Survey on Physics Engines, Simulation Frameworks, and Benchmarks for Robot Learning

Alexandre Brown[*1,2] **Vivek Shankar Varadharajan**[*2,3] **Nico Bohlinger**[*4]

Elham Daneshmand[*2,5] **Maeva Guerrier**[*2,3] **Reginald McLean**[*6]

Michael Przystupa[*7,8] **Ozgur Aslan**[1,2] **Artur Kuramshin**[1,2] **Charlie Gauthier**[1,2]

Miguel Saavedra-Ruiz[1,2] **Liam Paull**[1,2] **Jan Peters**[4,9,10,11] **Giovanni Beltrame**[2,3]

Glen Berseth[1,2]

Abstract: Robot learning research is rapidly advancing and relies heavily on simulation tools—physics engines, simulation frameworks, and benchmarks—to develop and evaluate algorithms faster, safer, and more cost-effectively than using physical hardware alone. However, the recent explosion of such tools has created a fragmented landscape, making it difficult for researchers to navigate options or forcing them to constantly re-implement baselines. This challenge is amplified by blurry terminology, where terms like "simulator" are used ambiguously. To address this, our survey makes four key contributions: (1) We introduce a formal Robot Learning Problem definition to standardize terminology and categorize existing tools accordingly. (2) We provide a comprehensive overview of the features, strengths, and weaknesses of current simulation tools, based on literature and hands-on evaluation. (3) We offer guidelines for tool selection tailored to major robot learning domains: vision manipulation, locomotion, and navigation. (4) We introduce an accompanying open, living repository to track updates in the field. By systematically structuring the simulation landscape, we aim to lower the entry barrier for newcomers, reduce redundant engineering for experts, and accelerate progress toward reproducible, general-purpose robot learning systems. Project webpage: https://robotlearningproblem.github.io/Robot-Learning-Problem/

1 Introduction

Robot learning—the field of addressing robot control tasks using problem formulations and tools from machine learning—is advancing at an unprecedented pace. Central to this progress are simulation tools: virtual physics engines, simulation frameworks and benchmarks that let researchers prototype, debug, and evaluate control policies and learning algorithms orders of magnitude faster, safer, and more cost-effective than learning only on physical hardware. The ability to generate vast amounts of data and test hypotheses quickly has significantly accelerated progress in the field [1, 2, 3, 4]. Well-designed simulation tools can provide the robot learning community with rich environments for reinforcement learning [5, 6], general benchmarks for imitation learning [7, 8], enable large-scale and diverse data generation [9, 10], and finally enable robust sim-to-real transfer for any kind of robot platform to unstructured real-world environments [3, 11].

[*]These authors contributed equally to this work. Author order randomized. Université de Montréal[1], Mila - Québec AI Institute[2], Polytechnique Montréal[3], Technical University of Darmstadt[4], McGill University[5], Toronto Metropolitan University[6], University of Alberta[7], Alberta Machine Intelligence Institute (Amii)[8], German Research Center for AI (DFKI)[9], Hessian.AI[10], Robotics Institute Germany (RIG)[11].

In recent years, the field of robot learning has witnessed an explosion of physics engines (e.g. **MuJoCo** [12], **PhysX** [13]), robot-learning-centric simulation frameworks (e.g. **Isaac Sim/Lab** [14, 15], **SAPIEN** [16], **Robosuite** [17], **Maniskill** [18]), and task benchmarks (e.g. **RLBench** [5], **Meta-World** [19], **LIBERO** [20]). While this growth has fueled innovation, it has also created a complex and often fragmented landscape. Researchers face increasing difficulty in navigating the available options, understanding their specific strengths and weaknesses, and selecting the most suitable tools for their particular research questions. Without a clear map, newcomers struggle to choose the right toolchain given their class of robot learning problem, and seasoned researchers spend significant effort to constantly re-implement baselines or replicate prior work. This challenge is compounded by inconsistent terminology within the community. Terms like "simulator" are often used ambiguously, sometimes referring to the underlying physics engine, other times to a complete simulation framework, or even to a specific suite of benchmarks.

To address these challenges and provide clarity to the field, this survey makes the following key contributions: **Standardised terminology** by building on a formal definition of the Robot Learning Problem (RLP), we disambiguate the terms physics engine, simulation framework, and benchmark, and we catalogue over simulation tools accordingly. Providing **guidelines for tool selection** from a comprehensive feature overview to help researchers select the most appropriate tools for their specific research needs. In particular, we analyze the major robot learning classes (vision manipulation, locomotion, navigation) and recommend guidelines for selecting the right tools for each class. Keep an **open, living resource** to keep pace with the field, we release an open-source repository[1] that keeps track of new releases and features of the relevant tools. This repository will be continuously updated to reflect the latest advancements in the field.

By providing a comprehensive overview of the current state of simulation tools, we aim to demystify the current ecosystem of simulation tools for robot learning and call to action to synergize the tools created by the community to work together to improve reproducibility and support to lift the community's aims to ultimately accelerate progress toward general-purpose robot learning systems.

2 Related Work

The landscape of simulation in robotics and machine learning has been surveyed from various perspectives. However, existing surveys often differ in scope and focus. Several works have focused on the limitations of physics engines for learning research in general [21, 22, 23, 24], where robotics is an application of these results. Robotics-specific literature reviews have only considered robot learning as *one* of many robotics research domains where software tool selection for experiments are an important consideration [25, 26]. Although we focus on software tools, we view them in conjunction with our formalization of the robot learning problem. Literature reviews that focus more specifically on robot learning have been algorithmic-focused. General reviews have included discussing general learning formulations such as reinforcement learning [27, 28], learning from demonstrations [29], or sim-to-real transfer [30, 31]. Other algorithmic surveys are organized by robot platforms such as manipulators [32], humanoids [33, 34], and mobile navigation robots [35, 36, 37]. The work of [38] motivates the importance of better simulators for learning, but is not a comprehensive review of existing tools like our paper. Our review focuses solely on robot learning software tools that can be used to compare algorithms, such as available benchmarks and simulation frameworks, or the underlying physics engines.

3 Robot Learning Problem

When attempting to train a robot policy, regardless of the method used for learning (i.e. Imitation Learning, Behavioral Cloning, Reinforcement Learning, etc.), the user makes many decisions that are relevant to the problem they are trying to solve. For example, if the user wants to train a policy to control a single robot arm for a vision manipulation task then this robot learning problem is very

[1] https://github.com/robotlearningproblem/Robot-Learning-Problem

different from attempting to train a single model that can control multiple locomotion-based robots across various terrains. In order to define this general set of problems, we propose the Robot Learning Problem (RLP). A robotics practitioners face a multi-layered decision problem that goes beyond traditional Markov Decision Problem formulation, they must simultaneously choose their robot platform, simulation environment, task specification, and learning approach. The RLP framework attempts to formalize these dependencies. The RLP is defined as the tuple:

$$< \mathcal{M}, \mathcal{W}, \mathcal{V}, \mathcal{A}, \mathcal{O}, \mathcal{R} > \tag{1}$$

where $\mathcal{M}$ is the set of all robot embodiments that can be used and $\mathcal{R}$ is the set of reward functions in an RLP. The reward function can also be used to model imitation learning tasks or behavior cloning. Next, $\mathcal{W}$ consists of the world parameters that are separate from a given embodiment m and more specific to the scene. Following is $\mathcal{V}$ the set of all possible physics engines.

With the embodiment m, reward r, world parameters w, and v physics engine chosen, the user can now define some of the components that affect the learning process of a RLP[2]. These components are the set of all possible action spaces $\mathcal{A}$, and the set of possible observation spaces $\mathcal{O}$. For more details on how action and state spaces depend on the embodiment m see Appendix B.1.

The RLP formulation is flexible enough to handle many different scenarios in robotics. For example, the set of embodiments $\mathcal{M}$ can include embodiments of various types (different base robot arms, different legged robots, etc), while the set of world parameters $\mathcal{W}$ encompasses the parameters that allow for the simulation to be as accurate as possible compared to the real world.

4 Simulation for Robot Learning

In this section, we go into the specifics of the instantiation of the RLP. Following prior work that distinguishes between software tools, such as physics engine from robot simulation framework [26, 39], rather than grouping them [25], we divide this section into physics engines, frameworks, and robot learning benchmarks. In many cases different terms such as physics engine, framework, or benchmark are used interchangeably in the literature. For example, the Mujoco physics engine is used in many different works while there are also environments within the Mujoco name-space (HalfCheetah, Hopper, Walker and Ant). These components are often determined from context, which can lead to confusion if a user does not have the ability to fully grasp the context. We use the RLP to create a clear delineation between these components, providing an effective method of describing different robotics problem settings. Given our formal definition of the RLP, in this section we discuss the challenges unique to several major types of robot control problems. As part of the discussion, we propose precise definitions for the following terms.

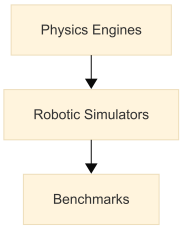

Figure 1: The levels of simulation in the RLP.

Physics Engine: A physics engine v is a computational software package that approximates real-world physical interactions between objects. These engines are responsible for simulating rigid-body dynamics, collision detection, contact mechanics, and provide basic support for visualization, navigation primitives, and perception interfaces.

Simulation Frameworks: are software that enables the simulation of robotic morphologies m, multiple observation types $o \subset \mathcal{O}$, and tasks $t \sim p(\cdot|m)$ by integrating a physics engine and providing a high-level modular interface for robotics, even providing support for making basic object in the world. These simulation frameworks can be used to define any number of Markov Decision Processes (MDP) representing different RLPs across diverse robotics domains.

Benchmark: A specific subset of rewards $r \subset \mathcal{R}$ typically with standardized reward functions, termination conditions, and success metrics, to facilitate the evaluation and comparison of learning algorithms within a simulation framework and provide a Gymnasium [40] style API for learning.

[2]We intentionally omit the learning of a set of parameters θ to limit the scope of our work to Physics Engines, Simulation Frameworks, and Benchmarks.

We explicitly provide these definitions because the exact definitions for these terms have been muddied in recent years, likely due to the lack of standard definitions of the terms. Therefore, we use these definitions throughout this work and call on our fellow researchers to adopt these definitions as well in order to disambiguate further research. An important acknowledgment is that the line between these classifications can be blurred at times. Figure 2 provides a high-level summary of commonly used systems in robot learning, grouped by our proposed definition. Arrows in the diagram indicate primary dependency relationships. The figure omits secondary dependencies for clarity, and should be interpreted as an illustrative, not exhaustive, map of the current simulation landscape, still there is a very prominate number of arrows coming out of open sources physics engines which indicates the importance of good software and community support and its impact on research growth.

Physics engines are implementations of mathematical models for modeling dynamic simulations. For each physics engine within the set of all possible physics engines $\mathcal{V}$, there are different features that each engine supports that may shape a user to choose that engine. Mathematical approximations of real world phenomena and implementation choices affect an engine's speed, accuracy, and ease of use. The beginning of this section defines qualitative considerations for choosing a physics engine $v \sim p(\mathcal{V})$. In Table 1 we highlight a full set of important physics engine characteristics for locomotion, navigation, and manipulation problems.

Speed and Parallelization: RLPs often require vast amounts of data that are generated most efficiently through thousands of parallel environments [3, 41]. Engines capable of high-speed simulation, especially those leveraging GPU acceleration for massive parallelization (like **MJX** [42], **PhysX** [13], and **Genesis** [43]), are highly desirable as they allow for faster training cycles and experiment iteration. While CPU-based engines like **MuJoCo** or **Bullet** offer high accuracy, their speed and parallelization capabilities are limited compared to GPU-based alternatives.

Solver Stability and Accuracy: Solvers use numerical methods to enforce physical constraints like joints and contacts. Accuracy reflects how well the solver computes the outcomes of the chosen contact dynamics model, while stability prevents errors that corrupt data and slow simulations. Engines utilize different solver algorithms offering distinct speed, accuracy, and stability trade-offs. Physics engines frequently used for complex robotics, like **MuJoCo** (known for stable solvers [23]), **PhysX** (robust, GPU-accelerated), **DART** (accuracy-focused solvers), and **Chrono** (high-fidelity options), generally provide robust solutions. Engines like **ODE** face more stability challenges [23]. Accurate and stable solvers are critical, especially for manipulation tasks, as they require complex, sensitive contacts which demand higher solver performance.[44].

Certain physics engines support manipulating world parameters w that affect the simulation dynamics. **Domain randomization** commonly includes variations in surface friction, terrain roughness, robot mass distribution, and random external disturbances within the chosen world parameters w [45, 46].

4.0.1 Locomotion considerations

The field of robot locomotion has seen rapid progress in capabilities in recent years, and most physics engines are capable of simulating the basic dynamics of locomotion necessary for training locomotion policies. The generation of different terrains with height fields, local sensing, and simple contact models for feet-ground interactions are present in all engines mentioned in Table 1. Practitioners gravitate to GPU-accelerated engines for learning agile skills, like running and parkour climbing, and engines with high solver accuracy for precise and delicate tasks, like walking on stepping stones.

4.0.2 Manipulation Considerations

Next, we focus on some of the core engine characteristics that are critical when selecting a physics engine v for for manipulation, with a focus on generating useful data and enabling effective learning.

Contact Dynamics Fidelity: Realistic contact simulation is essential for learning manipulation tasks and minimizing the sim-to-real gap [47]. Inaccurate models can hinder real-world performance. **MuJoCo** is particularly known for accurate contact modeling accuracy and sim-to-real success.

PhysX is also another strong choice, offering robust, high-performance physics suitable for complex robotics and **Chrono** [48] offers detailed multi-physics capabilities for high-fidelity scenarios.

Geometry Representation Speed and Stability: Physics engines represent objects using collision shapes like simple primitives (eg: box, sphere), convex hulls, or detailed non-convex triangle meshes. **MuJoCo** and **MJX** focus on optimized primitives and convex meshes. **PhysX**, **Bullet**, **DART**, and **Chrono** offer broader support for non-convex meshes, which accurately represent complex object details vital for manipulation. GPU-accelerated engines like **PhysX** [49, 18] and **MJX** [41] offer high performance when using more complex geometry. We recommend verifying if the task requires geometric detail of non-convex meshes or if faster convex approximations suffice.

4.0.3 Navigation considerations

Navigation planning refers to the process of computing a feasible and efficient path for a robot system. This typically requires the robot to perceive its surroundings, localize itself within a map, and reason about future states to minimize a given objective. From the perspective of physics engines, several key attributes are critical for navigation-focused simulations.

Sensor support: In navigation tasks, the observation space o is generally limited to observation spaces that provide realistic sensor data (eg, Camera, Lidar, RGB-D, IMU, GPS). The selection of a physics engine v must allow for generating realistic sensor data as it is vital for tasks like simultaneous localisation and mapping (SLAM), obstacle avoidance, and motion planning.

Scalability for Multi-Agent Systems: In fleet or swarm scenarios, the physics engine v must efficiently handle multiple agents interacting in the same environment without degrading performance or accuracy.

	Dynamics									
Physics Engine	Accuracy	GPU	6-DoF Configurable Joint	Soft Body Support	Contact Solver Characteristics	Docs	Open Source	Sensor Support	Multi-agent Scalability	Physics Params Support
Mujoco [12]	High	✗	✗	✗	No internal forces, Robust, Convergence guarantees	Comprehensive	✔	Limited (Camera, no LiDAR)	Limited	High
MJX [42]	Medium	✔	✗	✗	No internal forces, Robust, Convergence guarantees	Comprehensive	✔	Limited (inherits MuJoCo's model)	Support with JAX	Medium
Bullet [50]	Medium	∼	✔	✔	Hard contacts	Well-documented	✔	Moderate	CPU bottlenecks	Medium
Havok[51]	Low	✗	✔	✔	Not strict convergence	Limited documentation	✗	Very limited	Limited	Low
ODE [52]	Low	✗	✗	✗	Hard contacts	Partially documented	✔	Very limited	Limited	Medium
PhysX [13]	Medium	✔	✔	✔	Hard contacts	Well-documented	✔	Moderate	High	Medium
Dart [53]	High	✗	✗	✗	Hard contacts	Well-documented	✔	Good	Good	High
Genesis [43]	Medium	✔	✗	✔		Partially documented	✔	Limited	Moderate	Medium
Chrono [48]	High	✔	✔	✔	Hard contacts	Well-documented	✔	Comprehensive	Moderate	High

Table 1: Comparison of features across physics engines typically used in Robot Learning

4.1 Simulation Frameworks

Simulation frameworks build on physics engines by packaging them within end-to-end toolboxes tailored to RLPs. While physics engines focus on low-level simulation details, frameworks integrate them with high-level modules for robot modeling, environment construction, sensor emulation, task specification, and experiment management. Frameworks enable researchers to define robot learning tasks for many domains without delving into the intricacies of physics solver configurations or other low-level details. An overview of simulation framework characteristics in Table 2.

Domain Randomization Curriculum: The physics engine supports the option of utilizing domain randomization. The framework should support various types of curricula to control the domain randomization. The option to edit the curricula allows researchers to investigate these algorithms to improve performance. **Isaac Lab** and **Maniskill** have custom APIs and allow changing some physics parameters during training, as they have to be fixed when the simulation starts. **MuJoCo Playground** [41] (through **MJX**) support changing every parameter at any time, crucially also the terrain heightfield. The more flexible the domain randomization, the faster training can be accelerated.

Sim-to-Real: A critical challenge in robot learning is transferring policies trained in simulation to real-world settings—commonly referred to as sim-to-real transfer. Evaluating the robustness of these

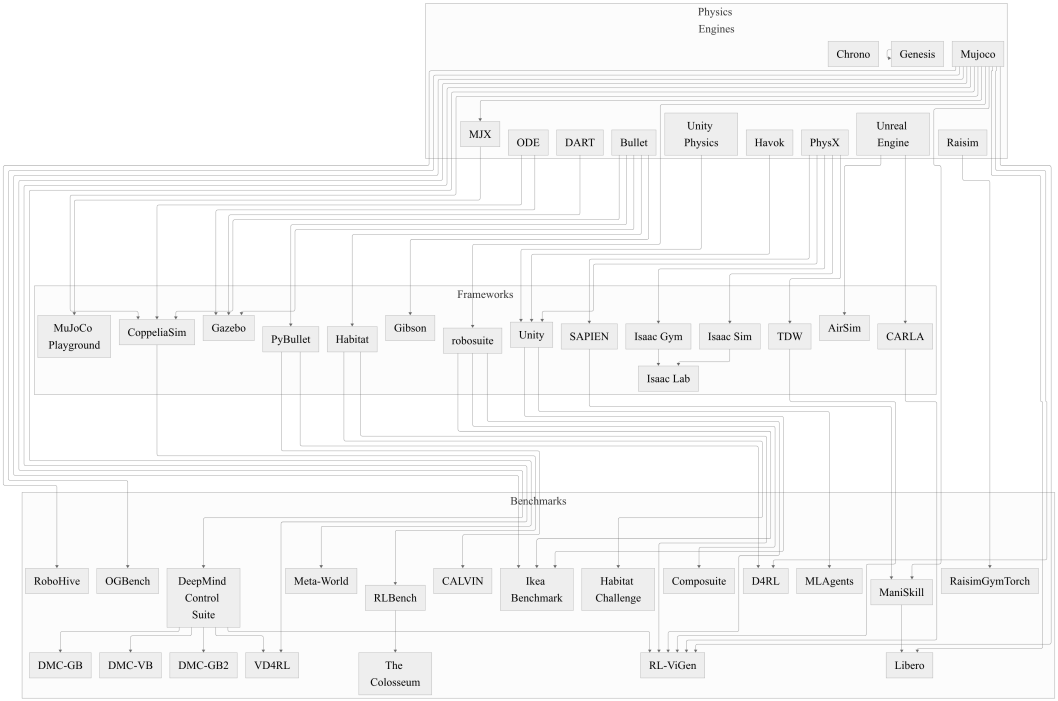

Figure 2: Dependency graph of physics engines, simulation frameworks, and benchmarks. Arrows indicate direct dependencies, pointing from the underlying system to the dependent one.

policies is essential, as physical deployment remains the ultimate goal for embodied intelligence. To explore this, we analyze trends in simulation usage across locomotion, manipulation, and navigation domains. Fig 3 shows a temporal breakdown of tool adoption from 2018 to 2024, comparing general usage with papers that explicitly address sim-to-real transfer. For instance, most recent locomotion papers rely on **Isaac Lab**[15], while other frameworks are less common for sim-to-real applications.

These trends do not necessarily indicate which simulation tools are most realistic but rather where sim-to-real workflows are easier to set up. A tool may be favored not for accuracy but for speed or convenience. ROS bridges for hardware-in-the-loop experiments [54] (e.g., in **Gazebo**) or deployment scripts can further extend a framework's utility beyond pure simulation. These insights can guide researchers in selecting tools that best align with their goals and practical constraints.

4.1.1 Locomotion considerations

Simulating realistic contact is crucial for locomotion yet current models often oversimplify or relax core physical principles [49, 50]. The fundamental challenge extends beyond these considerations; locomotion necessitates frameworks that facilitate comprehensive evaluations across different types of terrain variations, with domain randomization [55, 56] representing a critical functional parameter.

The development of locomotion simulation frameworks would benefit substantially from considering the following: **Contact formulation:** Implementation of physically accurate hard and soft contact dynamics promotes robustness and mitigates the gap in sim-to-real transfer. It appears that contact formulation using Bounded Linear Complementarity Problem [50, 52] suffers from unrealistic friction response, inaccurate sliding and contact instability in high-DoF. Whereas, contact formulation via convex optimization with soft constraints presents a promising approach [12].

No internal forces: Contact resolution avoids introducing internal forces in closed-loop systems (e.g., robots with multiple limbs on the ground), leading to minimal and physically consistent force distribution. **MuJoCo** [12] solely offers well-posed optimization that eliminates internal forces.

Robustness: MuJoCo [12], **Isaac Gym** [49], **Chrono** [48] deliver stable and fast performance, maintaining robustness across various configurations without degradation under complex scenarios.

4.1.2 Manipulation considerations

Simulation frameworks are crucial for developing and testing visual manipulation policies. Key factors influencing the choice of framework include the fidelity of the physics simulation, simulation speed, support for visual domain randomization, sensor simulation capabilities,

High Fidelity Rendering: The simulation's visual appearance is a feature of the rendering engine used and is separate from the physics engine. Rendering quality is important for policies using visual input, especially for sim-to-real transfer. Frameworks like **NVIDIA Isaac Sim**, **Unity**, and **SAPIEN** provide high-fidelity rendering options, supporting complex materials and lighting, sometimes including ray tracing. **Gazebo** and **CoppeliaSim** [57] offer medium-fidelity.

Visual Domain Randomization: To train models that generalize across task configurations, control over various environment variables is essential. Key variables to randomize include object properties, camera parameters, lighting conditions, background overlays, and out-of-distribution distractors.

Rich Sensors and Modalities Support: Different methods require different modalities and sensors. Common sensors for visual manipulation include RGB, depth, segmentation, and lidar. **Gazebo** provides support for a very extensive list of sensors through its plugin system and integrates well with ROS. **NVIDIA Isaac Sim** and **CoppeliaSim** and **SAPIEN** also offer robust support for common robotics sensors. **Unity** and **PyBullet** [58] provide core sensor support (primarily RGB cameras) and allow for custom sensor implementation requiring more work than other solutions.

4.1.3 Navigation considerations

On The Value of Tailor-Made Navigation Frameworks: Contrary to manipulation and locomotion, navigation does not (necessarily) require high-fidelity contact or friction simulation. Basic navigation can be achieved in any framework that supports simple motion primitives (e.g. "move forward", "turn") and some form of sensory input. However, advanced navigation tasks require rich semantics, high-quality visual modeling, and accurate sensor simulation.

Certain navigation regimes require extensive, specialized codebases that are not transferable to other tasks. Frameworks tailored for these niche applications include **SoundSpaces** [59, 60] and **ThreeDWorld** [61], which support high-quality audio simulation for sound-based navigation, as well as **CARLA** [62], which is designed for autonomous driving simulation.

Visual Fidelity vs. Asset Diversity: Simulation frameworks typically face a tradeoff between visual fidelity and the diversity of available assets. Due to the challenge obtaining datasets of high-quality visuals, frameworks such as **Habitat** [63, 64, 65] and **iGibson** [66, 67] that support datasets with photorealistic graphics such as **Matterport3D** [68][3], tend not to support the massive diversity found in frameworks that target the compositionality of many different assets, such as **AI2Thor** [69]. Yet, both rendering quality and task diversity are important for navigation methods, so the practitioner should choose frameworks carefully—or evaluate across both types when possible.

4.2 Benchmarks

Robot learning benchmarks are suites of tasks and evaluation protocols—each defined with fixed reward functions, termination conditions, and success metrics—to rigorously assess and compare the performance of different algorithms in a reproducible manner. The line between a benchmark and a framework can sometimes be blurred, especially when benchmarks are tightly coupled with specific frameworks or physics engines. Nevertheless, their function remains crucial: to standardize evaluation and drive progress by allowing researchers to measure advancements against established standards. In Table 3 we outline some of the features that are present in various benchmarks.

[3] See subsubsection 4.2.3 for further discussion on high-visual-quality datasets for Habitat.

Frameworks	**IsaacSim** [14]	**SAPIEN** [16]	**Unity** [70]	**CoppeliaSim** [57]	**Gazebo** [71]	**CARLA** [62]	**Habitat** [63, 64, 65]	**AI2Thor** [69]	**Robosuite** [17]	**iGibson** [66, 67]	**MuJoCo Playground** [41]	**PyBullet** [58]
Interoperability with Learning Frameworks	✔	✔	✔	∼	✗	✔	✔	✔	✔	✔	✔	∼
Domain Randomization	∼	∼	✗	✗	✗	✗	✗	✗	✗	✗	✔	✗
Curriculum												
Visual Domain Randomization	✔	✔	✗	✗	✗	✔	✔	✔	∼	✗	✗	∼
High Fidelity Rendering	✔	✔	✔	✗	✗	✔	✔	∼	✔	✔	✗	✗
Support												
Sim-to-Real Track Record	✔	✔	✗	∼	∼	✗	∼	∼	∼	✗	✗	✔
Locomotion Support	✔	✔	∼	∼	∼	✗	✗	✗	✗	∼	✔	✔
Manipulation Support	✔	✔	∼	✔	∼	✗	✗	✗	✔	✔	✔	✔
Navigation Support	∼	∼	✔	∼	✔	✔	✔	✔	✗	✔	✗	∼

Table 2: Comparison of Simulation Frameworks for Robot Learning

4.2.1 Locomotion considerations

In locomotion tasks, the diversity of tasks is of particular importance as the community improves results over current benchmarks [6, 7]. A benchmark should include a variety of locomotion tasks, such as walking, running, and climbing on varied terrains (flat, inclined, stepping stones, deformable ground) with different embodiments to ensure comprehensive evaluation across the different scenarios.

4.2.2 Manipulation considerations

There are a wide range of manipulation benchmarks: **Meta-World** & **RLBench** offer short-horizon manipulation tasks, while **LIBERO** and **CALVIN** [72] focus on long-horizon, compositional tasks, **RoboHive** provides complex tasks, and **ManiSkill** provides a smaller set of tasks but some with greater difficulty. Each benchmark targets a specific type of manipulation task(s).

Visual Robustness Evaluation: This assesses how well policies generalize to visual changes, often tested via domain randomization integrated into the benchmark's tasks. **The Colosseum** [73] applies systematic visual perturbations across 20 RLBench tasks, **DMC-GB/GB2** and **DMC-VB** introduce specific visual distractors, **RL-ViGen** [74] tests variations in appearance, lighting, and camera views but its setup can be less straightforward and much trickier. Benchmarks like **ManiSkill** and **RLBench** leverage simulation frameworks and physics engines with strong rendering and randomization capabilities, allowing for implicit tests of visual robustness depending on the task setup.

4.2.3 Navigation considerations

Diversity of Navigation Tasks: In navigation simulation frameworks, there exists a tradeoff between photorealism and task diversity, so different benchmarks will focus on one or the other. The **AI2Thor** simulation framework embraces task diversity, and supports, among others, two interesting suites of tasks: **ProcThor** is a procedural generation method for **AI2Thor** scenes, and **Holodeck** is a method for generating novel scenes using LLM queries applied to the **Objaverse** dataset (a dataset of 10 million 3D assets). In contrast, **Habitat** support specific datasets with very high quality visuals such as **Matterport3D**, **Replica**, **Gibson**. We report larger categories of navigation tasks that have been used to benchmark navigation methods in Table 4.

5 Conclusion

In this work, we have provided a survey of the rapidly growing space of physics engines, frameworks, and benchmarks for robot learning. Many of these pieces of software have been custom-designed for specific robotics applications, which limits their use. For example, the coupling between frameworks and specific learning libraries greatly reduces the reproducibility of robot learning methods, and

Benchmark	Domains Supported	Reward Diversity	Environment Complexity	Sim2Real	Visual Robustness	Diversity of Tasks
Mujoco Playground [41]	Manipulation, Locomotion	∼	✗	∼	∼	∼
DeepMind Control [75]	Locomotion	✗	✗	✗	✗	✗
OGBench [76]	Manipulation	∼	✗	✗	∼	✔
RoboHive [77]	Manipulation, Locomotion	✔	✔	✔	∼	✔
Meta-World [19]	Manipulation	✔	∼	∼	∼	✔
RlBench [5]	Manipulation	✔	✔	✔	✔	✔
ManiSkill [18]	Manipulation	✔	✔	∼	✔	✔
DMC-VB [78]	Manipulation	∼	✗	✗	∼	∼
DMC-GB2 [79]	Manipulation, Locomotion	∼	✗	✗	∼	∼
VD4RL [80]	Manipulation	∼	∼	✗	✔	∼
RL-ViGen [74]	Manipulation	∼	∼	✗	✔	✔
CALVIN [72]	Manipulation	✔	✔	✔	✔	✔
Colosseum [73]	Manipulation	∼	∼	∼	∼	∼
HumanoidBench [6]	Manipulation, Locomotion	✗	✔	✗	✔	✔
LocoMuJoCo [7]	Locomotion	✔	✗	∼	✗	✔
Habitat Challenge [81, 82, 83]	Navigation	✔	✔	✔	✔	∼

Table 3: Benchmarks Considerations

this work suggests that, to improve progress in the community, additional effort is needed to build common frameworks. Overall, the analysis provides a guide to new and experienced robot learning practitioners for choosing a library to begin their robot learning research. We suggest that future designers working at all levels of simulation consider the features above and work to reduce duplicated work in creating benchmarks.

Limitations This survey covers recent methods that are very particular for the robotics community. Many other simulation tools exist, for example, for medical simulation, but they were not covered in this work. This survey should be accurate up to the date it was submitted. It is possible that features are added after submission. This is why we will work to keep a community-supported live webpage where people can commit updates.

References

[1] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.

[2] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[4] X. Li, K. Hsu, J. Gu, O. Mees, K. Pertsch, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. In *8th Annual Conference on Robot Learning*, 2024.

[5] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment, 2019. URL https://arxiv.org/abs/1909.12271.

[6] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation, 2024.

[7] F. Al-Hafez, G. Zhao, J. Peters, and D. Tateo. Locomujoco: A comprehensive imitation learning benchmark for locomotion. In *6th Robot Learning Workshop, NeurIPS*, 2023.

[8] X. Jia, D. Blessing, X. Jiang, M. Reuss, A. Donat, R. Lioutikov, and G. Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. *CoRR*, 2024.

[9] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In *2018 IEEE International Conference on robotics and automation (ICRA)*, pages 5620–5627. IEEE, 2018.

[10] P. Hua, M. Liu, A. Macaluso, Y. Lin, W. Zhang, H. Xu, and L. Wang. Gensim2: Scaling robot data generation with multi-modal and reasoning llms. *arXiv preprint arXiv:2410.03645*, 2024.

[11] N. Bohlinger, G. Czechmanowski, M. P. Krupka, P. Kicki, K. Walas, J. Peters, and D. Tateo. One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=PbQOZntuXO.

[12] E. Todorov, T. Erez, and Y. 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.

[13] NVIDIA Corporation. NVIDIA PhysX. https://developer.nvidia.com/physx-sdk. Accessed on April 24, 2025.

[14] *NVIDIA Omniverse Isaac Sim (Version 2024.1)*. NVIDIA Corporation, 2024. URL https://developer.nvidia.com/isaac/sim. Accessed 30 Apr 2025.

[15] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. 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.

[16] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[17] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, A. Maddukuri, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning, 2025. URL https://arxiv.org/abs/2009.12293.

[18] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai, 2024. URL https://arxiv.org/abs/2410.00425.

[19] T. Yu, D. Quillen, Z. He, R. Julian, A. Narayan, H. Shively, A. Bellathur, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021. URL https://arxiv.org/abs/1910.10897.

[20] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

[21] T. Kim, M. Jang, and J. Kim. A survey on simulation environments for reinforcement learning. In *2021 18th International Conference on Ubiquitous Robots (UR)*, pages 63–67, 2021. doi:10.1109/UR52253.2021.9494694.

[22] T. Erez, Y. Tass, and E. Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, MuJoCo, ODE and PhysX. *IEEE Int. Conf. Robot. Autom.*, pages 4397–4404, 2015.

[23] M. Kaup, C. Wolff, H. Hwang, J. Mayer, and E. Bruni. A review of nine physics engines for reinforcement learning research, 2024. URL https://arxiv.org/abs/2407.08590.

[24] R. Newbury, J. Collins, K. He, J. Pan, I. Posner, D. Howard, and A. Cosgun. A review of differentiable simulators. *IEEE Access*, 12:97581–97604, 2024. doi:10.1109/ACCESS.2024.3425448.

[25] J. Collins, S. Chand, A. Vanderkop, and D. Howard. A review of physics simulators for robotic applications. *IEEE Access*, 9:51416–51431, 2021. doi:10.1109/ACCESS.2021.3068769.

[26] A. Farley, J. Wang, and J. A. Marshall. How to pick a mobile robot simulator: A quantitative comparison of coppeliasim, gazebo, morse and webots with a focus on accuracy of motion. *Simulation Modelling Practice and Theory*, 120:102629, 2022. ISSN 1569-190X. doi:https://doi.org/10.1016/j.simpat.2022.102629. URL https://www.sciencedirect.com/science/article/pii/S1569190X22001046.

[27] J. Kober and J. Peters. *Reinforcement Learning in Robotics: A Survey*, pages 9–67. Springer International Publishing, Cham, 2014. ISBN 978-3-319-03194-1. doi:10.1007/978-3-319-03194-1_2. URL https://doi.org/10.1007/978-3-319-03194-1_2.

[28] E. F. Morales, R. Murrieta-Cid, I. Becerra, and M. A. Esquivel-Basaldua. A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning. *Intelligent Service Robotics*, 14(5):773–805, Nov 2021. ISSN 1861-2784. doi:10.1007/s11370-021-00398-z. URL https://doi.org/10.1007/s11370-021-00398-z.

[29] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(Volume 3, 2020):297–330, 2020. ISSN 2573-5144. doi:https://doi.org/10.1146/annurev-control-100819-063206. URL https://www.annualreviews.org/content/journals/10.1146/annurev-control-100819-063206.

[30] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020. doi:10.1109/SSCI47803.2020.9308468.

[31] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters. Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, 9:799893, 2022.

[32] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research*, 22(30):1–82, 2021. URL http://jmlr.org/papers/v22/19-804.html.

[33] Z. Gu, J. Li, W. Shen, W. Yu, Z. Xie, S. McCrory, X. Cheng, A. Shamsah, R. Griffin, C. K. Liu, A. Kheddar, X. B. Peng, Y. Zhu, G. Shi, Q. Nguyen, G. Cheng, H. Gao, and Y. Zhao. Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning, 2025. URL https://arxiv.org/abs/2501.02116.

[34] S. Ivaldi, J. Peters, V. Padois, and F. Nori. Tools for simulating humanoid robot dynamics: A survey based on user feedback. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 842–849, 2014. doi:10.1109/HUMANOIDS.2014.7041462.

[35] H. Jiang, H. Wang, W.-Y. Yau, and K.-W. Wan. A brief survey: Deep reinforcement learning in mobile robot navigation. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 592–597, 2020. doi:10.1109/ICIEA48937.2020.9248288.

[36] K. Zhu and T. Zhang. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691, 2021. doi:10.26599/TST.2021.9010012.

[37] X. Xiao, B. Liu, G. Warnell, and P. Stone. Motion planning and control for mobile robot navigation using machine learning: a survey. *Autonomous Robots*, 46(5):569–597, Jun 2022. ISSN 1573-7527. doi:10.1007/s10514-022-10039-8. URL https://doi.org/10.1007/s10514-022-10039-8.

[38] H. Choi, C. Crump, C. Duriez, A. Elmquist, G. Hager, D. Han, F. Hearl, J. Hodgins, A. Jain, F. Leve, et al. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences*, 118(1):e1907856118, 2021.

[39] S. M. Kargar, B. Yordanov, C. Harvey, and A. Asadipour. Emerging trends in realistic robotic simulations: A comprehensive systematic literature review. *IEEE Access*, 12:191264–191287, 2024. doi:10.1109/ACCESS.2024.3404881.

[40] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[41] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. 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.

[42] M. X. Authors. Mujoco xla (mjx). https://mujoco.readthedocs.io/en/stable/mjx.html, 2024.

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

[44] Q. L. Lidec, W. Jallet, L. Montaut, I. Laptev, C. Schmid, and J. Carpentier. Contact models in robotics: a comparative analysis, 2024. URL https://arxiv.org/abs/2304.06372.

[45] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. doi:10.1109/IROS.2017.8202133.

[46] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018. doi:10.1109/ICRA.2018.8460528.

[47] X. Zhang, C. Wang, L. Sun, Z. Wu, X. Zhu, and M. Tomizuka. Efficient sim-to-real transfer of contact-rich manipulation skills with online admittance residual learning, 2023. URL https://arxiv.org/abs/2310.10509.

[48] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut. Chrono: An open source multi-physics dynamics engine. In T. Kozubek, R. Blaheta, J. Šístek, M. Rozložník, and M. Čermák, editors, *High Performance Computing in Science and Engineering*, pages 19–49, Cham, 2016. Springer International Publishing. ISBN 978-3-319-40361-8.

[49] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[50] Bullet Physics Authors. Bullet physics engine. http://www.bulletphysics.org, 2015.

[51] Havok. Havok physics. https://www.havok.com/.

[52] R. Smith. Open dynamics engine (ode). https://www.ode.org/, 2001. [Online]. Available: www.ode.org.

[53] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu. DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22): 500, Feb 2018. doi:10.21105/joss.00500. URL https://doi.org/10.21105/joss.00500.

[54] N. Bohlinger, J. Kinzel, D. Palenicek, L. Antczak, and J. Peters. Gait in eight: Efficient on-robot learning for omnidirectional quadruped locomotion. *arXiv preprint arXiv:2503.08375*, 2025.

[55] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *Robotics: Science and Systems XVII*, 2021.

[56] M. Stasica, A. Bick, N. Bohlinger, O. Mohseni, J. Fritzsche, C. Hübler, J. Peters, and A. Seyfarth. Bridge the gap: Enhancing quadruped locomotion with vertical ground perturbations. In *Under review*, 2025. URL https://www.ias.informatik.tu-darmstadt.de/uploads/Team/NicoBohlinger/bridge_the_gap.pdf.

[57] E. Rohmer, S. P. N. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326, 2013. doi:10.1109/IROS.2013.6696520.

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

[59] C. Chen, U. Jain, C. Schissler, S. V. A. Gari, Z. Al-Halah, V. K. Ithapu, P. Robinson, and K. Grauman. Soundspaces: Audio-visual navigation in 3d environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 17–36. Springer, 2020.

[60] C. Chen, C. Schissler, S. Garg, P. Kobernik, A. Clegg, P. Calamia, D. Batra, P. Robinson, and K. Grauman. Soundspaces 2.0: A simulation platform for visual-acoustic learning. *Advances in Neural Information Processing Systems*, 35:8896–8911, 2022.

[61] C. Gan, J. Schwartz, S. Alter, D. Mrowca, M. Schrimpf, J. Traer, J. De Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, et al. Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*, 2020.

[62] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[63] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[64] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[65] X. Puig, E. Undersander, A. Szot, M. D. Cote, R. Partsey, J. Yang, R. Desai, A. W. Clegg, M. Hlavac, T. Min, T. Gervet, V. Vondruss, V.-P. Berges, J. Turner, O. Maksymets, Z. Kira, M. Kalakrishnan, J. Malik, D. S. Chaplot, U. Jain, D. Batra, A. Rai, and R. Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.

[66] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. P. Tchapmi, M. E. Tchapmi, K. Vainio, J. Wong, L. Fei-Fei, and S. Savarese. igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page accepted. IEEE, 2021.

[67] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. E. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, K. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 455–465. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/li22b.html.

[68] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[69] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[70] Unity Technologies. Unity, 2023. URL https://unity.com/. Game development platform.

[71] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi: 10.1109/IROS.2004.1389727.

[72] O. Mees, L. Hermann, E. Rosete-Beas, and W. 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.

[73] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. 2024.

[74] Z. Yuan, S. Yang, P. Hua, C. Chang, K. Hu, X. Wang, and H. Xu. Rl-vigen: A reinforcement learning benchmark for visual generalization. *arXiv preprint arXiv:2307.10224*, 2023.

[75] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. Deepmind control suite, 2018. URL https://arxiv.org/abs/1801.00690.

[76] S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025.

[77] V. Kumar, R. Shah, G. Zhou, V. Moens, V. Caggiano, J. Vakil, A. Gupta, and A. Rajeswaran. Robohive: A unified framework for robot learning, 2023. URL https://arxiv.org/abs/2310.06828.

[78] J. Ortiz, A. Dedieu, W. Lehrach, J. S. Guntupalli, C. Wendelken, A. Humayun, G. Zhou, S. Swaminathan, M. Lázaro-Gredilla, and K. Murphy. Dmc-vb: A benchmark for representation learning for control with visual distractors. 2024.

[79] A. Almuzairee, N. Hansen, and H. I. Christensen. A recipe for unbounded data augmentation in visual reinforcement learning, 2024.

[80] C. Lu, P. J. Ball, T. G. J. Rudner, J. Parker-Holder, M. A. Osborne, and Y. W. Teh. Challenges and opportunities in offline reinforcement learning from visual observations, 2023. URL https://arxiv.org/abs/2206.04779.

[81] Abhishek Kadian*, Joanne Truong*, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? volume 5, pages 6670–6677, 2020.

[82] K. Yadav, S. K. Ramakrishnan, J. Turner, A. Gokaslan, O. Maksymets, R. Jain, R. Ramrakhya, A. X. Chang, A. Clegg, M. Savva, E. Undersander, D. S. Chaplot, and D. Batra. Habitat challenge 2022. https://aihabitat.org/challenge/2022/, 2022.

[83] K. Yadav, J. Krantz, R. Ramrakhya, S. K. Ramakrishnan, J. Yang, A. Wang, J. Turner, A. Gokaslan, V.-P. Berges, R. Mootaghi, O. Maksymets, A. X. Chang, M. Savva, A. Clegg, D. S. Chaplot, and D. Batra. Habitat challenge 2023. https://aihabitat.org/challenge/2023/, 2023.

[84] T. Wang, R. Liao, J. Ba, and S. Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1sqHMZCb.

[85] W. Huang, I. Mordatch, and D. Pathak. One policy to control them all: shared modular policies for agent-agnostic control. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[86] J. Whitman, M. Travers, and H. Choset. Learning modular robot control policies. *IEEE Transactions on Robotics*, 39(5):4095–4113, 2023. doi:10.1109/TRO.2023.3284362.

[87] L. Campanaro, S. Gangapurwala, W. Merkt, and I. Havoutis. Learning and deploying robust locomotion policies with minimal dynamics randomization. In A. Abate, M. Cannon, K. Margellos, and A. Papachristodoulou, editors, *6th Annual Learning for Dynamics & Control Conference, 15-17 July 2024, University of Oxford, Oxford, UK*, volume 242 of *Proceedings of Machine Learning Research*, pages 578–590. PMLR, 2024. URL https://proceedings.mlr.press/v242/campanaro24a.html.

[88] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018. doi:10.1109/ICRA.2018.8460528.

[89] A. Hendawy, J. Peters, and C. D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=aZH1dM3GOX.

[90] R. McLean, E. Chatzaroulas, J. Terry, I. Woungang, N. Farsad, and P. S. Castro. Multi-task reinforcement learning enables parameter scaling, 2025. URL https://arxiv.org/abs/2503.05126.

[91] V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson. My body is a cage: the role of morphology in graph-based incompatible control. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=N3zUDGN5lO.

[92] B. Trabucco, M. Phielipp, and G. Berseth. AnyMorph: Learning transferable polices by inferring agent morphology. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 21677–21691. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/trabucco22b.html.

[93] R. McLean, K. Yuan, I. Woungang, N. Farsad, and P. S. Castro. Overcoming state and action space disparities in multi-domain, multi-task reinforcement learning. In *[CoRL 2024] Morphology-Aware Policy and Design Learning Workshop (MAPoDeL)*, 2024. URL https://openreview.net/forum?id=T7bA2zjobB.

[94] M. Lentine. Chaos scene queries and rigid body engine in ue5. https://www.unrealengine.com/en-US/tech-blog/chaos-scene-queries-and-rigid-body-engine-in-ue5, May 2022. Unreal Engine Tech Blog.

[95] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[96] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[97] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems*, 07 2020. doi:10.15607/RSS.2020.XVI.064.

[98] M. Theile, H. Cao, M. Caccamo, and A. L. Sangiovanni-Vincentelli. Equivariant ensembles and regularization for reinforcement learning in map-based path planning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024.

[99] M. Theile, H. Bayerlein, M. Caccamo, and A. L. Sangiovanni-Vincentelli. Learning to recharge: Uav coverage path planning through deep reinforcement learning. *arXiv preprint arXiv:2309.03157*, 2023.

[100] C. Gulino, J. Fu, W. Luo, G. Tucker, E. Bronstein, Y. Lu, J. Harb, X. Pan, Y. Wang, X. Chen, J. D. Co-Reyes, R. Agarwal, R. Roelofs, Y. Lu, N. Montali, P. Mougin, Z. Z. Yang, B. White, A. Faust, R. T. McAllister, D. Anguelov, and B. Sapp. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=7VSBaP2OXN.

Appendix

Extended details on the survey.

A Contributions

E.D., N.B., and R.M. contributed to the paper organization, with E.D. reviewing and condensing the paper to meet page limits, N.B. writing most of the abstract and introduction, and R.M. handling overall writing, editing, and organization. R.M., M.P., and M.G. produced the technical framework of the paper, where R.M. expanded the robot learning framework and proposed the three-category organization of physics engines, frameworks, and benchmarks, while M.P. and M.G. created the robot learning problem outline and co-defined MDP formulations for each simulator. M.P. and M.G. conducted the comprehensive literature review and taxonomy development, systematically reviewing robot benchmark papers and analyzing task representations across existing benchmarks. Domain-specific analysis was divided among team members: E.D., N.B., and M.G. focused on locomotion considerations, with N.B. conducting speed tests across simulators and M.G. authoring the locomotion considerations section; A.B. and A.K. handled all manipulation-related sections, with A.B. as the primary author and A.K. providing expertise and proofreading; V.S.V., C.G., and M.S.R. contributed to navigation analysis, with V.S.V. focusing on navigation planning requirements and sensor analysis while C.G. and M.S.R. co-wrote navigation subsections. Data compilation and visualization efforts were led by E.D. who created the main figures and assembled tables, with contributions from A.B. on manipulation-focused figures, O.A. on table completion, and V.S.V. on benchmark table entries. L.P, J.P, Gio.Bel., and Gl.Ber. advised on the project.

B Robot Learning Problem Details

B.1 Action and State Spaces depend on Morphologoies

These components directly influence the type of algorithm that can be used for the learning of the RLP. Firstly, the choice of action space a depends on the embodiment m and the selected task(s) t. The action space a for a Franka robot could be the set of velocities and that are applied to each joint of the arm, or it could be the directional choice for a navigational robot in combination with the desired displacement. Thus the choice of action space is formalized as $a \sim p(\cdot|m, t)$. Next, the observation space can be chosen for the problem. In robot learning problems there are a few choices of observation space the user can choose from. When selecting an observation space $o \sim P(\cdot|m, t, v)$, the user can use environment state-vectors, RGB pixel values, and/or sensors. This choice of observation space is tied to the choice of embodiment m, chosen task t, and the physics engine v as not all physics engines will support all types of sensors. We do note that we frame the choice of action space and observation space as a probability distribution to capture how the choices of task, embodiment, and physics engine also influence this choice. In designing the RLP, we found that to be complete we needed to handle the situation where for the same embodiment and task, it can be possible to define the problem with unique configurations. Here, the use of probability notation reflects that for some problem setting, multiple valid parameterizations exist. The distribution captures this choice space, not randomness.

B.2 Simulator choice affects RLP

After selecting the embodiment m and reward r for their respective problem, the user must select a physics engine v from the set of all possible physics engines $\mathcal{V}$. There is an ever-growing list of possible physics engines to use, each with their own pros and cons. We postulate that this choice of physics engine is dependent on the task t, embodiment m, and the ability of a physics engine $v \in \mathcal{V}$ to modify the world parameters $w \sim p(\mathcal{W})$. These world parameters $\mathcal{W}$ introduce the ability for the user to define their exact environment settings, such as: gravity, friction, wind resistance, or system noise. Thus, the user may attempt to choose a physics engine through the following process. First

the user identifies the set of world parameters that each physics engine supports $w_v \subseteq \mathcal{W}$. Next, the user must identify the set of world parameters that are needed for their chosen task $w_t \subseteq \mathcal{W}$. Once the world parameters of the physics engine w_v and task w_t are chosen, the user can select a physics engine $v \sim p(v|t, m, W_v \supseteq W_t)$. Finally, the user can also select the required world parameters w $p(w|v, t, m)$ where $w \in W_v \cap W_t$. The world parameters w chosen for the users specific problem affect the dynamics of the overall system $p(s_{i+1}|s_i, a_i, t, m, w, v)$, where the current state s_i and action a_i at timestep i influence the outcome of the next state s_{i+1}, in addition to the current task(s) t, embodiment m, world parameters w, and simulator v. We formulate the dynamics of a system being affected by each of these various components due to the decision-making that goes into selecting the simulator v and world parameters w. Due to not all simulators being able to effectively simulate all phenomena that we may require, the choice of task influences the choice of simulator. For a single simulator may not be able to handle one task that requires simulating contacts between rigid bodies compared to another task that requires simulating soft contacts.

B.3 Applications of the Robot Learning Framework

Given the broad definition of the robot learning problem, we re-frame a number of existing classes of learning problems and the relevant components to the general formulation that have been considered in the robot learning community.

In addition to the groupings that we provide for various robot learning problems, our RLP framework can also be used to effectively group various works in the multi-task robot learning literature. In general, there are three different strains of multi-task robot learning research, where the definition of a task is modified slightly: cross embodiment learning, domain randomization, and task specification. In the cross embodiment setting, the distribution of embodiments $p(\mathcal{M})$ that an embodiment can sample from includes more than one robot type. For example this line of research aims to control multiple types of robots where the embodiment distribution $p(\mathcal{M})$ may include quadruped, humanoid, and hexapod legged robots, either in simulation or the physical world [84, 85, 86, 11].

Next, the domain randomization setting can also be posed as a multi-task learning problem. In this problem setting the choice of world parameters w and physics engine v affect the ability for a single robotic policy to be learned, and learning across various configurations of state transition dynamics can be framed as a multi-task problem. For example Campanaro et al. [87] & Peng et al. [88] inject noise into the training process to perturb system dynamics, effectively forcing their model to learn multiple sets of dynamics, which can then generalize well to unseen sets of dynamics during evaluation on real-world hardware after being trained entirely in simulation.

Finally, the task specifications problem setting is likely the most 'traditional' multi-task learning setting. In this problem setting there are any number of unique tasks in the task distribution $t \sim p(\cdot|m, \mathcal{T})$ that a single policy must learn to accomplish. In this setting, the choice of embodiment m is generally limited to a single choice of m. This framing of the RLP fits benchmarks such as Meta-World [19], and the works that use these benchmarks [89, 90]. There are also some works that don't necessarily fit into a single category and cross over between axes of the RLP. For example some works have explored learning representations over multiple morphologies $m_i \sim p(\mathcal{M})$ where i is the index of the ith morphology chosen, across a mix of tasks $t_j \sim p(\cdot|m_i)$, where j is the index of the jth task chosen for morphology m_i. Several works can be framed using our RLP with this extension, with the goal of learning a single policy across multiple morphologies and/or tasks[84, 91, 92, 93].

B.3.1 Navigation considerations

Navigation planning refers to the process of computing a feasible and efficient path for a robotic system to move through an environment while achieving high reward r. This typically requires the robot to perceive its surroundings, localize itself within a map, and reason about future states to minimize a given objective—such as distance traveled, energy consumed, or risk encountered.

From the perspective of physics engines, several key attributes are critical for navigation-focused simulations:

1. **Sensor support:** In navigation tasks, the observation space o is generally limited to observation spaces that provide realistic sensor data (eg, Camera, Lidar, RGB-D, IMU, GPS). The selection of a physics engine v must allow for generating realistic sensor data as it is vital for tasks like Simultaneous Localisation and Mapping (SLAM), Obstacle avoidance, and motion planning.
2. **Kinematic and Dynamic Fidelity:** The chosen physics engine v must accurately model robot motion across various locomotion embodiments m —including differential drive, skid-steering, and legged locomotion—to ensure that planned trajectories correspond to feasible real-world movements.
3. **Environmental Interactions:** The ability to manipulate world parameters w allows the phsyics engine to model various properties such as friction, elevation, deformability, and slippage. This enables realistic testing of navigation in both structured and unstructured terrains.
4. **Real-Time and Deterministic Execution:** For real-time applications such as onboard navigation systems or hardware-in-the-loop testing, the choice of physics engine v should maintain low-overheads to accelerate performance and deterministic execution to maintain timing guarantees.
5. **Collision Detection and Contact Modeling:** A physics engine v must enable robust and accurate contact modeling. This helps in testing realistic collision-avoidance and recovery strategies. Fine-grained collision detection between robot components and obstacles contributes to reliable safety evaluation.
6. **Scalability for Multi-Agent Systems:** In scenarios involving fleets of robots or swarm navigation, the physics engine v must handle multiple agents interacting within the same environment efficiently without degrading performance.

Physics engines allow agents to learn robust navigation policies transferable to the real-world. When choosing an engine, we recommend prioritizing the following features:

Holonomic Constraints: A physics engine v uses holonomic (joint) constraints to model the coupling between two rigid parts, limiting the effective degrees of freedom of the embodiment. Engines with standard and configurable joints support a larger set of robot embodiments, $\mathcal{M}$, that are not limited to wheeled robots. For wheeled robots, **Bullet** [50] and **PhysX** [13] offer good customization and speed, while **MuJoCo** [12] handles custom morphologies better, providing higher accuracy and composite joints at the cost of speed. **MJX** [42] strikes a balance between accuracy and speed. Self-driving applications typically rely on **Chrono** [48] and **Chaos** [94] for advanced simulation of vehicle dynamics. Field robotics applications generally tend to use Gazebo [71] and its underlying physics engine **DART** [53] due to its support for accurate simulation of environmental interactions.

Collision & Contact Dynamics: Accurate and fast collision dynamics are fundamental for navigation applications. Without them, embodied agents cannot accurately represent real-world dynamics. The main tradeoff in selecting a physics engine is speed versus accuracy. For simple embodiments, it suffices to have optimized primitives to handle collisions, but as the embodiment complexity increases, more accurate mesh representations are required, as discussed in 4.0.2. Similarly, the fidelity of contact dynamics is crucial for navigation tasks, as it directly impacts the set of world dynamics $\mathcal{W}$ the agent will be exposed to. For an outline of physics engines for contact dynamics, see 4.0.2. When selecting a physics engine, we recommend assessing the complexity of the embodiment and the level of accuracy needed to model the dynamics of the agent and the world.

C Additional Simulation Framework Considerations

Interoperability with Learning Frameworks Table 2: In order to learn the robotic control policy parameters θ, a simulation framework should support integration of a machine learning library. There are many libraries to choose from such as Stable Baselines3 [95], rsl-rl [3]) or flexible implementations of RL algorithms like Proximal Policy Optimization (PPO) [96].

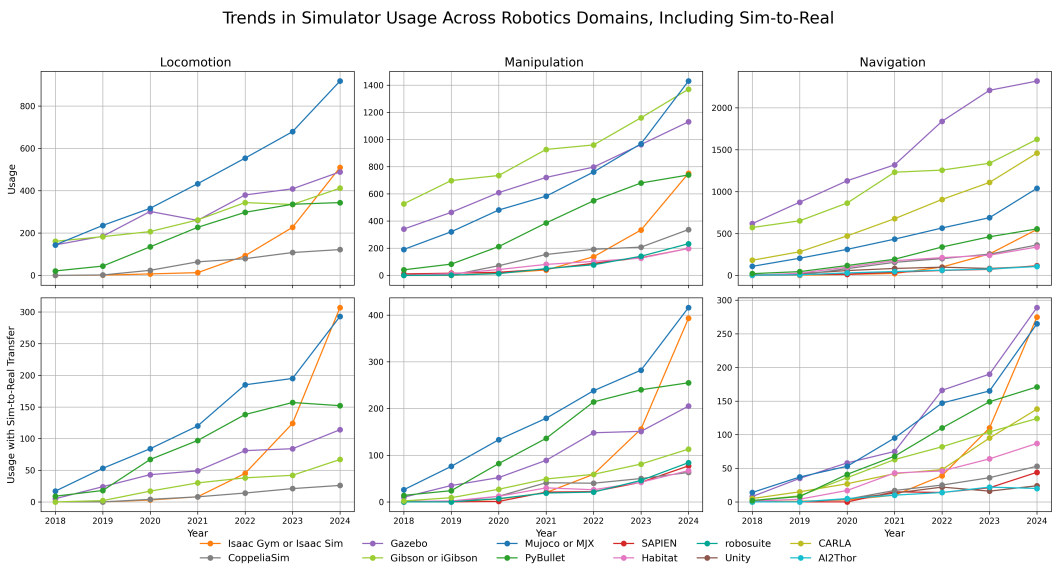

Figure 3: The top row illustrates usage in purely simulated settings—without explicit sim-to-real intent—for locomotion, manipulation, and navigation tasks. The bottom row highlights studies that explicitly mention sim-to-real transfer. Data represent approximated values, collected via keyword-based Google Scholar queries (e.g., "simulator name", "robot", "learning", "sim-to-real"). Simulators with low overall relevance or limited domain-specific usage were excluded for clarity.

Visual Domain Randomization (VDR) Table 2: Common types of visual randomization include being able to change at least the following properties of the scene during simulation : camera (position, orientation, focal length, sensor size), lighting (direction, intensity, brightness, color, number of lights), textures and materials, objects (color, size, position), scene composition, rendering effects (different quality settings), being able to load distracting objects (out-of-distribution objects), static overlay (eg: changing the background to a fixed image) and dynamic overlay (eg: changing the background to a video). These variations are fundamental for training robust visual policies, and different frameworks offer varying levels of built-in support.

Isaac Sim (often via **Isaac Lab**) provides extensive, integrated support for adding distracting objects to the scene, randomizing cameras, lighting, PBR materials/textures, and object properties through configuration files and APIs. It supports nearly all common visual randomization features mentioned earlier, however dynamic video overlays might be more challenging to achieve. The **SAPIEN** simulator (used by **Maniskill**) [16] also offers strong programmatic control over visual randomization, including similar support to what NVIDIA **Isaac Sim** offers. Implementing dynamic overlays would also require custom work here. Other frameworks require more programmatic effort for extensive randomization. **PyBullet** offers Python APIs to change visual properties like colors, textures, lighting, and camera parameters, making randomization scriptable but less integrated and it does not support all VDR features.

Modularity and Abstractions: In order to enable effective research, many components of a research project should be modular and leverage abstractions to ensure clean coding practices. Abstracting away details allows for better code reuse and reproducibility. For example, Peng et al. [97] includes a number of wrappers to perform state filtering, history stacking, or action delays. These wrappers are now open-sourced which enables researchers to use and combine these wrappers into their own works. However, these features are not abstract enough to be applied to any physics engine. Frameworks should include modular components for task definitions, action and observation spaces, terrains and obstacles, to facilitate rapid task design and experimentation. Isaac Lab and Maniskill provide modular and user-friendly APIs allowing easy swapping of terrain generators, robot models, or reward

Table 4: Navigation Benchmark Tasks

Task	Description	Notable Simulators
PointNav, ImageNav, ObjectNav	Nav. to a position/image/object.	Habitat, AI2Thor, Minos, iGibson
Audio Goal Nav	Nav. to an audio source, or nav. using audio information	SoundSpaces, ThreeDWorld
Mapping	Build a map (e.g., using SLAM methods)	Habitat, Minos, Gazebo
Exploration	Maximize environment coverage in a given amount of time	UAVSim[98, 99], Gazebo
Autonomous Driving	Nav. while obeying safety constraints and road signage, etc.	Carla[62], Waymax[100]

functions, while MuJoCo Playground is more rigid and Gazebo provides no custom APIs for many components necessary in locomotion tasks.

D Additional Benchmark Considerations

Reward Diversity: R refers to the flexibility user's can specify different functions $r : \mathcal{M} \times \mathcal{W} \times S \times A \to \mathbb{R}$ that encode a meaningful signal to accomplish different tasks. This is different from simply specifying a task which we claim can be simplified as having been accomplished or not. This is exactly the same as a sparse reward function. For a given task, there may be a number of ways of writing a dense function r that are useful learning signals associated with task success (the ultimate goal).

Environment Complexity: Similar to the forward model M a benchmark provides environments with varying levels of complexity, including obstacles, terrain variations, and dynamic elements. This complexity is crucial for evaluating the robustness and adaptability of locomotion algorithms.

Sim-to-Real Relevance: The benchmark should be designed with sim-to-real transfer in mind, ensuring that the tasks and environments are relevant to real-world scenarios. This may involve using realistic physics engines, accurate robot models, and domain randomization techniques.