robustness and generalization capabilities of our system. The quantitative results on more challenging language-guided tasks, as shown in Tab. V, further demonstrate the high success rate of models trained on the RoboVerse dataset. Additional details are provided in the supplementary materials.

G. Reinforcement Learning in RoboVerse Enables Sim-to-Sim-to-Real Transfer

Large-scale parallel environments offer significant potential for large-scale exploration and are highly effective for reinforcement learning (RL) tasks. However, while they provide excellent efficiency, their accuracy may be limited in certain scenarios [25]. We also demonstrated some failure cases

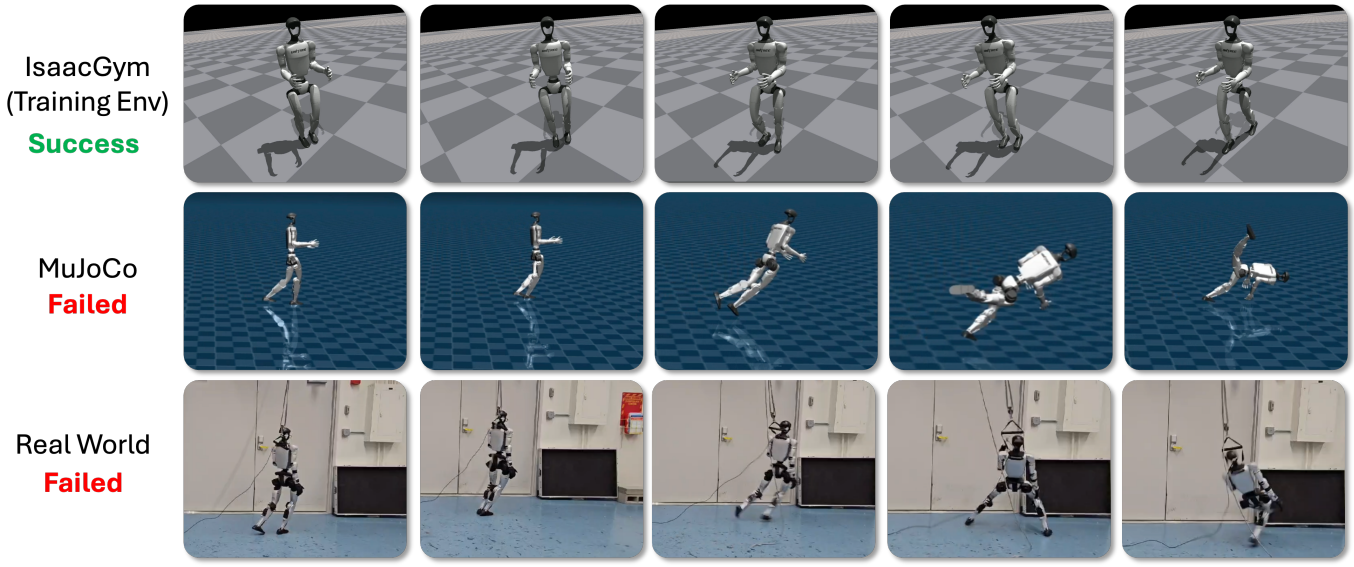


Fig. 12: **Simulation Comparison and Failures of Direct Sim-to-Real.** Training a locomotion policy directly in current parallel simulation, e.g., IsaacGym[71], and deploying it to the real world often results in significant failures. Similar failures are observed in sim-to-sim transfers. We observe that MuJoCo [110] demonstrates higher fidelity to real-world dynamics in this context.

GraspNet Objects	Pick up Wash Soap	Lift Mouth Rinse	Grasp Green Dish
Octo[85]	5.0/10.0	3.0/10.0	6.0/10.0
OpenVLA[56]	7.0/10.0	8.0/10.0	5.0/10.0

TABLE V: **Direct Sim-to-Real.** We fine-tune two baseline models using demonstrations adapted from GraspNet [27] to validate the effectiveness of the RoboVerse dataset. The final performance score for each task is reported, where a baseline receives 1 point for successfully grasping the target. Additionally, we adopt the partial reward scheme from OpenVLA, awarding 0.5 points when the gripper makes contact with the target.

directly from sim-to-real transfer (see Fig. 12). To address this problem, Sim-to-sim evaluation and fine-tuning present promising solutions [62]. The RoboVerse platform seamlessly supports such functionalities, enabling robust sim-to-sim and sim-to-real transitions. We further demonstrate the effectiveness of sim-to-sim-to-real generalization through comprehensive experiments, highlighting the platform's ability to bridge simulation and real-world performance.

VII. Limitations

While RoboVerse provides a comprehensive and scalable platform, several limitations remain. First, the integration of a unified format for non-rigid objects is not yet fully supported, which we leave for future work to develop. Additionally, while our large-scale dataset presents significant potential for pretraining a foundation model, this exploration falls beyond the scope of this paper due to resource constraints. Furthermore, despite our extensive efforts to fully reimplement and optimize all baseline methods within the RoboVerse baselines, some implementations may still be suboptimal. Our primary goal is not to directly compare policy performance but to demonstrate that the system is comprehensive, supports diverse policies, and ensures strong alignment between simulation and real-world performance. While we have made every effort to build a robust platform, it is inevitable that some oversights or errors may remain. We encourage the broader research community to contribute to maintaining and refining the baselines, fostering collaboration to further enhance the platform's capabilities.

Acknowledgement

We thank Hanyang Zhou and Sicheng He for providing valuable suggestions for setting up robotics hardware. We thank Yufeng Chi and Sophia Shao for providing humanoid robots for testing. We thank Jiawei Yang for facilitating access to the facility where the experiments were conducted. We thank Jie Yang and Muzhi Han for valuable discussion.

Pieter Abbeel holds concurrent appointments as a professor at UC Berkeley and as an Amazon Scholar. This paper describes work performed at UC Berkeley and is not associated with Amazon.

Fig. 13: **Sim-to-Real and Sim-to-Sim-to-Real Experiment Results.** We demonstrate that learning within the RoboVerse framework enables seamless direct Sim-to-Real transfer for manipulating unseen objects in new environments (imitation learning) and Sim-to-Sim-to-Real transfer for whole-body humanoid control (reinforcement learning).

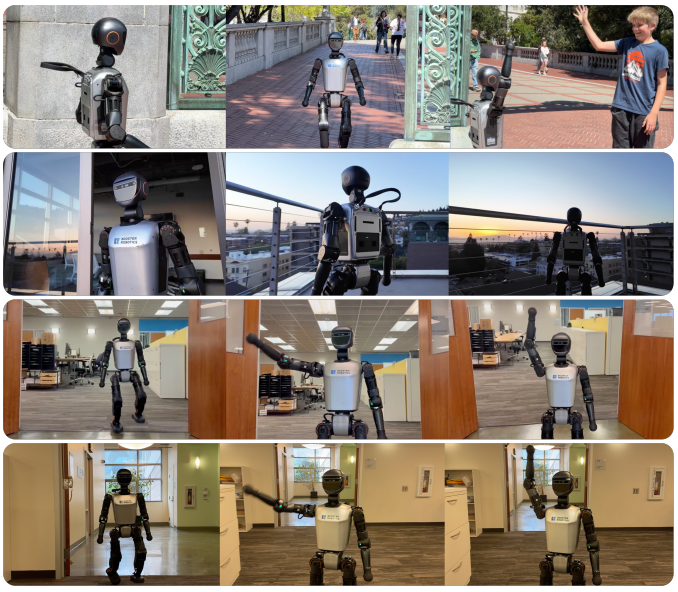


Fig. 14: **Generalization of Sim-to-Sim-to-Real.** This figure shows the in-the-wild generalization ability of the trained whole-body control policy by the sim-to-sim-to-real approach.

References

[1] Unai Antero, Francisco Blanco, Jon Oñativia, Damien Sallé, and Basilio Sierra. Harnessing the power of large language models for automated code generation and verification. *Robotics*, 13(9):137, 2024.

[2] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

[3] Tamir Blum, Gabin Paillet, Mickael Laine, and Kazuya Yoshida. Rl star platform: Reinforcement learning for simulation based training of robots. *arXiv preprint arXiv:2009.09595*, 2020.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

[6] Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim

Rocktäschel. Genie: Generative interactive environments, 2024. URL https://arxiv.org/abs/2402.15391.

[7] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics & Automation Magazine*, 22(3):36–52, 2015. doi: 10.1109/MRA.2015.2448951.

[8] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015.

[9] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pages 667–676. IEEE, 2017.

[10] Yuanpei Chen, Chen Wang, Yaodong Yang, and Karen Liu. Object-centric dexterous manipulation from human motion data. In *8th Annual Conference on Robot Learning*, 2024.

[11] An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024.

[12] 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.

[13] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[14] Alberto Silvio Chiappa, Alessandro Marin Vargas, Ann Huang, and Alexander Mathis. Latent exploration for reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[16] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[17] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2023.

[18] Wenbo Cui, Chengyang Zhao, Songlin Wei, Jiazhao Zhang, Haoran Geng, Yaran Chen, and He Wang. Gapartmanip: A large-scale part-centric dataset for material-agnostic articulated object manipulation. *arXiv preprint arXiv:2411.18276*, 2024.

[19] Qiyu Dai, Jiyao Zhang, Qiwei Li, Tianhao Wu, Hao Dong, Ziyuan Liu, Ping Tan, and He Wang. Domain randomization-enhanced depth simulation and restoration for perceiving and grasping specular and transparent objects. In *European Conference on Computer Vision*, pages 374–391. Springer, 2022.

[20] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[21] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[22] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram S. Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *ArXiv*, abs/2307.05663, 2023. URL https://api.semanticscholar.org/CorpusID:259836993.

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[24] Yufei Ding, Haoran Geng, Chaoyi Xu, Xiaomeng Fang, Jiazhao Zhang, Songlin Wei, Qiyu Dai, Zhizheng Zhang, and He Wang. Open6dor: Benchmarking open-instruction 6-dof object rearrangement and a vlm-based approach. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7359–7366. IEEE, 2024.

[25] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning, 2019. URL https://arxiv.org/abs/1904.12901.

[26] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 4397–4404. IEEE, 2015.

[27] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 11444–11453, 2020.

[28] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 653–660. IEEE, 2024.

[29] 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.

[30] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision (IJCV)*, 2021.

[31] Caelan Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment. *arXiv preprint arXiv:2410.18907*, 2024.

[32] Haoran Geng, Ziming Li, Yiran Geng, Jiayi Chen, Hao Dong, and He Wang. Partmanip: Learning cross-category generalizable part manipulation policy from point cloud observations. In *CVPR*, pages 2978–2988, 2023.

[33] Haoran Geng, Songlin Wei, Congyue Deng, Bokui Shen, He Wang, and Leonidas Guibas. Sage: Bridging semantic and actionable parts for generalizable articulated-object manipulation under language instructions, 2023.

[34] Haoran Geng, Helin Xu, Chengyang Zhao, Chao Xu, Li Yi, Siyuan Huang, and He Wang. Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7081–7091, 2023.

[35] Yiran Geng, Boshi An, Haoran Geng, Yuanpei Chen, Yaodong Yang, and Hao Dong. Rlafford: End-to-end affordance learning for robotic manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5880–5886. IEEE, 2023.

[36] Ran Gong, Jiangyong Huang, Yizhou Zhao, Haoran Geng, Xiaofeng Gao, Qingyang Wu, Wensi Ai, Ziheng Zhou, Demetri Terzopoulos, Song-Chun Zhu, et al. Arnold: A benchmark for language-grounded task learning with continuous states in realistic 3d scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[37] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

[38] Xinyang Gu, Yen-Jen Wang, and Jianyu Chen. Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer, 2024. URL https://arxiv.org/abs/2404.05695.

[39] Huy Ha, Pete Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 3766–3777. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/ha23a.html.

[40] Huy Ha, Yihuai Gao, Zipeng Fu, Jie Tan, and Shuran Song. Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. *arXiv preprint arXiv:2407.10353*, 2024.

[41] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

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

[43] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[44] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

[45] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. doi: 10.1145/3641519.3657428.

[46] Marcel Hussing, Jorge A Mendez, Anisha Singrodia, Cassandra Kent, and Eric Eaton. Robotic manipulation datasets for offline compositional reinforcement learning. *arXiv preprint arXiv:2307.07091*, 2023.

[47] Stephen James, Marc Freese, and Andrew J. Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

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

[49] Baoxiong Jia, Yixin Chen, Huangyue Yu, Yan Wang, Xuesong Niu, Tengyu Liu, Qing Li, and Siyuan Huang. Sceneverse: Scaling 3d vision-language learning for grounded scene understanding. In *European Conference on Computer Vision (ECCV)*, 2024.

[50] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. *arXiv preprint arXiv:2410.24185*, 2024.

[51] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020.

[52] Pushkal Katara, Zhou Xian, and Katerina Fragkiadaki. Gen2sim: Scaling up robot learning in simulation with generative models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6672–6679. IEEE, 2024.

[53] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[54] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *CoRR*, 2024.

[55] Chung Min Kim, Michael Danielczuk, Isabella Huang, and Ken Goldberg. Ipc-graspsim: Reducing the sim2real gap for parallel-jaw grasping with the incremental potential contact model, 2022. URL https://arxiv.org/abs/2111.01391.

[56] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag 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. *arXiv preprint arXiv:2406.09246*, 2024.

[57] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[58] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, 2020. URL https://api.semanticscholar.org/CorpusID:214802389.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[60] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4392–4412, 2020.

[61] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023.

[62] 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.

[63] Yu Li, Xiaojie Zhang, Ruihai Wu, Zilong Zhang, Yiran Geng, Hao Dong, and Zhaofeng He. Unidoormanip: Learning universal door manipulation policy over large-scale and diverse door manipulation environments. *arXiv preprint arXiv:2403.02604*, 2024.

[64] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[65] Ruoshi Liu, Alper Canberk, Shuran Song, and Carl Vondrick. Differentiable robot rendering, 2024. URL https://arxiv.org/abs/2410.13851.

[66] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21013–21022, 2022.

[67] Haozhe Lou, Yurong Liu, Yike Pan, Yiran Geng, Jianteng Chen, Wenlong Ma, Chenglong Li, Lin Wang, Hengzhen Feng, Lu Shi, Liyi Luo, and Yongliang Shi. Robo-gs: A physics consistent spatial-temporal model for robotic arm with hybrid representation, 2024. URL https://arxiv.org/abs/2408.14873.

[68] Haoran Lu, Ruihai Wu, Yitong Li, Sijie Li, Ziyu Zhu, Chuanruo Ning, Yan Shen, Longzan Luo, Yuanpei Chen, and Hao Dong. Garmentlab: A unified simulation and benchmark for garment manipulation. In *Advances in Neural Information Processing Systems*, 2024.

[69] Jiangran Lyu, Yuxing Chen, Tao Du, Feng Zhu, Huiquan Liu, Yizhou Wang, and He Wang. Scissorbot: Learning generalizable scissor skill for paper cutting via simulation, imitation, and sim2real. *arXiv preprint arXiv:2409.13966*, 2024.

[70] Xin Ma, Yaohui Wang, Gengyun Jia, Xinyuan Chen, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation, 2024. URL https://arxiv.org/abs/2401.03048.

[71] 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.

[72] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[73] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733*, 2020.

[74] 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. *arXiv preprint arXiv:2108.03298*, 2021.

[75] 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.

[76] Jiageng Mao, Siheng Zhao, Siqi Song, Tianheng Shi, Junjie Ye, Mingtong Zhang, Haoran Geng, Jitendra Malik, Vitor Guizilini, and Yue Wang. Learning from massive human videos for universal humanoid pose control. *arXiv preprint arXiv:2412.14172*, 2024.

[77] Pablo Martinez-Gonzalez, Sergiu Oprea, Alberto Garcia-Garcia, Alvaro Jover-Alvarez, Sergio Orts-Escolano, and Jose Garcia-Rodriguez. Unrealrox: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. *Virtual Reality*, 24:271–288, 2020.

[78] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[79] 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.

[80] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

[81] 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.

[82] NVidia. Physx, 2024. URL https://nvidia-omniverse.github.io/PhysX/physx/5.5.0/.

[83] NVidia. vmaterials, 2024. URL https://developer.nvidia.com/vmaterials.

[84] NVIDIA. Isaacsim simulator, 2025. URL https://developer.nvidia.com/isaac/sim.

[85] 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, 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.

[86] Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, and Angela P Schoellig. Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7512–7519. IEEE, 2021.

[87] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3D with transformers. In *CVPR*, 2024.

[88] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871.

[89] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023. URL https://arxiv.org/abs/2307.01952.

[90] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023.

[91] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dietor Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023.

[92] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9, 2019.

[93] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[94] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. URL https://arxiv.org/abs/2408.00714.

[95] Pengzhen Ren, Min Li, Zhen Luo, Xinshuai Song, Ziwei Chen, Weijia Liufu, Yixuan Yang, Hao Zheng, Rongtao Xu, Zitong Huang, et al. Infiniteworld: A unified scalable simulation framework for general visual-language robot interaction. *arXiv preprint arXiv:2412.05789*, 2024.

[96] E. Rohmer, S. P. N. Singh, and M. Freese. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*,

2013. www.coppeliarobotics.com.

[97] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2022. URL https://arxiv.org/abs/2109.11978.

[98] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[99] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

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

[101] Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation, 2024.

[102] Arth Shukla, Stone Tao, and Hao Su. Maniskill-hab: A benchmark for low-level manipulation in home rearrangement tasks, 2024. URL https://arxiv.org/abs/2412.13211.

[103] Simulately Wiki. Simulately wiki. https://simulately.wiki, 2025. Accessed: 31 Jan 2025.

[104] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

[105] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. Curobo: Parallelized collision-free robot motion generation. In *ICRA*, pages 8112–8119. IEEE, 2023.

[106] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023.

[107] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266, 2021.

[108] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

[109] Movie Gen team. Movie gen: A cast of media foundation models, 2024. URL https://arxiv.org/abs/2410.13720.

[110] 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*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

[111] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[112] Christine M Vaccaro, Catrina C Crisp, Angela N Fellner, Christopher Jackson, Steven D Kleeman, and James Pavelka. Robotic virtual reality simulation plus standard robotic orientation versus standard robotic orientation alone: a randomized controlled trial. *Urogynecology*, 19 (5):266–270, 2013.

[113] Daniel Vlasic, Rolf Adelsberger, Giovanni Vannucci, John Barnwell, Markus Gross, Wojciech Matusik, and Jovan Popović. Practical motion capture in everyday surroundings. *ACM transactions on graphics (TOG)*, 26 (3):35–es, 2007.

[114] Fang Wan, Haokun Wang, Xiaobo Liu, Linhan Yang, and Chaoyang Song. Deepclaw: A robotic hardware benchmarking platform for learning object manipulation. In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 2011–2018. IEEE, 2020.

[115] Hanqing Wang, Jiahe Chen, Wensi Huang, Qingwei Ben, Tai Wang, Boyu Mi, Tao Huang, Siheng Zhao, Yilun Chen, Sizhe Yang, et al. Grutopia: Dream general robots in a city at scale. *arXiv preprint arXiv:2407.10943*, 2024.

[116] Jun Wang, Yuzhe Qin, Kaiming Kuang, Yigit Korkmaz, Akhilan Gurumoorthy, Hao Su, and Xiaolong Wang. Cyberdemo: Augmenting simulated human demonstration for real-world dexterous manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17952–17963, 2024.

[117] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. *arXiv preprint arXiv:2310.01361*, 2023.

[118] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.

[119] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Muller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. BundleSDF: Neural 6-DoF tracking and 3D reconstruction of unknown objects. In *CVPR*, 2023.

[120] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6d pose estimation and tracking of novel objects. In *CVPR*, 2024.

[121] 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.

[122] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tUM39YTRxH.

[123] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4737–4746, 2023.

[124] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

[125] Xintong Yang, Ze Ji, Jing Wu, and Yu-Kun Lai. An open-source multi-goal reinforcement learning environment for robotic manipulation with pybullet. In *Annual Conference Towards Autonomous Robotic Systems*, pages 14–24. Springer, 2021.

[126] Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[127] Chongjie Ye, Yinyu Nie, Jiahao Chang, Yuantao Chen, Yihao Zhi, and Xiaoguang Han. Gaustudio: A modular framework for 3d gaussian splatting and beyond. *arXiv preprint arXiv:2403.19632*, 2024.

[128] Chongjie Ye, Lingteng Qiu, Xiaodong Gu, Qi Zuo, Yushuang Wu, Zilong Dong, Liefeng Bo, Yuliang Xiu, and Xiaoguang Han. Stablenormal: Reducing diffusion variance for stable and sharp normal. *ACM Transactions on Graphics (TOG)*, 2024.

[129] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. URL https://arxiv.org/abs/2409.06765.

[130] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897.

[131] 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.

[132] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017.

[133] Jialiang Zhang, Haoran Liu, Danshi Li, XinQiang Yu, Haoran Geng, Yufei Ding, Jiayi Chen, and He Wang. Dexgraspnet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes. In *8th Annual Conference on Robot Learning*, 2024.

[134] Jialiang Zhang, Haoran Liu, Danshi Li, Xinqiang Yu, Haoran Geng, Yufei Ding, Jiayi Chen, and He Wang. Dexgraspnet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes, 2024. URL https://arxiv.org/abs/2410.23004.

[135] Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*, 2024.

[136] Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. *Robotics: Science and Systems*, 2024.

[137] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[138] Chengwei Zheng, Lixin Xue, Juan Zarate, and Jie Song. Gstar: Gaussian surface tracking and reconstruction, 2025. URL https://arxiv.org/abs/2501.10283.

[139] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130 (9):2337–2348, 2022.

[140] Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. Irasim: Learning interactive real-robot action simulators, 2024. URL https://arxiv.org/abs/2406.14540.

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

CONTENTS

VIII. Simulators Overview

In the field of robotics, simulators play an important role. It is the womb of a robot, taking responsibility for training and testing a robot's behaviors before it was "born" into the real world. Therefore, the functionalities are crucial for a successful robotic application. Users require different functions of simulators according to their specific scenarios: whether it is a photorealistic task which requires accurate rendering of a close-to-real virtual world, or a massive parallel scene that is designed for efficient reinforcement learning. All the requirements may influence the choice of the simulator. In order to reduce the pain users need to endure in getting them familiarized with each new simulator, we incorporated multiple simulators into the `RoboVerse` platform and listed specifications of the simulators currently supported by `RoboVerse` in Fig. VI.

Due to the complexity of physics simulation and rendering, current simulators cannot depict the real world well enough. Our experiments revealed some common issues of nowadays simulators in the basic physics laws. The experimental results on fundamental conservation laws may be a pessimistic sign on our hope of direct sim-to-real transfer of more complicated robotic behaviors.

We conducted experiments on three basic conservation laws of physics in three simulators.

In the experiments for Conservation of Momentum, two rigid bodies are placed in a gravity-free environment, their initial states are set to have an elastic collision.

In the experiments for Conservation of Angular Momentum, one or two rigid bodies are placed in the gravity-free environment, and their initial states are set to rotate. We calculate and record the overall angular momentum as the system evolves.

In the experiments for Conservation of Kinetic Energy, two rigid bodies are placed in the gravity-free environment, and their initial states are set to have a rotation-free elastic collision. This setup allows us to directly observe the conservation of kinetic energy regardless of the results of experiments on angular momentum.

From the results listed in Fig. 15, we can easily notice that basic conservation laws are not kept in the three simulators. However, different simulators behave differently in different experimental setups, which indicates that depending on the needs of different tasks, we may need to choose different simulators for more accurate results. This highlights the necessity of a tool that helps users to easily transfer tasks among simulators.

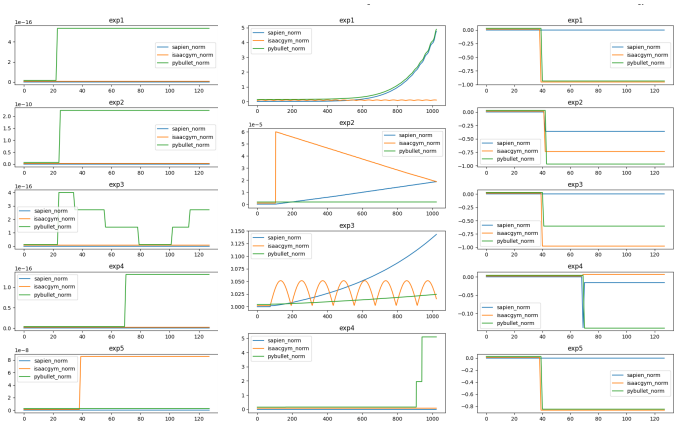
IX. The MetaSim Framework

A. Architecture Overview

The MetaSim framework is a unified simulation framework as shown in Fig. 16. On the front-end side, it provides user-friendly Gym APIs as well as easy-to-use parallel environment support. On the back-end side, it supports multiple simulators to allow seamless transfer of tasks across simulators. Users only need to master simple skills on writing a simulator-agnostic

Simulator	Physics Engine	Rendering	Sensor Support	Dynamics	GPU	Open
SAPIEN [121]	PhysX-5, Warp	Rasterization RayTracing	RGBD; Force; Contact	Rigid; Soft; Fluid	✓	✓
Pybullet [16]	Bullet	Rasterization	RGBD; Force IMU; Tactile	Rigid; Soft; Cloth		✓
MuJoCo [110]	MuJoCo	Rasterization	RGBD; Force IMU; Tactile	Rigid;Soft;Cloth	✓	✓
CoppeliaSim [96]	MuJoCo; Bullet ODE; Newton; Vortex	Rasterization	RGBD; Force; Contact	Rigid;Soft;Cloth		✓
IsaacLab [79]	PhysX-5	RayTracing	RGBD; Lidar; Force Effort; IMU; Contact Proximity	Rigid; Soft Cloth; Fluid	✓	
IsaacGym [71]	PhysX-5, Flex	Rasterization	RGBD; Force; Contact	Rigid; Soft; Cloth	✓	
Genesis [2]	Genesis	Rasterization RayTracing	RGBD; Force; Tactile	Rigid; Soft	✓	✓

TABLE VI: Comparison of Physics Simulators [103]. The column **GPU** denotes whether the simulator can use GPU-accelerated computation. The column **Open** denotes whether the simulator is open-source.



(a) Momentum (b) Angular (c) Kinetic Energy

Fig. 15: Three series of experiments on conservation laws in simulators. Blue, orange and green lines are data collected from Sapien, Isaacgym and Pybullet respectively.

`MetaConfig` configuration class, the environment will then be automatically instantiated with the designated back-end simulator.

B. `MetaConfig` Configuration System

The METASIM framework uses `MetaConfig`, a unified configuration class to describe a scenario in simulation environments.

We designed a configuration system that set up the simulator, define the tasks, set up the domain randomization. In order to run the same setting of environments across different simulators, the configuration system is defined to be simulator-agnostic as much as possible. For simulator-specific settings (*e.g.* rendering mode, physics engine solver type, *etc.*), there is a seperate simulator-specific part which defines those things.

To make changing the settings and debug more easily, we design the configuration system in a Hydra[124]-like way, making each item in the configuration system can be modified from commandline just like Hydra [124]. The configuration system is implemented based on Python dataclass, and could therefore use Python type annotation to help user use them.

In order to run the tasks seamlessly across all simulators, it is necessary to define them in a simulator-agnostic way. We configure the task and define its objects list, robot in use, success checker and the reward. The success checker is used to determine when the task is successfully execucated, and is the most difficult part in task definition. To standardize, we offer some structured success checker templates which cover the most cases, and leave option for users to define a callback function for flexibility to implement those stuctured success checker could not cover.

C. Aligned Simulation APIs

METASIM support different simulator backends, including IsaacSim [84], IsaacGym [71], MuJoCo [110], PyBullet [16], SAPIEN [121], CoppeliaSim [96, 47]. The framework is implemented in Python, as these simulators either natively support Python or provide Python APIs.

Common simulator operations are unified in a `Handler` class. Each handler supports only tree basic APIs: `get_state()`, `set_state()` and `step()`. The `get_state()` method takes a descriptive Python dict (*e.g.*, `{object_name: {'pos': ..., 'rot': ..., '...': ...}}`) as input, and returns current simulation states according to the dict in another Python dict structured in the same manner. The `set_state()` method also takes a descriptive Python dict as input, and modifies current

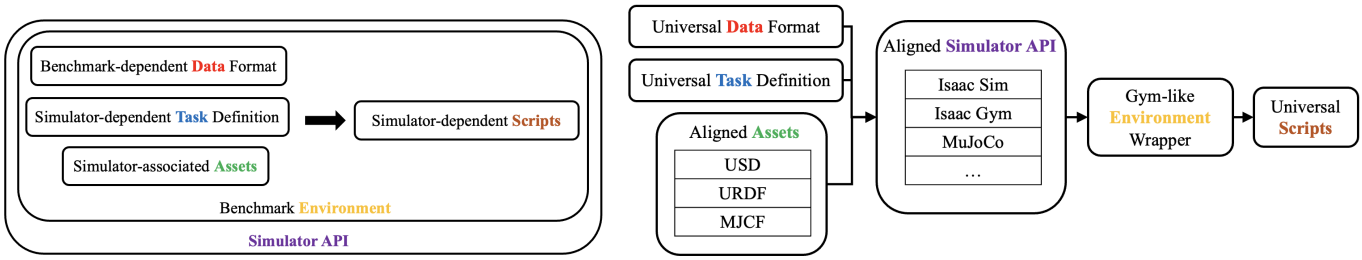


Fig. 16: Comparison between the METASIM and the other simulation environments. Left: Other simulator and benchmark, using self-defined data format, simulator-associated assets, simulator-dependent task definition, and scripts. Right: The METASIM, decoupling all components to be agnostic to specific simulators or benchmark environments.

simulation states to the ones included in the dict. The `step()` method will prompt the simulation to proceed one timestep.

D. Gym API Wrappers

To support building learning environments, we define an `Env` class built on top of `Handler`. It offers Gymnasium-like APIs (`step`, `reset`, `render`, and `close`), implementing these methods by leveraging the underlying `Handler` methods.

It is worth noting that most simulation environments provide the underlying APIs (corresponding to our `Handler`) and upper-level environments (corresponding to our `Env`) seperately, such as SAPIEN [121] with ManiSkill [108], IsaacSim [84] with IsaacLab [79], CoppeliaSim [96]/PyRep [47] with RLBench [48], and Mujoco [110] with Mujoco Playground [131]. This fact proves our `Handler` and `Env` two-level abstraction reasonable.

E. Backend Support

1) Isaac Lab: Isaac Lab [84] is an advanced robotics simulation platform developed by NVIDIA. By leveraging high-fidelity physics, GPU acceleration, and photorealistic rendering, it enables rapid prototyping, testing, and deployment of AI-driven robotics solutions in virtual environments. Through seamless integration with NVIDIA's Omniverse framework, Isaac Lab offers robust features such as domain randomization, sensor simulation, and support for large-scale reinforcement learning, making it a powerful tool for both research and industrial applications.

A key advantage of Isaac Lab is its compatibility with the Isaac ROS infrastructure, which includes valuable models such as foundationpose [120, 119] and curobo [106], among others.

2) Isaac Gym: Isaac Gym [71] is a physics simulation environment designed for reinforcement learning research. Although it remains available for download, official support has ended. Nevertheless, multiple works published before 2024—such as hora [90], humanoid-gym [38], and IPC-graspsim [55]—were developed using Isaac Gym.

Key features of Isaac Gym include support for importing URDF and MJCF files with automatic convex decomposition, a GPU-accelerated tensor API for managing environment states and actions, and a range of sensors (e.g., position, velocity, force, torque). Additional capabilities include runtime domain randomization of physics parameters, Jacobian and inverse kinematics support, and customizable friction settings.

3) Mujoco: MuJoCo [110] is a physics engine and simulation framework designed to accurately model the dynamics and control of complex robotic systems in real-time. Its name, MuJoCo, stands for Multi-Joint dynamics with Contact, highlighting its primary emphasis on efficient computation of contact forces and multi-joint dynamics. The engine supports advanced features such as frictional contact models, user-defined actuators, and customizable sensor modalities, allowing researchers and developers to prototype, test, and refine control algorithms across a wide range of robot morphologies and tasks.

A key strength of MuJoCo is its computational precision, which enables high simulation throughput and real-time interactive control. It supports rigid-body dynamics, articulated mechanisms, and a variety of constraints, making it suitable for tasks involving locomotion, manipulation, and reinforcement learning. Furthermore, MuJoCo's flexible XML-based model description streamlines creating and modifying simulated environments, providing a straightforward way to experiment with novel designs. The compatibility between MuJoCo and Brax offers a high-speed, differentiable pipeline crucial for reinforcement learning. This powerful blend of accuracy, speed, and flexibility has solidified MuJoCo's status as a leading choice in robotics research and machine learning, particularly for advanced control, motion planning, and reinforcement learning applications [29].

4) Genesis: Genesis[2] is a comprehensive physics platform developed for robotics and physics simulation research, unifying multiple core capabilities in a single environment. At its foundation is a universal physics engine, rebuilt from the ground up to simulate diverse materials and physical phenomena while seamlessly integrating various solvers. Alongside this engine, Genesis provides a swift, Python-friendly robotics simulation toolkit, an efficient photo-realistic rendering system, and a data-generation module that converts natural language prompts into multi-modal datasets. We leverage the Genesis backend to support loading, simulation, and rendering in ROBOVERSE workflow.

5) Sapien: SAPIEN [121] is a robot simulation framework that allows highly efficient simulation and rendering of robotic

tasks. It uses PhysX [82] as the underlying physics engine. We supported the released version Sapien 2.2 for the METASIM framework.

We use the multipocessing library to support parallel environments in the `Handler` class for Sapien. When instantiating the environment from configurations, a desired number of processes are forked to run the simulation of different environments. To support the `get_states` and `set_states` API, data for different environments are distributed to different processes, and the return values are then gathered.

6) Pybullet: PyBullet [17] is a fast and easy-to-use robotics simulator. It uses its own physics solvers for accurate and efficient simulations. We supported the released version PyBullet 3.2 for the METASIM framework.

We use the same techniques as for Sapien to achieve parallel-environment simulation.

F. Hybrid Simulation Implementation

METASIM allows launching two simulators in one single process with one command. Taking our demo collection command as example: `python collect_demo.py --sim=mujoco --renderer=isaaclab --task=$task`. The implementation is illustrated in Code. 2.

```
class HybridEnv:
    def __init__(self, env_physic: Env,
    env_render: Env):
        ...
    def step(self, action):
        env_physic.handler.set_states(action=
    action)
        phys_states = env_physic.handler.
    get_states()
        env_render.handler.set_states(states=
    phys_states)
        env_render.handler.refresh_render()
        states = env_render.handler.get_states()
        return ...
```

Code 2: Pseudocode for implementing hybrid simulation using two different simulator environments simultaneously. The core of this implementation is using `states` as a unified representation across both simulation environments.

X. ASSET CONVERSION

A. Asset types

The diverse landscape of robotic assets, stemming from prior research initiatives [141, 48, 80] and a multitude of software platforms [110, 71, 121], necessitates a robust strategy for managing a wide array of file formats. To facilitate dependable cross-simulator training and uphold data integrity throughout the development lifecycle, the establishment of an efficient and reliable asset conversion pipeline is of paramount importance [26]. Such a pipeline is crucial for ensuring seamless interoperability, minimizing potential data loss or inaccuracies, and promoting the uniform application of metadata and configurations across disparate simulation environments. A selection of frequently encountered asset formats includes, but is not limited to, MuJoCo XML control files [110], URDF files [8], and USD files [84].

The three predominant file formats in robotics simulation: MJCF, URDF, and USD. Each of them serves distinct purposes and offers unique capabilities. MJCF (MuJoCo Configuration Format) stands out for its exceptional expressiveness in physics simulation, featuring sophisticated capabilities to model complex dynamical systems including tendons, actuators, and advanced joint configurations, along with an integrated compiler for handling complex compile-time computations [110]. URDF (Unified Robot Description Format), while more constrained in its feature set, has emerged as the de facto standard in robotics due to its remarkable cross-platform compatibility and universal adaptability across various simulation environments including Isaac Sim [84], Isaac Gym [71], MuJoCo [110], Gazebo, and PyBullet [16], making it ideal for robot model exchange despite its limitations in representing parallel mechanisms or complex sensor configurations [8]. USD (Universal Scene Description), originally developed by Pixar Animation Studios, excels in high-fidelity rendering and scene composition through its sophisticated layering system and variant sets [22], making it particularly valuable for applications requiring advanced visual properties and collaborative workflows [83], although its physics simulation capabilities are more limited compared to dedicated robotics formats like MJCF [26].

Features	**MJCF**	**URDF**	**USD**
Basic Geometries	✓	✓	✓
Mesh Support	✓	✓	✓
Texture Support	✓	Limited	✓
Material Properties	✓	Basic	✓
Physics Properties	✓	✓	Limited
Joint Types	Many	Basic	Basic
Collision Properties	Advanced	Basic	Advanced
Deformable Objects	✓	✗	✓
Animation Support	Limited	✗	✓
Scene Composition	Basic	✗	Advanced
File Format	XML	XML	ASCII/Binary

TABLE VII: Comparison of Robot Description Formats

B. Conversion Pipeline

Given that our simulation pipeline primarily utilizes Isaac Sim for rendering while many of our assets are originally stored in MJCF format, a two-stage conversion pipeline (MJCF → URDF → USD) becomes necessary and advantageous. This approach leverages URDF as an intermediate format for several reasons. First, while direct conversion from MJCF to USD is theoretically possible, such conversion would be complex and error-prone due to MJCF's rich feature set for physics properties

(like tendons and actuators) that lack direct equivalents in USD [114]. Instead, converting to URDF first allows us to standardize the robot's basic kinematic and dynamic properties in a format that has well-established conversion tools and widespread support. The subsequent URDF to USD conversion benefits from Isaac Sim's robust URDF importing capabilities, which have been extensively tested and optimized for robotics applications. This two-stage pipeline thus ensures more reliable asset conversion while maintaining essential physical properties and compatibility across different simulation environments.

1) MJCF to URDF conversion: We implemented our own MJCF to URDF converter by first parsing everything with MuJoCo's MJCF importer, then exporting all texture, collision mesh and joint information to the correct URDF format. The inspiration is taken from Genesis [2], which they built their own class for each asset object that encode all joint, texture and mesh information. We then recursively generate the body information to URDF and align everything with texture.

a) Parsing Link, Joint, and Body Information from the MJCF file: To parse link, joint, and body information from the MJCF file, we leverage MuJoCo's parsing capabilities to load the MJCF XML into a MuJoCo model structure. From this parsed model, we employ a recursive approach, starting from the root body and descending into each child body to systematically process the hierarchical structure. For each body, we extract detailed link properties such as name, position, orientation, inertial characteristics, and associated geometry. Simultaneously, we parse joint information connected to each body, including joint type, limits, and axis of motion. All of this extracted link and joint data is systematically organized and stored in dictionary structures. These dictionaries serve as intermediate representations, holding all the necessary information from the MJCF model in a structured format that is readily accessible for subsequent stages of the URDF conversion process.

b) Aligning Meshes and textures: The management of collision meshes across existing asset libraries presents a notable challenge, as these assets are typically stored in various formats including .msh, .obj, and .stl files. While URDF natively supports .obj and .stl formats, the conversion of .msh files into URDF-compatible formats requires careful consideration. Although MuJoCo's repository provides a conversion utility for transforming .msh files to .obj format—accomplished by parsing the .msh files through the MuJoCo interface and subsequently exporting vertex and face information—this approach introduces potential complications with texture mapping alignment.

The complexity arises from the specific requirements of texture files, which are predominantly stored as albedo PNG files. These textures depend on precise UV mapping coordinates within the .obj file to ensure proper alignment. The current .msh to .obj conversion utility provided in the MuJoCo repository does not adequately address texture support, leading to potential misalignment issues in the converted models. This limitation is particularly evident in comprehensive robotics frameworks such as Libero [64] , where both static and articulated objects frequently exhibit texture alignment discrepancies following the .msh to .obj conversion process.

Fortunately, we discovered that many asset collections maintain redundant mesh representations, often including a properly UV-mapped .obj file alongside the .msh file, typically sharing the same filename or designated as "textured.obj". Leveraging this observation, we implemented a robust mesh alignment pipeline that follows a hierarchical decision process:

- First, the system searches for an existing .obj file within the same directory as the .msh file
- If found, this pre-existing .obj file is utilized, ensuring proper texture alignment
- In the absence of a pre-existing .obj file, the system proceeds with the .msh to .obj conversion
- In the latter case, users receive a warning notification regarding potential texture misalignment issues

Following the mesh format resolution, the pipeline systematically maps these processed mesh files back to their corresponding links within the URDF structure, maintaining the integrity of the robot's geometric representation while preserving texture information where possible.

c) Building URDF: The assembling procedure after all the conversions become very aparent: we first processes robot links and joints, incorporating their properties and relationships into the URDF format. This automated approach ensures a robust and flexible method for generating URDF files, accommodating a wide range of robot configurations and properties derived from the preceding conversion steps.

Even though this pipeline roughly works for most of the MJCF, for some specific MJCF files in some specific folder, we have to modify our conversion approach on a case by case basis. Below is a table for some special treament we employed to specific packages, and its conversion success rate:

Despite the general efficacy of the described pipeline across a broad spectrum of MJCF assets, it is important to acknowledge that certain MJCF files, particularly those within specific packages or directories, necessitate bespoke conversion strategies. These exceptions arise due to the inherent complexity and variability in MJCF file structures across different projects and asset libraries. To address these unique cases, we have adopted a tailored approach, implementing case-specific modifications to our conversion pipeline as required. The subsequent table details instances where such specialized treatment has been applied, along with the corresponding conversion success rates achieved for each package.

2) URDF to USD conversion: IsaacSim has implemented a robust solution for converting URDF files to USD format. The conversion process comprehensively preserves the robot's structural and kinematic information, including joint hierarchies, geometric properties, and physical attributes. The implementation demonstrates exceptional fidelity in translating complex robotic descriptions, ensuring that all essential components—such as joint configurations, collision geometries, and visual representations—are accurately encoded in the resulting USD files.

Given the proprietary nature of IsaacSim's conversion implementation, we utilize their framework as an external tool in our pipeline. This approach leverages the proven reliability and performance of IsaacSim's converter while maintaining compatibility with our broader system architecture. The conversion process serves as a critical bridge between standard robotics formats and the high-performance USD representation required for our simulation environment.

XI. Task and Data Migration

A. ManiSkill

ManiSkill [80, 37, 108] provides a series of robotic manipulation tasks under single-arm or dual-arm settings.

Tasks and assets: We migrate basic single-arm tasks and demonstrations to RoboVerse, including the pick-and-place tasks like `PickCube` and `PickSingleYCB`, as well as the insertion tasks like `PegInsertionSide` and `PlugCharger`. The corresponding assets are manually crafted with primitives or process from the mesh files, with proper physics API set up.

Demonstrations: For each task, a great number of demonstration trajectories are available in the released data. Noteworthy, the data does not come with the initial scene states, which are obtained by replaying the demonstrations within the SAPIEN simulator. With the specified seed set, the states are recovered by the random samplers.The success checkers are implemented according to the task designs.

B. RLBench

RLBench [48] is a large-scale benchmark and learning environment for robotic manipulation, featuring 100 diverse, hand-designed tasks ranging in complexity, from simple actions like reaching to multi-stage tasks like opening an oven and placing a tray inside. Each task includes an infinite supply of demonstrations generated via waypoint-based motion planning.

Tasks and assets: We roll out $\sim 2K$ trajectories in RLBench [48] for each task, and migrate them to RoboVerse.

C. CALVIN

CALVIN [78] provides 6-hour teleopreation trajectories on 4 environments, each involve an articulated table with three blocks in blue, pink, or red.

Tasks and assets: We migrate the demonstrations in all 4 environments and transform the original assets (URDF for the table, and primitives for the cubes) into USD files with proper physics APIs.

Demonstrations: We segment the trajectories according to the text annotations, which specified the task category (*e.g.*, `PlaceInSlider`), the text annotation (*e.g.*, *place the red block in the slider*), and the timestamps of the demonstration segment. The states of the first frame is adopted as the scene initial states.

Success checkers: We carefully implement the success checkers according to the original implementation to make sure the failed executions can be filtered out. This is because the coarsely annotated timestamps in the dataset, which may cause the failed execution in part of the demonstrations.

D. MetaWorld

MetaWorld [130] is a widely used benchmark for multi-task and meta-reinforcement learning, comprising 50 distinct tabletop robotic manipulation tasks involving a Sawyer robot.

Tasks and Assets: We integrate five representative tasks into RoboVerse: *drawer open*, *drawer close*, *door close*, *window open*, and *window close*. The corresponding assets are manually converted from MJCF to USD files with appropriate physics APIs.

Demonstrations: As the benchmark does not provide demonstrations, we generate trajectories for each task by rolling out reinforcement learning policies from [122].

E. Open6DOR

Open6DOR is a benchmark for open-instruction 6-DoF object rearrangement tasks, which requires embodied agents to move the target objects according to open instructions that specify its 6-DoF pose.

Tasks and Assets: The synthetic object dataset comprises 200+ items spanning 70+ distinct categories. Originally derived from YCB[7] and Objaverse-XL[22], the objects are carefully filtered and scaled using a standardized format of mesh representation. Overall, the Open6DOR Benchmark consists of 5k+ tasks, divided into the position-track, rotation-track, and 6-DoF-track, each providing manually configured tasks along with comprehensive and quantitative 3D annotations.

Success checkers: We determine success by comparing the target object's final pose with the annotated ground-truth pose range.

F. ARNOLD

Arnold [36] is a benchmark for language-conditioned manipulation. The benchmark uses motion planning and keypoints for robot manipulation tasks, focusing on fine-grained language understanding.

Tasks and Assets: : We integrate six out of eight tasks from Arnold into RoboVerse: picking up objects, reorienting objects, opening/closing drawers, and opening/closing cabinets.

Demonstrations: As the benchmark does not use trajectory-level demonstrations, we use motion planning for trajectory generation to interpolate between keypoints

G. RoboSuite & MimicGen

RoboSuite [141] provides a suite of task environments for robotic manipulation, built on the MuJoCo physics engine. Each task is implemented as a separate class, with most configuration details embedded in the source code. Based on these environments, MimicGen [75] offers thousands of demonstrations, serving as a widely used benchmark for imitation learning.

Tasks and Assets: For tasks with separate object description files (MJCF), we directly migrate the corresponding assets through our Asset Conversion pipeline. However, some tasks contain hard-coded assets within the source code, such as a hammer composed of multiple cubes, cylinders and other primitives with carefully designed relative poses. To integrate

these tasks, we will manually reconstruct the assets within our framework. We also argue that hard-coded asset and task definitions, as opposed to modular task descriptions, are not scalable for future robotic task benchmarking.

Demonstrations: We convert MimicGen demonstrations into our format. Specifically, we transform the robot actions from 6-DoF Cartesian space representations to joint space. Additionally, the state of the first frame is adopted as the initial scene state.

Success Checkers: We meticulously implement success checkers based on the original definitions to ensure failed executions are effectively filtered out.

H. SimplerEnv

SimplerEnv is a set of tasks and methods designed to do trustworthy benchmarking in simulation for manipulation policies that can reflect the real-world success rate.

There are in total 25 different tasks in SimplerEnv. We ignore all tasks that are just a subset of another task and migrated in total 6 tasks and 52 object assets to ROBOVERSE. The tasks all use Google Robot.

SimplerEnv provided some controller models trained with RT-1 [4] and RT-X [15] dataset. We did not use the trajectories from the dataset directly because some environmental settings are different from the environments from SimplerEnv. We used the trained model to collect trajectories. Hooks are inserted into the original SimplerEnv codebase to extract and maintain the recordings at different stages of simulation. We then rollout the model trained with RT-1 dataset on each task to collect the trajectories.

I. GAPartNet

For tasks in GAPartNet [34], we generate both motion planning [34] and reinforcement learning [32] trajectories. GAPartNet is implemented in IsaacGym [71] with various articulated objects. To integrate it into RoboVerse, we first align all articulated object initial states to the MetaSim format and convert the asset format to USD for compatibility across different simulators.

For trajectory generation:

(1) **Motion Planning**: GAPartNet [34] introduces a part-centric manipulation approach. We roll out heuristics to generate manipulation trajectories, providing three demonstrations per part with different object and part initial states. (2) **Reinforcement Learning Rollout**: The follow-up work, PartManip [32], proposes several reinforcement learning methods. We re-train all policies based on our robot setup and roll out trajectories for dataset collection. With aligned task configurations, trajectories, and assets, we successfully adapt GAPartNet into RoboVerse.

J. GAPartManip

Instead of providing direct demonstrations, GAPartManip [18] offers a large-scale, part-oriented, scene-level dataset with annotations for actionable interaction poses. We utilize the mesh-level grasping pose annotations in this dataset to generate diverse demonstrations for articulated object manipulation.

Tasks and Assets: We currently implement two tasks: `OpenBox` and `OpenToilet`. For the `OpenBox` task, we collect 12 object assets from the *Box* category in the original dataset. For the `OpenToilet` task, we gather 30 objects from the *Toilet* category. We convert these assets into USD files with appropriate physics APIs to ensure compatibility with our simulation environment.

Demonstrations: We generate demonstrations for our tasks in simulation using motion planning with CuRobo [105]. First, we filter potential grasping poses for the target object link by assessing their feasibility through motion planning. Specifically, we discard poses that the end-effector cannot reach or that would cause a collision between the robot and the object. Next, we generate an end-effector pose trajectory to complete the task using heuristics. Based on the object's kinematic tree, we could define an ideal trajectory. We then apply motion planning to perform inverse kinematics, computing the corresponding joint poses of the robot along this trajectory. Finally, we execute the planned trajectory in simulation to verify task completion, saving successful trajectories as demonstrations. The entire demonstration generation process is conducted in IsaacSim [84].

Success Checkers: To determine task success, we require the manipulated object to be opened by at least 60 degrees for all tasks.

K. GraspNet-1B

GraspNet-1B [27] is a general object grasping dataset for predicting 6 DoF grasping pose given partial pointcloud input. It contains 256 realworld tabletop scenes consists of total 88 different objects. We carefully filter out 58 objects as our target grasping objects based on the availability of purchasing real items because we need to evaluate our policies to grasp them in the real world experiments. To generate grasping demonstrations, we use CuRobo [106] as motion planner to generate robot end effector trajectories starting from a fixed initial pose and ending to an target object grasping pose. The grasping pose is obtained from the grasping annotations used to train GraspNet [27]. We also randomized the object positions to generate more diverse layouts. Finally, we validate the trajectories in our framework and filter out invalid ones by controlling robots to follow the generated grasping trajectories. In the end, we successfully generated about 100k valid grasping trajectories.

L. GarmentLab

GarmentLab [68] is the first robotic manipulation benchmark for deformable object and garment manipulation. It integrates 10 categories of versatile garment assets and the total number of USD assets reaches 6k. To generate manipulation demonstrations, we directly roll out the trajectories provided by the official codebase in IsaacSim and collect the corresponding state information in a parallel process. Although the trajectory provided by the official codebase is limited and hard-coded, we further extend the number of demonstrations by applying different garments and textures, and all the demonstrations are

validated by the original success checker. Finally, we have successfully collected 6k trajectories.

M. UniDoorManip

UniDoorManip [63] provides an articulated manipulation environment reflecting different realistic door manipulation mechanisms, and a large-scale door dataset containing 6 door categories with hundreds of door bodies and handles stored in URDF format. We convert those door assets into USD format with physics APIs from IsaacSim and manually further verify the correctness of the joint-link relationship. Demonstrations are collected by directly rolling out the hard-coded trajectories in IsaacGym. We eventually collect about 1k successful legal demonstrations.

N. RLAfford

RLAfford [35] investigates the generalization ability of Deep Reinforcement Learning models on articulated object manipulation tasks with the presence of a computer vision model that is co-trained with it in an end-to-end manner. This work provided a dataset of articulated objects and 8 tasks for benchmarking.

In Roboverse, we have adapted 4 tasks (open cabinet, open drawer, close cabinet, close drawer) and in total 40k trajectories from RLAfford.

In the task adaptation, we included 40 articulated objects from the RLAfford dataset, and uses the same robot description file from RLAfford. Then we record 1000 trajectories for each object in its corresponding task.

The trajectory recording is achieved with several hooks we inserted into the original RLAfford codebase. The hooks are used to extract and maintain the recordings at different stages of simulation. We evaluated the released RLAfford model with hook-inserted scripts. In the initialization stage, objects and robots are initialized with randomization, their pose, and DoF information are recorded. For each simulation step, the DoF position information of objects and robots is recorded in the trajectories. In the end, for each object, a separate trajectory file of 1000 different trajectories is saved in the RoboVerse supported format.

O. LIBERO

LIBERO [64] manages data loading and task execution through a combination of INIT(initialization files), BDDL (Behavior Description Definition Language), and HDF5 datasets. Specifically, the initialization files define scene layouts, object properties, and basic task goals; the BDDL format captures semantic details and object affordances; and the HDF5 files store structured data such as object positions and robot actions for dynamic retrieval at runtime.

To migrate a LIBERO task into `MetaSim`, we parse the relevant BDDL file to identify which objects are involved and what type of manipulation context is required. Then we get the robot and object initial states from the INIT files, followed by the corresponding robot actions from the HDF5 dataset. These elements are combined into our `PKL` file format while also recording the participating objects in our `MetaCfg`. This process ensures that all necessary components of a LIBERO task, initial states, and action data, are fully translated and ready for execution in `MetaSim`.

We further augment the data by randomly sampling initial positions around each LIBERO demonstration, thus increasing the effective number of demos well beyond the original 50 per task. The spatial sampling range is dynamically chosen based on the task context and object dimensions, ensuring that the augmented configurations remain physically plausible.

XII. Task Generation

A. Robot & Object Generation Protocol

Our task generation pipeline (Fig. 17) begins with a *user prompt* describing the desired theme or constraints of a robotic task (e.g., "place the butter in the drawer and close it"). From here, the system proceeds in two main phases, mediated by large generative model calls:

1) **`call_gpt_to_generate_task()`: Conceptual Task Generation.** This initial function queries the model for a high-level task overview. It requests:
   - A *unique task name* (e.g., "ButterDrawerTask").
   - A short, *human-readable instruction* (e.g., "Place the butter in the drawer, then close the drawer.").
   - A candidate list of *robots and objects* to appear in the scenario, referencing an internal asset library (see below).

   The large generative model draws on its generative abilities to propose creative or contextually relevant tasks, while remaining loosely guided by the user prompt.[118, 117, 39, 139] As shown in Fig. 17, the model might retrieve a "drawer" asset from a different benchmark and a "butter" asset from a separate dataset, combining them into a single scene idea.
2) **`call_gpt_to_get_init_state()`: Physical Layout Refinement.** After receiving the conceptual description, we call the model again to specify `x,y` coordinates for each listed item. During this second phase, user can provide the prompts that include minimal bounding constraints (e.g., permissible table edges, object height) to help modelgenerate various initial states by few-shot learning.

Asset Library. To ground the large generative model's outputs in realistic data, we maintain an *asset library* (via JSON files) that describes each robot or object's core attributes (e.g., assets filepath, default rotation, size). The two core functions above selectively pull from this library.

Input and Output Format.

- *Input*: A user prompt (e.g., "create a tabletop scene with a random container and a snack food"). The pipeline loads relevant asset definitions and passes them to the large generative model calls.
- *Output*: A merged `init_state` or "initial state" dictionary capturing the initial state config needed for simulation: the chosen robot/object list, each item's final

`x,y,z` coordinate, and the textual instructions, as shown in the right half of Fig. 17.

XIII. TELEOPERATION

Ensuring flexible and intuitive remote operation is critical in robotic teleopration system, particularly when collecting large volumes of high quality data. In this work, we designed a suite of input methods to facilitate robot teleopration within the METASIM infrastructure. By supporting keyboard, DualSense Joystick, smartphone, and VR-based controls, our system accommodates varying user preferences and experimental needs. This section details our design rationale, implementation steps, and practical considerations for each control interface.

A. Keyboard

Keyboard input is an accessible method for controlling robots in simulation. Our implementation supports multi-key combinations for diagonal movement and enables full six-degree-of-freedom manipulation of the end effector. Translational movement follows the world coordinate frame (UP: +X, DOWN: -X, LEFT: +Y, RIGHT: -Y, 'e': +Z, 'd': -Z), while rotations in the local EE frame are controlled via 'q'/'w' (roll), 'a'/'s' (pitch), and 'z'/'x' (yaw). The spacebar toggles the gripper. To assist users and avoid hotkey conflicts with the simulation viewer, we provide an operation window displaying instructions using `pygame`. While efficient and hardware-independent, this method lacks 3D spatial representation, reducing user intuition. Additionally, Euler angle-based rotation control risks gimbal lock, potentially leading to loss of rotational degrees of freedom and failure in certain configurations.

B. Smartphone

Modern smartphones, equipped with advanced sensors and wireless communication, offer an ideal low-cost solution for intuitive teleoperation from any location. However, existing smartphone-based 6-DoF methods, such as those relying on accelerometers or vision-based Visual Inertial Odometry (VIO) systems (e.g., ARKit), suffer from instability due to sensor noise, low update rates, or weak visual features [40, 72, 73, 74]. Additionally, no open-source Android app exists for such implementations. To overcome these limitations, we adopt a hybrid approach: using smartphone orientation for motion control and on-screen buttons for precise translation. Unlike the keyboard interface, where roll, pitch, and yaw are controlled incrementally via discrete keypresses (i.e., delta orientation adjustments), the smartphone directly provides absolute orientation data in the form of quaternions. Quaternions, due to their compactness and immunity to gimbal lock, allow for a more stable and accurate representation of the smartphone's orientation in the world frame. As illustrated in Fig. 19, real-time data from the smartphone's inclination, rotation, and magnetic field sensors is fused to compute spatial orientation with ±5° accuracy at a frequency of 50 Hz. This data is transmitted via WebSocket, ensuring low-latency communication. The app interface features six buttons for translation control in the local coordinate system and two switches for toggling orientation updates and gripper control. Multi-touch input is supported to enable users to send combined control signals, such as simultaneous movement along multiple axes, improving control flexibility and efficiency. As shown in the Fig. 20 and Fig. 18, tilting the smartphone controls the gripper's orientation, while combining multi-touch signals from on-screen buttons enables precise and complex manipulation in 3D space. However, to mitigate magnetic interference, users should maintain a minimum distance of 10 cm from strong magnetic sources such as laptops and other electronic devices. This design optimizes resource utilization, providing a high-precision 6-DoF remote operation experience at minimal cost, rivaling professional-grade teleoperation systems.

Method	Simple		Language-conditioned Grasping		
	PickCube	MoveSliderLeft	Object Set 1	Object Set 2	Object Set 3
OpenVLA	40.0	45.0	46.0	33.3	14.4
Octo	50.0	30.0	42.0	14.4	2.2

TABLE VIII: **Vision-Language-Action (VLA) Model Results on ROBOVERSE Imitation Learning Benchmark.** Constrained with time and resources, we report VLA models' results on two simple tasks from ROBOVERSE and grasping tasks with diverse and challenging language instructions. We split 58 objects in GraspNet into three sets, each containing progressively more challenging objects based on their geometry.

C. Others

Beyond keyboard and smartphone controls, our system incorporates support for DualSense Joysticks and VR controllers. The DualSense joystick provides ergonomic advantages and high-fidelity analog inputs for nuanced velocity control, mapping triggers and joysticks seamlessly to robot motion. The VR interface enhances spatial awareness and precision by enabling natural gestures and directional cues for control.

Future work could extend VR capabilities by integrating haptic feedback to improve user immersion and task accuracy. Additionally, the modular design of our system facilitates the integration of emerging input devices with minimal development effort.

XIV. REAL2SIM TOOLSET FOR ASSET AND TASK GENERATION

A. Overview

The *Real2Sim* toolset, specifically *Video2URDF*, provides a systematic pipeline to reconstruct environment geometry and robotic assets from monocular video input. By leveraging advanced reconstruction techniques, this pipeline produces meshes and unified robot descriptions that can be used in simulation-based experiments. In doing so, it helps bridge the gap between real-world data and simulated environments, enabling more accurate and comprehensive benchmarking [67]

B. Components

1) Gaussian Splatting Reconstruction: The first step in the pipeline involves *Gaussian splatting [53]*, which converts

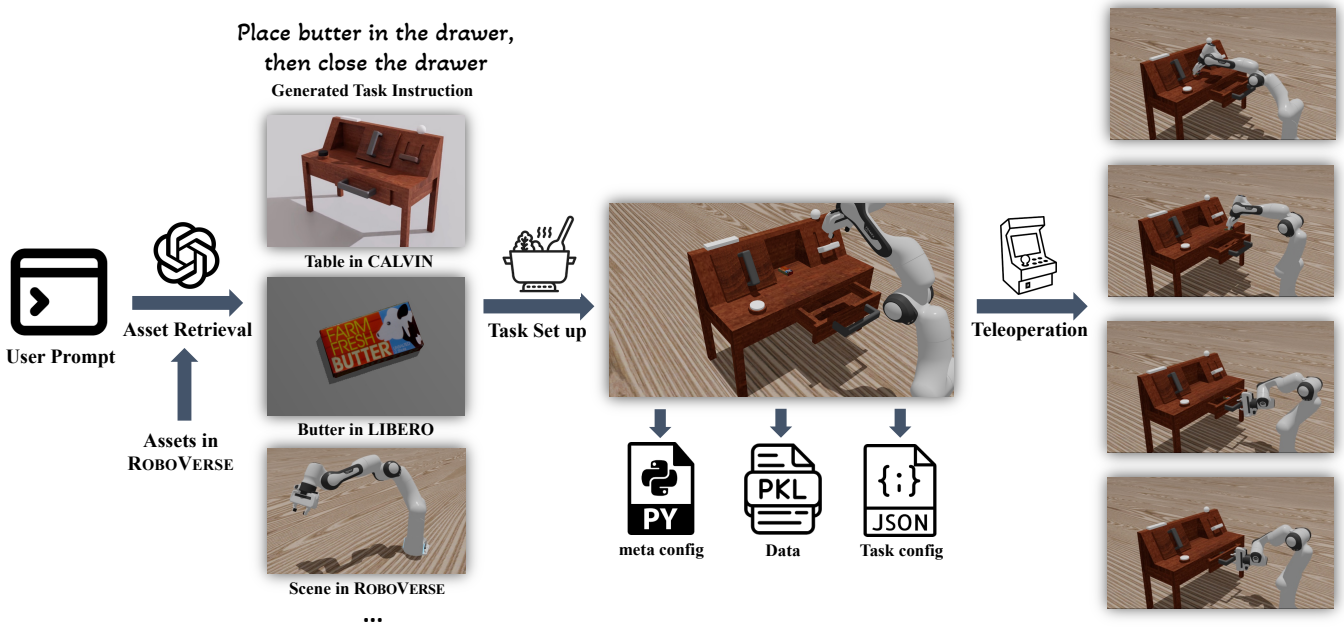


Fig. 17: Illustration of the two-phase generation protocol. A user prompt guides the LLM to propose an overall task and item list. The system then refines object positions and merges them into a final initial state.

monocular video frames into a set of Gaussian kernels for rendering [129]. This representation captures key scene features such as depth, color, and collision boundaries in a compact and efficient way. As a result, it provides a visually faithful preview of the scene and serves as an intermediate step before detailed mesh reconstruction.

2) Mesh Reconstruction: Once the high-level scene structure is represented by Gaussian splatting, we perform mesh reconstruction to obtain a more precise geometric model utilize tsdf extraction [132, 127, 128, 45]. This step recovers the meshes of:

- The *environment*, including rigid, immovable structures (e.g., a table).
- The *manipulatable object*, which is central to the task at hand.
- The *robotic arm and end effector*, assumed to have a deterministic configuration during real-to-sim and sim-to-real transitions.

We use a visual-language model (VLM) and available CAD design information to generate a unified URDF (or MJCF) description for these components. This division of the workspace follows the notion of `worldconfig` in `curobo [106]`, ensuring that each element of the scene (robot, object, environment) is cleanly separated and can be easily adapted or replaced as needed.

3) Loading the URDF into the Simulation Environment: After the URDF (or MJCF) files are generated, the final step is to import them into a simulator, such as *Mujoco* [110] in Roboverse. This allows researchers to configure tasks that accurately reflect real-world scenarios, forming a benchmark for training and evaluating robotic manipulation algorithms. The resulting simulated environment benefits from high-fidelity geometry and a consistent representation of the physical workspace.

4) Real-to-Sim boost Sim-to-Real Performance: We train model on our real2sim module compared with DexGraspNet[133], demonstrating 80% success rate compared to the 50% baseline from DexGraspNet. We use our real2sim assets in physics-based simulations that closely replicate real-world grasping conditions, enabling robust grasp execution. See Fig. 21 for visualization.

C. Limitations and Challenges.

While the Real2Sim pipeline effectively reconstructs most of the relevant geometry, it struggles with completely unseen meshes and complex material properties [138]. Furthermore, parameters such as friction and mass are inherently difficult to estimate purely from visual data, introducing uncertainties that may affect simulation fidelity. Despite these challenges, Real2Sim offers a powerful approach to rapidly generating simulation-ready assets for benchmarking in robotic manipulation tasks.

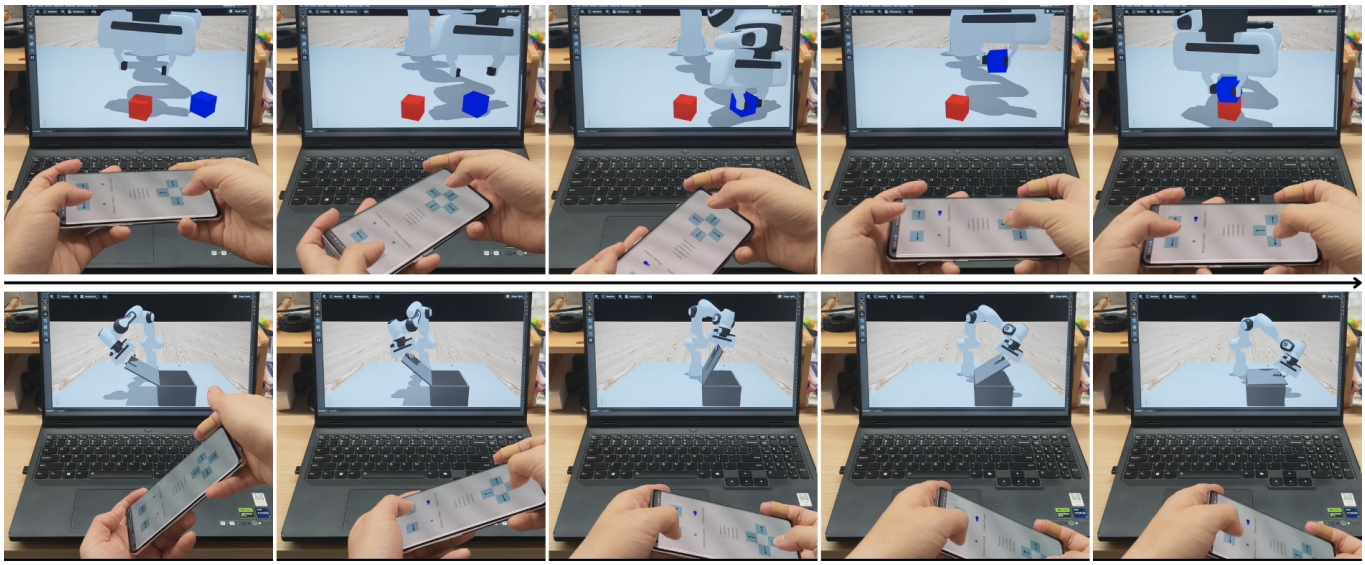


Fig. 18: Sequential demonstration of smartphone-based control for stack cube and close box tasks.

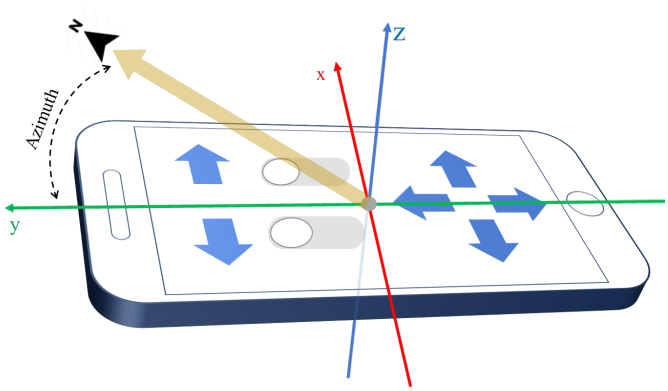


Fig. 19: Visualization of the smartphone's local coordinate system, world-frame orientation, and app functionality: six buttons control translation, and two switches toggle orientation control and gripper state.

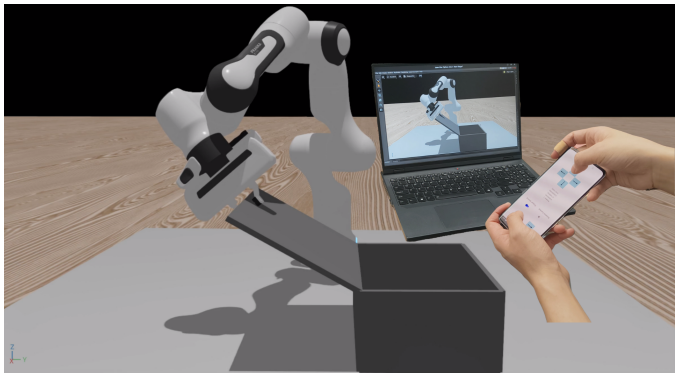


Fig. 20: The smartphone app enables 6-DoF control using orientation sensing and multi-touch buttons for translation commands, while the simulated robot's movements are visualized in real-time on the workstation.

XV. Domain Randomization

A. Scene Randomization

For scene randomization, we curate 3D simulatable scene assets from existing 3D scene datasets [30, 36, 21, 49]. Specifically, we convert all assets to the USD format for integration. Additionally, we employ the articulated scene generation method PhyScene [126] to create realistic scenes with articulated objects and mix the generated room-level scenes with house-level 3D scenes like ProcTHOR for greater diversity. We replay demonstrations in these scenes by selecting surfaces (*e.g.*, floors, tables) that provide sufficient workspace, guided by heuristic-based spatial constraints, following [36].

B. Visual Material Randomization

It's optinal to attach random visual material to object surfaces. Visual materials are randomly selected from a curated subset of ARNOLD [36] and vMaterials [83], providing more the 300 high-quality visual material candidates. Additionally, user can also randomize the reflection properties of a given visual material, by setting roughness, specular, and metallic to random number between 0 and 1.

C. Light Randomization

Two lighting configurations are supported: distant light and cylinder light arrays. For distant lighting, the polar angle of the light source is randomized. For cylinder lighting, a randomly generated $n \times m$ matrix of cylinder lights, each with a randomized size, is added at a fixed height above the agents. In both configurations, the intensity and color temperature of the lights are randomized within physically plausible ranges.

D. Camera Randomization

A total of 59 candidate camera poses are carefully selected, with the majority oriented to face the robot directly and a

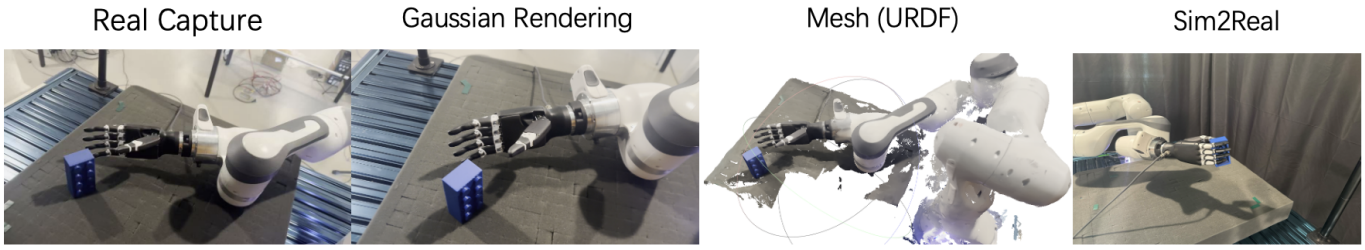


Fig. 21: Visualization of our real2sim pipeline for robotic grasping.

smaller subset positioned at side-facing angles.

XVI. Navigation and Locomotoin Tasks

A. Navigation Tasks

To integrate vision-and-language navigation into IsaacSim, we first correct the error-containing instructions by refining incorrect punctuation and grammar using ChatGPT. Next, we validate the ground truth trajectory by sweeping the robot's 3D model (based on the ground truth trajectory) through the scene. The trajectory is deemed invalid if collisions occur between the robot and the scene. Additionally, we adopt the same evaluation metrics as VLN-CE [58]. For controlling the robot, we provide two different types of mobile embodiments, including a Unitree Go2 robot dog and a JetBot wheeled robot, making our task suitable for a variety of policies (with different navigation capabilities).

B. Humanoid Tasks

We migrated the data samples from the Humanoid-X dataset [76], and re-implemented the inference pipeline of UH-1 [76] in our framework. We use the Unitree-H1-2 humanoid robot as the simulated embodiment and set up the locomotion and humanoid pose control task in our framework. The humanoid pose control task is to control the humanoid robot to follow some human poses while maintaining its stability on the ground. The demonstrated poses in our framework include arms crossing, boxing, dancing, left and right punch, playing violin, playing guitar, praying, waving to a friend, *etc.* Our pretrained policy can successfully follow the demonstrated pose to control a humanoid robot while maintaining stable locomotion in IssacGym, and also obtain a decent performance in IssacLab. The humanoid environment and task configurations are highly flexible and scalable, and we are able to support more humanoid pose control tasks from Humanoid-X without modifying the infrastructure.

C. HumanoidBench

HumanoidBench[101] is a high-dimensional simulated benchmark designed to accelerate research in humanoid robot learning, focusing on whole-body locomotion and manipulation tasks. The benchmark features a humanoid robot equipped with dexterous hands, enabling a wide range of complex interactions in human-like environments.

Tasks and Assets: We migrate three fundamental locomotion tasks: run, walk, and stand. These tasks are designed to test the robot's ability to maintain balance, achieve forward motion, and stabilize in a standing position. The primary robot model used is the Unitree H1, augmented with two dexterous Shadow Hands, though the environment supports other humanoid models such as Unitree G1 and Agility Robotics Digit.

Demonstrations: While HumanoidBench does not provide pre-collected demonstrations, it supports the use of reinforcement learning algorithms to generate task-specific policies. The benchmark is designed to facilitate learning from scratch, with dense and sparse reward structures to guide the learning process.

Success Checkers: Each task in HumanoidBench is equipped with a success checker that evaluates task completion based on predefined criteria. For example, in the walk task, success is determined by the robot's ability to maintain a forward velocity of 1 m/s without falling, while in the stand task, success is measured by the robot's ability to maintain a stable upright posture for a specified duration.

Experiment and Results: We trained the walk, stand, and run tasks in both the RoboVerse MuJoCo and IsaacLab simulators using the PPO and TD-MPC2[41, 42] algorithms, and compared the results with the HumanoidBench baseline based on the original MuJoCo environment. As shown in Figure23 and Figure24, the training curves from the RoboVerse MuJoCo simulator eventually converged and approached the performance of HumanoidBench, validating the feasibility of the RoboVerse reinforcement learning infrastructure. Additionally, we trained the same tasks in the RoboVerse IsaacLab simulator with identical configurations. While training efficiency in IsaacLab was comparatively lower under non-parallelized settings (to maintain configuration consistency), it still demonstrated a clear upward trend in reward accumulation. This confirms the rapid migration capability of the MetaSim framework and highlights its potential to enable sim-to-sim learning while leveraging the strengths of different simulators, such as IsaacLab's support for GPU-accelerated large-scale parallel training.

XVII. RoboVerse Benchmark Set up Details

A. Generalization Levels

To systematically evaluate the generalization capability of a robot policy, we establish a benchmark based on a carefully curated asset set designed for domain randomization. This asset set encompasses a diverse range of environmental factors, including materials, textures, lighting conditions, scene configurations, and camera perspectives. By leveraging this set, we assess how well different policies generalize to unseen

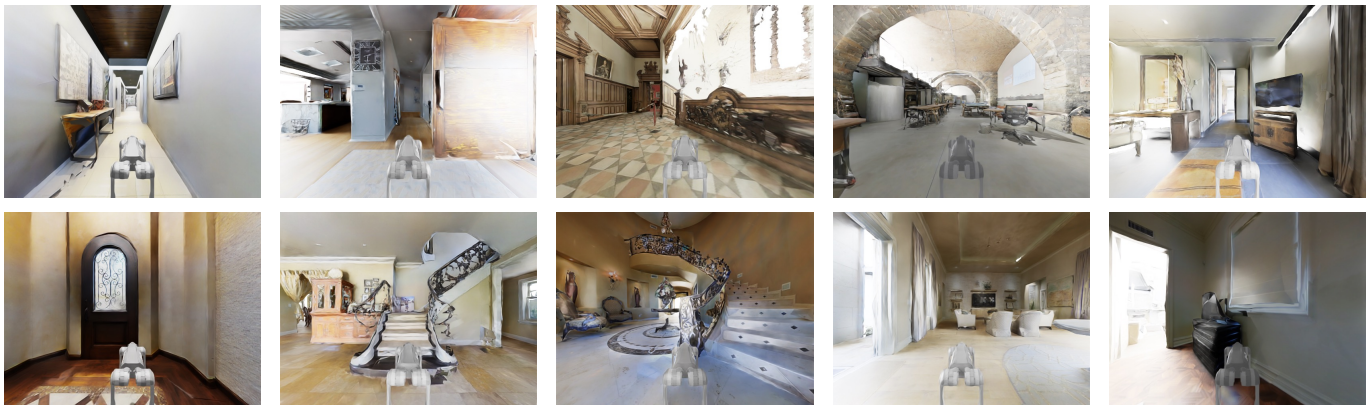


Fig. 22: Navigation gallery. We deploy the Unitree Go2 robot within Matterport 3D environments, primarily integrated with RoboVerse Isaac Lab branch. The robot is tasked with navigating the environment based on provided instructions.

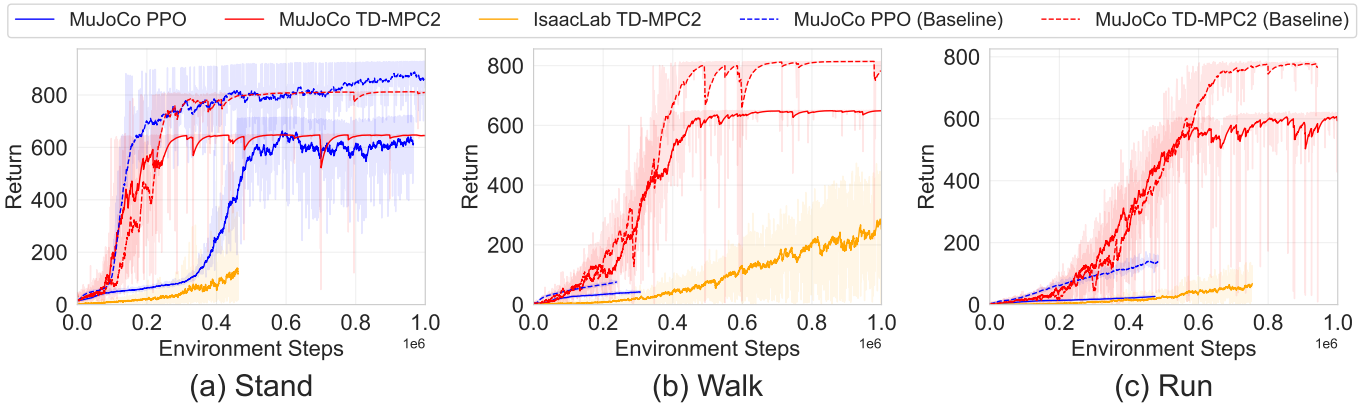


Fig. 23: **Learning curves of RL algorithms on HumanoidBench task migratation**: We also run PPO in the IsaacLab simulator in RoboVerse, but it is not visible in the plot since it only achieves very low returns.

conditions. Specifically, we split the available assets into a 9:1 ratio for training and testing, ensuring that the testing environment contains novel variations not encountered during training. Below, we detail the key components of this domain randomization setup:

- **Table, Ground, and Wall.** In tasks where a predefined scene is absent, we incorporate **walls (and ceilings)** to introduce structural complexity. Additionally, customizable tables are included for tasks requiring tabletop interactions. The visual materials applied to these elements are randomly sampled from a carefully curated subset of ARNOLD [36] and vMaterials [83], ensuring a diverse range of appearances. The table features approximately 300 distinct material options, while both the wall and ground have around 150 material choices each. This variation enhances the robustness of the learned policy by exposing the model to a wide spectrum of surface appearances and textures.
- **Lighting Conditions.** We introduce two distinct lighting scenarios: distant lighting and cylinder light arrays, each designed to test the adaptability of the learned policy to different illumination conditions.
  - **Distant Light:** The polar angle of the light source is randomized within a predefined range, influencing the way shadows and reflections appear in the scene.
  - **Cylinder Light Arrays:** A randomized $n \times m$ matrix of cylinder lights, varying in size and intensity, is placed at a fixed height above the agent.

  In both configurations, light intensity and color temperature are randomly varied within reasonable limits to ensure that the model encounters a broad range of lighting effects.
- **Camera Poses.** To further evaluate the robustness of visual perception, we carefully select 59 candidate camera poses, strategically positioned to provide diverse viewpoints. The majority of these cameras are oriented directly towards the robot, ensuring consistent frontal perspectives, while a subset is placed at side-facing angles to introduce additional viewpoint variability.
- **Reflection Properties.** To simulate the wide range of reflective surfaces encountered in real-world environments, we randomize key material reflection properties, including roughness, specular intensity, and metallic characteristics.

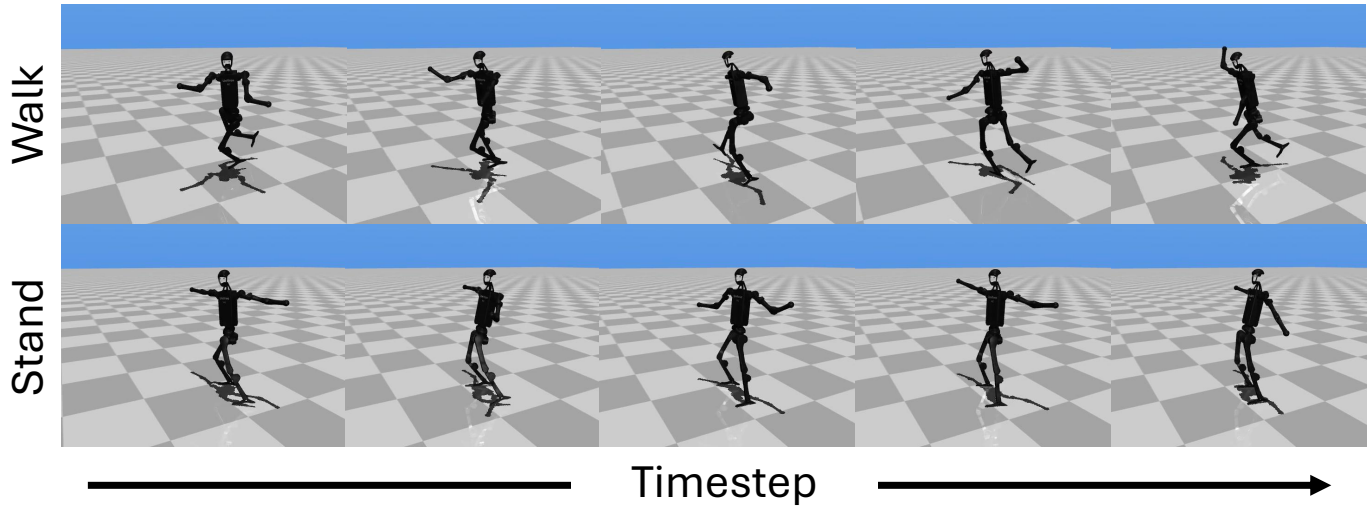


Fig. 24: Demonstration of TD-MPC2 policys trained in the RoboVerse MuJoCo simulator on the Walk and Stand tasks migrated from the HumanoidBench benchmark

These properties are adjusted within reasonable physical ranges to ensure that the robot policy learns to handle various levels of surface reflectivity.

By integrating these domain randomization techniques into our benchmark, we create a controlled yet diverse testing environment that challenges the generalization ability of different robot policies. This setup ensures that trained policies are not merely overfitting to a limited set of conditions but are instead capable of adapting to a broader range of real-world variations.

B. RoboVerse Benchmark Protocol

We rigorously design a training and evaluation protocol to ensure a structured and reliable assessment of the policy's performance. Given the training data, the policy learns to imitate the demonstrated behavior. For evaluation, we provide a standardized API that enables systematic assessment. As mentioned earlier, the training and evaluation follow a 9:1 ratio, ensuring that the policy is tested on novel scenarios not encountered during training.

XVIII. Policy Training Details

A. Implementation Details

For specialist models, we train from scratch with action in 9-dim robot joint state space. Diffusion Policy [13] is implemented based on its original framework. We search several key hyperparameters, including observation and prediction length, to optimize performance for our tasks. ACT [137] is implemented with the original architecture and hyperparameters, except that the batch size has been increased to 512, with learning rate correspondingly enlarged to $1e-4$ to accelerate convergence. We train ACT on one A100 GPU for 2000 epochs and evaluate with the best checkpoints on the validation set.

For generalist models, the action is pre-processed into delta end-effector position space from absolute end-effector position space, and the gripper action is binarized to $\{0, +1\}$. Owing to the lack of time and resources, we are only able to fine-tune the generalist models in the single-task setting. For each task, OpenVLA [56] is LoRA [44] fine-tuned (rank= 32) with 8 A100 GPU under official settings to convergence and reaches over 95% action token accuracy as proposed by Kim et al. [56] during the training stage. During evaluations, we employ Curobo [105] as the inverse-kinematics solver to transform the action to robot joint state space.

B. Diffusion Policy

We implemented the training and validation code for *Diffusion Policy* based on the requirements of our tasks and relevant research papers.

Modeling Diffusion Policy as Denoising Diffusion Probabilistic Models (DDPMs), we train a noise predictor network:

$$\widehat{\epsilon^k} = \epsilon_\theta \left(a^k, s, k\right) \tag{1}$$

that takes in noisy actions a^k, current observations s, and denoising iterations k and predicts the noise $\widehat{\epsilon^k}$.

As for observation s, We use ResNet18 to extract the features of scene images f_{img} and use 3-layer MLP to extract the features of robot joint states f_{robot}. f_{img} concatenating with f_{robot} is just the conditioning input for Diffusion Policy.

During training, we randomly choose a denoising step k and sample noise ϵ^k added to the unmodified sample a^0. Our training loss is the difference between ϵ^k and predicted noise:

$$L_{DP} = MSELoss(\epsilon^k, \widehat{\epsilon^k}) \tag{2}$$

During inference time, our policy starts from random actions a^K and denoises for K steps to obtain the final action predictions. At each step, the action is updated following:

$$a^{k-1} = \alpha \left(a^k - \gamma \epsilon_\theta \left(a^k, s, k\right) + \mathcal{N}\left(0, \sigma^2 I\right)\right) \tag{3}$$

, where α, β and γ are hyperparameters.

XIX. World Model Details

A. Methodology

We adopt a video generation framework based on Latte[70]—a transformer-driven latent diffusion model equipped with an efficient spatial-temporal attention mechanism. For action conditioning, we use frame-level Adaptive Layer Normalization[88] (AdaLN), following insights from IRASim[140] that show more precise control of the gripper with frame-level conditioning compared to video-level conditioning.

In the forward pass, raw video frames are encoded using a frozen autoencoder from Stable Diffusion[89]. The first frame serves as the initial condition, while noise is introduced into the latent representation of subsequent frames during training. Both the noise schedule and action conditions (gripper states with either Cartesian position plus orientation or joint position) are encoded by separate MLPs into latent space and then added together.

These noisy latent frames are then fed into a transformer composed of alternating spatial and temporal attention blocks, where action conditions are applied at each frame via AdaLN. For inference, we employ DDIM[104] as a denoising scheduler, using 200 sampling steps.

B. Data Preparation

The DROID[54] dataset's episodes typically last from 120 to 360 frames. To amplify motion, we skip every 6 frames, effectively reducing the frame rate to 4 fps with sequence lengths from 20 to 60. In the RoboVerse simulation, we adjust the control frequency so that most episodes span 20 to 60 frames, mirroring the number of frames of DROID in one episode. We filter out any sequence shorter than 20 or longer than 60 frames, resulting in about 50,000 unique episodes from DROID.

We only generate 50,000 unique RoboVerse episodes due to time and resource constraints. The full-scale RoboVerse is planned to train more capable world models in future works.

We exclude the gripper camera view because the model struggles with drastic camera pose changes, which leads to poor frame generation quality. Since we consider left and right camera views as separate samples, each dataset effectively doubles to 100,000 samples.

C. Experiments

Our experiments involve training three datasets, DROID-50K, RoboVerse-50K, and DROID-RoboVerse-100K, on 8 NVIDIA H100 GPUs. We use a spatial resolution of 240×320 and sequences of 16 frames per episode. Starting with a model of 100M parameters and a batch size of 16, training converges at around 100K steps on RoboVerse and 200K steps on DROID.

We first compare Cartesian position plus orientation to joint positions as action conditions and find that using joint positions as action conditions yields more precise gripper movement control in frame generation, as shown in Fig.26. We believe it is due to joint positions being less ambiguous than Cartesian position plus orientation as the robot states representation.

However, generation quality remains suboptimal when training on the DROID-50K or DROID-RoboVerse-100K datasets and validating on DROID samples due to the complexity of DROID scenes. Scaling the model to 500M parameters and reducing the batch size to 8 leads to better preservation of object geometry, as does the prediction of robot arm movement.

As discussed in the main paper, although the larger model trained on DROID-RoboVerse-100K shows an improved understanding of object shapes in DROID samples compared to the model trained on DROID-50K, it still struggles with intricate real-world physics. In contrast, training with RoboVerse-50K or DROID-RoboVerse-100K and validating on RoboVerse scenes produces more physically and geometrically consistent predictions as shown in Fig. 27.

We believe it is because RoboVerse offers cleaner backgrounds, more comprehensive views of the robotic arm, and the implementation of domain randomization and augmentation. By comparison, many DROID frames contain cluttered backgrounds or incomplete arm visibility, creating challenges for learning robust temporal dynamics from raw pixels.

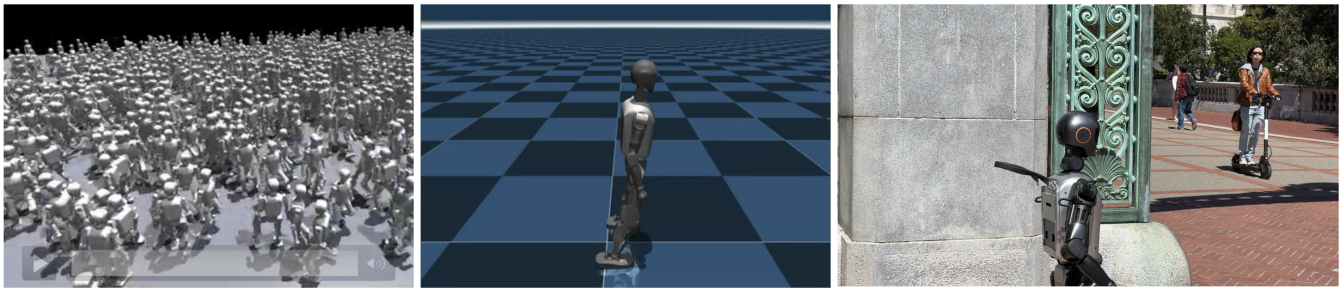


Fig. 25: **Visualization of Sim-to-Sim-to-Real Experiments.**

Fig. 26: Visualization of ground truth and predicted frames by models conditioned on cartesian position (plus orientation) and joint position.

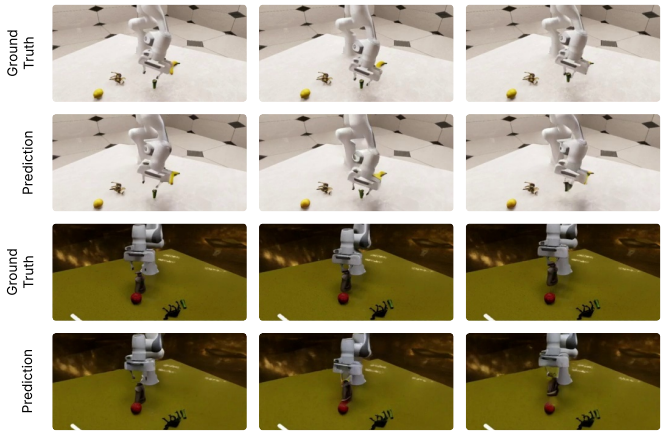


Fig. 27: Visualization of ground truth and predicted frames by models trained on DROID-RoboVerse dataset.