The Reality Gap in Robotics: Challenges, Solutions, and Best Practices

Elie Aljalbout¹, Jiaxu Xing¹, Angel Romero¹, Iretiayo Akinola², Caelan Reed Garrett², Eric Heiden², Abhishek Gupta⁴, Tucker Hermans^{2,5}, Yashraj Narang², Dieter Fox⁴, Davide Scaramuzza¹, Fabio Ramos^{2,3}

Preprint to appear in the 2026 Annual Review of Control, Robotics, and Autonomous Systems, Vol. 9

Copyright © 2026 by the author(s). All rights reserved

Keywords

simulation, robot learning, sim-to-real transfer, reality gap

Abstract

Machine learning has facilitated significant advancements across various robotics domains, including navigation, locomotion, and manipulation. Many such achievements have been driven by the extensive use of simulation as a critical tool for training and testing robotic systems prior to their deployment in real-world environments. However, simulations consist of abstractions and approximations that inevitably introduce discrepancies between simulated and real environments, known as the reality gap. These discrepancies significantly hinder the successful transfer of systems from simulation to the real world. Closing this gap remains one of the most pressing challenges in robotics. Recent advances in sim-to-real transfer have demonstrated promising results across various platforms, including locomotion, navigation, and manipulation. By leveraging techniques such as domain randomization, realto-sim transfer, state and action abstractions, and sim-real co-training, many works have overcome the reality gap. However, challenges persist, and a deeper understanding of the reality gap's root causes and solutions is necessary. In this survey, we present a comprehensive overview of the sim-to-real landscape, highlighting the causes, solutions, and evaluation metrics for the reality gap and sim-to-real transfer.

 $^{^{\}mathbf{1}} \text{Robotics}$ and Perception Group, University of Zurich, Zurich, Switzerland, 8050

²NVIDIA, Seattle, USA, 98105

³The University of Sydney, Sydney, Australia, 2006

⁴University of Washington, Seattle, USA, 98195

⁵University of Utah, Salt Lake City, USA, 84112

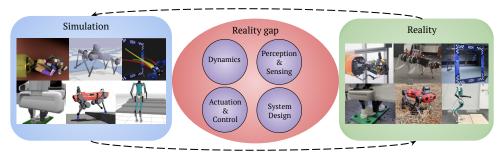
1. Introduction

Simulation holds great potential for robot learning, benchmarking and large-scale data collection in robotics due to its scalability, safety, and efficiency. Robots can be safely trained in simulation before being deployed to the real world. Algorithms can be compared against each other in simulation over multiple simulated scenarios. Through simulation, massive amounts of data can be collected from robots performing complex tasks at a fraction of the cost compared to the real world. However, the gap between simulated and realworld environments often stands in the way of fully leveraging its potential. Bridging the gap between simulation and the real world has become one of the most critical and long-standing challenges in robotics. Overcoming this challenge holds the potential to accelerate progress in robotics similar to what has been achieved in other computational fields. For example, sustained progress observed in the natural language processing and computer vision communities would not have been possible without significant effort in the development of large training and benchmarking datasets (1, 2). In contrast, robotics still lags behind in these aspects. One reason for this is that static test datasets cannot reflect the complexity of perceiving and acting in the real world. Instead, most robotics problems require training and benchmarking in interactive environments. While training in the real world has the benefit of making sure that systems are trained and tested in realistic settings, complex real-world tasks are difficult to scale and replicate with sufficient reproducibility. In addition, data collection in the real world is bottlenecked by multiple factors such as hardware cost, human dependence, and difficulty of automation. Simulation offers an affordable alternative for these challenges, making it an invaluable tool for data collection, benchmarking, and building various components of typical robotic pipelines, such as perception, planning, learning, and control. Namely, for benchmarking, simulation ensures the repeatability of experiments, which is impossible in real-world robotics systems due to their stochastic nature. For data generation, simulation provides an efficient approach that can leverage multiple robots operating faster than in real-time and in a parallelized manner.

Simulation attempts to replicate physical reality with mathematical abstractions, models, and approximations. This means that there are no perfect simulators, and therefore there is always a difference from the real world, which we call reality gap. While it is a common misconception to talk about this gap as a single element, the reality gap consists of a large number of sub-gaps resulting from the simulation's failure to replicate various physical real-world mechanisms and phenomena accurately. As a consequence, transferring any sort of behavior observed in simulation to a real-world environment can be extremely challenging. For instance, transferring control policies designed or learned in simulation is not trivial (9). Due to the differences between the simulated and the real environments, policies obtained in simulation could achieve great performance in simulation, simply by abusing modeling inaccuracies and simulator-specific corner cases. Hence, the successful transfer of such policies is not guaranteed and can even be dangerous to the robot, its surroundings, and any human in its proximity.

Despite these challenges, recent progress in robot learning has shown great promise in sim-to-real transfer, where control policies learned in a simulation are transferred to a similar real-world environment, and multiple techniques have been proposed to overcome

Real-to-sim transfer



Sim-to-real transfer

Figure 1

The reality gap is the difference between a simulated and real environment in aspects related to dynamics, perception & sensing, actuation & control, and system design. By carefully designing these modules, the gap can be reduced to a reasonable size. Sim-to-real as well as real-to-sim transfer require methods that can carefully overcome the remaining reality gap. Figures are adapted from successful sim-to-real applications in various robotics domains (3, 4, 5, 6, 7, 8).

the reality gap for different robotic platforms (10, 11, 12, 13, 3, 14, 6, 5, 8, 15). Besides improvements in physics engines and rendering quality, simulation technology progressed to meet the demands of the robot-learning community by following the deep learning trend of massive parallelization and large-scale data collection. Modern simulators can leverage GPU parallelization to simulate thousands of robots simultaneously (16, 17, 18, 19). This ability to efficiently parallelize simulations has led to significant breakthroughs in the field, particularly in locomotion (6, 20), agile flight (21, 8), and manipulation (3, 14, 22, 23). In locomotion, highly parallelized simulation enabled learning very robust quadruped locomotion across challenging terrains (6) and, more recently bipeds control (24, 7). Furthermore, sim-to-real transfer has been used for dexterous manipulation as well as other various single-arm (25, 5) and dual-arm tasks (26), ranging from simple 3D reaching to more complex and contact-rich tasks such as table-top rearrangement (25) and assembly (26, 5). Additionally, sim-to-real transfer played a key role in learning agile quadrotor control policies that outperformed human champions in drone racing (21, 8).

However, sim-to-real still faces many challenges, such as photorealistic rendering (more generally, sensor modeling) and simulation of complex dynamics such as contacts, deformations, and material variations. To further progress sim-to-real transfer and understand its limitations, a common and structured understanding of the problem is necessary. Ideally, such an understanding should go all the way from the roots of the problem to the existing and needed solutions to overcome them.

In this survey, we provide a comprehensive overview of the sim-to-real landscape for robot learning. We dissect the problem into atomic components, identify the sources of the reality gap and the symptoms they cause, and provide metrics and solutions to understand these problems and alleviate them in practice. Our objective is to boost the understanding of the problem by providing a guide for researchers and practitioners. We first introduce the problem, its notation and its theory in section 2. In section 3, we identify and exhaustively list the different causes of the reality gap. We then survey existing solutions and metrics for

sim-to-real transfer in section 4 and section 5, respectively. Finally, in section 6, we discuss open problems and an outlook for future research on this topic.

2. Preliminaries

In this section, we formalize the concepts and definitions required to understand and characterize the reality gap as well as sim-to-real transfer.

2.1. States, Observations, and Actions

We first introduce a general mathematical model for robot systems that receive observations of the world using their sensors and make decisions that maximize a particular objective. We formulate this decision-making problem as a Partially Observable Markov Decision Process (POMDP) ¹. A POMDP is defined as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O}, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ denotes the state space encompassing all feasible configurations of the robot and its environment. $\mathcal{A} \subseteq \mathbb{R}^m$ is the action space comprising all control commands the robot can execute; $\mathcal{T}: \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denotes the transition dynamics, describing how the state evolves given the current state \mathbf{s}_t at time t and a given action \mathbf{a}_t ; $\mathcal{R}: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function encoding the task objective; $\mathcal{Z} \subseteq \mathbb{R}^h$ is the observation space, and $\mathcal{O}: \mathcal{S} \to \mathcal{Z}$ is the sensor (observation) model that maps the latent state to an observation \mathbf{z}_t ; finally, $\gamma \in [0,1)$ is the discount factor, representing the relative importance of future rewards with respect to immediate rewards. The agent maintains a belief $\mathbf{b}_t \in \mathcal{B}(\mathcal{S})$, a probability distribution over states, updated from the history of past actions and observations. Its goal is to learn a policy $\pi: \mathcal{B}(\mathcal{S}) \to \mathcal{A}$ that maximizes the expected discounted return

$$\pi^* = \arg \max_{\pi} \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(\mathbf{s}_t, \pi(\mathbf{b}_t)) \right],$$
 1.

where $\mathcal{R}(\mathbf{s}_t, \pi(\mathbf{b}_t))$ is the reward the agent receives when in belief \mathbf{b}_t acting under the policy π . In practice, sim-to-real work often represents \mathbf{b}_t by a compact feature vector extracted directly from raw observations such as camera images and proprioceptive signals.

2.2. Simulation

A simulation is a computational approximation of the real world. If we denote the real world dynamics perfectly capturing the behavior of the real world system as $\mathcal{T}_r(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, and the real observation model $\mathcal{O}_r(\mathbf{z}_t|\mathbf{s}_t)$, a simulation approximates the real world dynamics and observation model by employing physical models derived from first principles, numerical integration, and approximations to reduce complexity and computational cost, such that $\mathcal{T}_{\mathbf{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \approx \mathcal{T}_r(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and $\mathcal{O}_{\mathbf{s}}(\mathbf{z}_t|\mathbf{s}_t) \approx \mathcal{O}_r(\mathbf{z}_t|\mathbf{s}_t)$, $\forall \mathbf{s}_t \in \mathcal{S}$, $\mathbf{a}_t \in \mathcal{A}$. A perfect simulator aims to minimize the discrepancy between the simulated and real world dynamics as well as the geometric structure and the simulated observations. However, perfect fidelity is impractical and computationally infeasible in realistic robotic scenarios due to the inherent complexity of physical effects (friction, noise, sensor latency, etc).

¹When the sensors provide the full state without noise (e.g. in simulation with privileged information) observations are equivalent to states and the formulation collapses to a Markov Decision Process (MDP).

2.3. Reality Gap

We distinguish between the *reality gap* characterizing the difference between the simulated and the real environments, and the *performance gap*, characterizing the difference in performance in simulation and the real world for a given policy.

The reality gap is the gap between the simulated POMDP \mathcal{M}_s and the real one \mathcal{M}_r . It is mainly composed of the dynamics and perception gaps,

$$G_{\text{dyn}}(\mathcal{M}_{s}, \mathcal{M}_{r}) = \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \mathcal{M}_{r}} \left[D\left(T_{\text{sim}}(\cdot \mid \mathbf{s}, \mathbf{a}) \parallel T_{\text{real}}(\cdot \mid \mathbf{s}, \mathbf{a}) \right) \right]$$

$$G_{\text{perc}}(\mathcal{M}_{s}, \mathcal{M}_{r}) = \mathbb{E}_{\mathbf{s} \sim \mathcal{S}_{r}} \left[D\left(O_{\text{sim}}(\cdot \mid \mathbf{s}) \parallel O_{\text{real}}(\cdot \mid \mathbf{s}) \right) \right],$$
2.

where D is a measure of divergence or dissimilarity, $T_{\rm real}$, $T_{\rm sim}$ and $O_{\rm real}$, $O_{\rm sim}$ refer to transition dynamics and observation models with subscripts real and sim indicating the real and simulated environments.

The performance gap is the difference in performance of a given policy when executed in simulation versus in the real world. Let $\pi: \mathcal{B}(\mathcal{S}) \to \mathcal{A}$ denote a policy trained in an environment \mathcal{M} , where $\mathcal{B}(\mathcal{S})$ is the belief space. Starting from an initial state \mathbf{s}_0 , the closed-loop trajectory $\tau = (\mathbf{s}_0, \mathbf{z}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{z}_1, \mathbf{a}_1, \dots, \mathbf{s}_T, \mathbf{z}_T)$ evolves under transition dynamics \mathcal{T} and observation model \mathcal{O} . The expected discounted return of π in environment \mathcal{M} is

$$J_{\mathcal{M}}(\pi) = \mathbb{E}_{\tau \sim p(\tau \mid \pi, \mathcal{M})} \left[\sum_{t=0}^{T} \gamma^{t} R(\mathbf{s}_{t}, \pi(\mathbf{b}_{t})) \right].$$
 3.

The performance gap $G_{\text{perf}}(\mathcal{M}_s, \mathcal{M}_r, \pi)$ for the policy π is formally defined as the absolute difference between its expected performance in the simulated and real environments:

$$G_{\text{perf}}(\mathcal{M}_{s}, \mathcal{M}_{r}, \pi) = |J_{\mathcal{M}_{s}}(\pi) - J_{\mathcal{M}_{r}}(\pi)|.$$
 4.

Minimizing $G_{\text{perf}}(\mathcal{M}_s, \mathcal{M}_r, \pi)$ ensures effective transfer and robust performance from simulation to real-world scenarios. It is important to note that exact replication of real-world dynamics (i.e., elimination of \mathcal{G}_{dyn}) and observation models (i.e., elimination of \mathcal{G}_{obs}) in a simulator is not practically achievable nor required to achieve successful sim-to-real gap minimization. Successful transfer can still occur if the policy is robust against differences between the environments. Therefore, the objective of sim-to-real transfer is

$$\pi^* = \min_{\mathbf{G}} G_{\text{perf}}(\mathcal{M}_s, \mathcal{M}_r, \pi).$$
 5.

3. Sources of Reality Gap

In this section we elaborate on the differences in dynamics (Section 3.1), perception (Section 3.2), actuation (Section 3.3), and system design (Section 3.4), all contributing to the sim-to-real gap.

3.1. Dynamics

One of the most important causes of the reality gap is the dynamics of the system. The dynamics gap $G_{\text{dyn}}(\mathcal{M}_s, \mathcal{M}_r)$ is defined by sim-to-real discrepancies in the transition model \mathcal{T} in the POMDP formulation. When creating a simulator, many decisions and simplifications are made about what to model and what to leave out, how to model dynamics, what

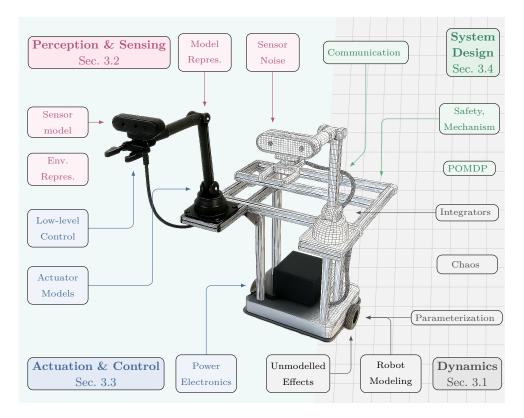


Figure 2

Illustration of the different sources of the reality gap in the real world (left) and simulation (right).

parameters best approximate the real world, and how to represent continuous phenomena in discrete computations. These decisions and assumptions can lead to a plethora of dynamics gaps. A policy trained in a simulation with inaccurate dynamics may learn to exploit these inaccuracies, leading to a potential performance drop in the real world. In the following, we list the most important sources.

Symptom: A policy trained with rigid-body assumptions may lead to unexpected behaviors when dealing with real, deformable objects.

- **3.1.1.** Modeling. Simulators attempt to replicate real-world dynamics with models. The models represent various aspects of physics, such as rigid-body dynamics, batteries, chaotic nature, and stochasticity. Each of these components can contribute to the reality gap.
- Rigid body dynamics. Most robot simulators assume perfectly rigid bodies and joints, but objects can be deformable, and real robots can be flexible, compliant or have uneven joints. For example, real robot links and frames can bend or vibrate under load, whereas simulators usually treat them as rigid bodies. Similarly, simulated joints are often modeled as ideal, but they suffer from damping, internal springs, backlash, etc. This mismatch is a fundamental source of errors in dynamics modeling.
- Chaotic nature. Some real-world phenomena exhibit chaotic behavior, characterized by a sensitive dependence on initial conditions, making them inherently non-reproducible even with perfect models. These are fundamentally difficult to capture with a dynamic

- model in a simulation. For example, atmospheric turbulence, complex flow patterns and pressure waves in fluid dynamics cannot be fully captured by simplified models.
- Stochasticity. The real world contains numerous sources of stochastic dynamics that are fundamentally difficult or impossible to represent in simulation. For instance, ground-based robots (wheeled or legged) encounter stochastic surface and friction variations and debris that create unpredictable disturbances. Simulators typically either ignore these effects entirely or model them as simplified Gaussian noise, failing to capture the complex spatiotemporal correlations and non-linear coupling effects present in reality.
- Battery. This is particularly relevant for *mobile robots*. Even if the actuator model is accurate, the energy source that feeds it can introduce inaccuracies. Since motor torque scales directly with the voltage that the battery provides, any drop in battery voltage immediately reduces the maximum joint torque the hardware can deliver. Hence, rapid joint accelerations can experience a transient torque deficit that is absent in simulation. Batteries are rarely modeled because their behavior is very nonlinear and history-dependent; terminal voltage drops with instantaneous load and varies with cell temperature and age. Neglecting these effects in simulation leaves controllers unaware of the reduced control authority they encounter on real hardware, especially when operating at the limit of the robot's performance.
- Contact Dynamics. A key contributor to the dynamics gap in robotics is the inaccurate modeling of physical contact. Real-world contact interactions are inherently
 complex and nonlinear. Materials deform under pressure, friction varies with relative
 velocity, and contact states can alternate between sticking, slipping, and separation. To
 maintain computational efficiency, simulators typically rely on simplified models such
 as point contacts, linearized friction cones, or compliant spring-damper systems. While
 these approximations enable faster simulation, they often fail to capture critical physical
 behaviors. This can result in non-physical artifacts such as spurious internal forces, unstable grasps, or unrealistic motion patterns. Such mismatches are particularly problematic in contact-rich tasks like robotic manipulation, where precise interaction modeling
 is crucial for successful control and planning.
- **3.1.2. Parameterization.** A fundamental source of the dynamics gap arises from the incorrect parameterization of the physical model. While simulators account for various physical parameters such as friction, aerodynamics, mass, inertia, etc., assigning accurate values to these parameters can be challenging. Some physical properties can be difficult to measure precisely and can be subject to changes over time.
- **3.1.3.** Numerical Integrators. The choice of numerical integration methods for unrolling differential equations significantly impacts the reality gap. Simulators rely on numerical integration schemes to approximate continuous dynamics. Discretization and the specific integration method employed (e.g., Euler, Runge-Kutta, Quadrature, etc) can introduce discrepancies between the simulated and the real system. Additionally, there is a trade-off between fidelity and computation time. Increasing the numerical integrator's accuracy by using smaller time steps or higher-order methods, generally leads to more accurate simulations at the cost of increased computational time.
- **3.1.4.** Human-Robot Interaction. Modeling human behavior in simulation presents unique challenges that create substantial reality gaps (27). Humans exhibit complex, context-

Symptom: Due to chaotic effects, an overfitted policy with slightly different initial conditions might fail in an unpredictable manner.

Symptom:

Unmodeled battery dynamics (e.g., voltage drop with load or age) may cause the policy to produce torques that are less than required to achieve the desired real-world motion.

Symptom: A policy may fail to grasp objects because its learned model is based on simplistic contact assumptions, leading to unexpected slipping or instability.

Symptom: A policy may overshoot or undershoot because its learned dynamics are based on incorrect parameters.

Symptom:

Performance may degrade over long tasks as small integration errors accumulate. Symptom: A policy could take advantage of wrong assumptions in the simulated human behavior.

Symptom:

Performance may drift over time, or the robot may exhibit unexpected vibrations and instability as its physical properties change.

Symptom: A policy trained with idealized digital assets might act on inaccurate geometric assumptions, leading to collisions with unmodeled effects.

dependent, and often irrational behaviors that are difficult to capture with computational models. Therefore, humans are typically simulated as simplified agents with predefined motion patterns, and basic reactive responses to robot actions.

3.1.5. Unmodeled Effects. There is a range of physical phenomena that are often overlooked or simplified in simulation, and that can cause significant sim-to-real differences. These include wear and tear of robot components, which degrade over time due to friction, material fatigue, repetitive usage, etc. These degradations lead to changes in material properties such as stiffness, backlash, vibrations, or texture, which are not typically captured in simulators. Thermal effects can also add to this category, as temperature fluctuations can affect the performance of motors, sensors, batteries, and other components, potentially leading to a significant change in their behavior.

3.1.6. Asset Fidelity. Simulations represent the environment using digital assets to encode geometry, including primitive shapes, meshes, and Signed Distance Fields (SDFs) and material parameters such as friction coefficients, restitution, density, etc. However, for computational efficiency reasons, these representations are often simplified and approximated, utilizing clean layouts and low-resolution models that lack the complexity and properties of real-world environments, such as irregular terrain and fine-grained intricate structures. Similarly, the robot's own geometry is often simplified, omitting important physical details and using simplified shapes for the sake of efficiency, which can lead to unexpected behaviors such as self-collisions or unstable motions in reality.

3.2. Perception and Sensing

In this section, we discuss the sources of the reality gap originating from perception and sensing. The perception gap is defined by sim-to-real discrepancies in the observation model O in the POMDP formulation. State-of-the-art simulators with advanced rendering pipelines (16, 28, 29, 30, 31) have significantly improved the realism of sensor effects. However, they still fall short of capturing the full complexity and variability of the real world, often resulting in discrepancies between simulated and real sensory measurements. Consequently, policies trained in such simplified settings struggle to generalize during real-world deployment. In the following, we outline the major sources of these mismatches and their implications for sim-to-real transfer.

- **3.2.1. Sensor Model.** Simulation models are designed to mimic the physical characteristics of real-world sensors, including properties such as field of view, resolution, and noise patterns. Additionally, they simulate how sensors respond to motion and how they are mounted on the robot since placement significantly influence the resulting sensor data.
- RGB Cameras. For RGB cameras, NVIDIA Isaac Sim (28) uses ray tracing and physically based rendering to generate photorealistic images. CARLA (29) and AirSim (30), both built on Unreal Engine, simulate realistic visuals for navigation. These simulators are capable of modeling realistic lighting, shadows, and material interactions. Many platforms optimized for real-time performance, such as Gazebo Classic (32) and early MuJoCo (19), use simplified OpenGL-based rendering with idealized pinhole camera models and Z-buffer depth computation (33). This omits real-world effects such as lens flares, chromatic aberration, flying pixels, and rolling shutter distortions, causing learned

features to poorly transfer to real-world settings (34).

- Depth Sensors. Many simulators support depth sensor simulation, including stereo cameras, structured light (e.g., Kinect v1), and time-of-flight (e.g., Intel RealSense) systems. Platforms such as Isaac Sim (28), Habitat-Sim (31) offer synthetic depth maps derived from rendered 3D geometry. However, these simulations typically assume ideal depth projections and often omit real-world artifacts such as quantization noise, depth shadows, and ambient light interference (35, 36).
- LiDAR Sensors. Simulators such as CARLA simulate LiDAR sensors using raycasting techniques from the sensor origin to the scene geometry. While CARLA approximates various real-world effects, many intricate characteristics remain difficult to reproduce accurately. These include beam divergence patterns, material-dependent reflectivity, angle-of-incidence effects (37), and sensor-specific interference artifacts observed in real devices like Velodyne, Ouster, or Livox.
- Other Sensors. Simulators also model sensors including joint encoders, IMUs, GPS, and tactile sensors. However, they often idealize sensor behavior, neglecting real-world effects such as IMU drift, GPS multipath, or latency. These simplifications can introduce reality gaps, especially in tasks relying on sensor fusion or precise state estimation.
- **3.2.2. Sensor Noise.** Real-world sensor measurements are inherently noisy due to factors such as thermal effects (38), quantization errors (39), and environmental interference (40). Crucially, sensor noise is often complex, non-Gaussian, state-dependent, temporally correlated, and influenced by motion, temperature, lighting, and surface properties. For example, depth sensors exhibit range-dependent uncertainty, structured missing data at depth discontinuities, and surface-dependent noise patterns (35). Despite these complexities, many simulators use simple Gaussian noise models with fixed variance.
- **3.2.3. Environment Representation.** In Section 3.1, we discussed how asset fidelity affects the accuracy of dynamics simulation. A similar issue arises in perception and sensing: using low-resolution assets, overly simplified scene graphs, or generic materials can fail to capture fine-grained perceptual cues such as surface textures, reflectance, and subtle geometry. Additionally, the lack of High Dynamic Range Image (HDRI) backgrounds may result in unrealistic lighting, while not differentiating between static and dynamic bodies can obscure critical motion and occlusion relationships. These limitations significantly degrade perceptual realism, which becomes particularly problematic in tasks requiring fine-grained object recognition or precise physical interaction.
- **3.2.4. Robot Model.** Simulated robots are typically defined by their geometry, kinematics, and dynamics. These models are often based on CAD files and URDF or USD descriptions. While accurate in structure, they usually simplify or omit important physical details. Realworld factors such as manufacturing tolerances, material wear, and mechanical backlash are rarely modeled. These discrepancies can introduce reality gaps. It can cause self-collisions, unstable motions, or failed task execution (41, 42).
- **3.2.5. Collision Sensing.** Simulators rely on efficient collision detection algorithms to determine contact events between the robot and its environment, or within the robot itself. To achieve real-time performance, they typically use simplified geometric approximations, such as bounding volumes, convex decompositions, or sphere decompositions—instead of

Symptom:

Sensor-specific artifacts (e.g., lens distortion, LiDAR beam patterns) can lead to massive sim-to-real discrepancies in the distribution of observations.

Symptom: Policies trained on simple Gaussian noise may overfit and fail under complex, state-dependent, and temporally correlated real-world sensor noise.

Symptom: The robot may fail to recognize objects or get lost in a real room that looks different from the visually simplistic simulation.

Symptom: When attempting dexterous manipulation, the robot may apply incorrect forces or fail to grasp an object, as it's trained on inaccurate collision models.

high-resolution visual meshes (43, 19). These proxy shapes are evaluated at discrete time steps, which further limits the accuracy of contact modeling, especially in tasks involving fine-grained manipulation or dense contact.

3.3. Actuation and Control

Actuators, together with the low-level control loops around them, are the last interface of the robot's actions and the real world. They turn the actions from the policy into real-world motion and interaction. Even when perception and dynamic models are perfect, any mismatch or gap between what the policy predicts will occur, and what actually happens at this interface can dominate the overall behavior of the robot and can lead to performance degradation. Feasible commands in simulation can become delayed or unstable once they are filtered through real actuators. These discrepancies constitute the *actuation gap*.

Symptom: The robot's movements may be jerky, delayed, or unstable, especially during high-speed maneuvers.

3.3.1. Actuator Models. Most simulators treat motors as first-order systems that are able to perfectly produce torque responses instantaneously and linearly to the command signal. However, real actuators (e.g., motors) behave as higher-order systems whose electrical and mechanical time constants introduce a non-negligible phase lag. In addition, actuators are non-linear due to dead-zones, backlash, slew rate constraints, time constants, hysteresis, etc. If we take into account physical mechanisms attached to the motor, such as gears, these problems only get worse, leading to added delays and increased steady state error.

Symptom: A policy's commands may not produce the expected motion, or may even lead to instability, as they are modified by hidden filters on the real hardware.

3.3.2. Low-level Control. Control policies typically do not produce raw torque/force commands directly, and real robots rarely accept those commands directly. Instead, there exist one or several low-level control layers that drive the actuator low-level control signal (e.g., PWM), and take as input a higher-level signal setpoint (e.g., position or velocity). This conversion is generally done through dedicated hardware and firmware, for which most of the time access is restricted by the vendor. Additionally, there are often hidden filters for antialiasing and resonance suppression, saturation, anti-windup logic, or protective non-linear effects like rate limiting. General robotics simulators do not simulate the lowest-level controllers, since they strongly depend on the manufacturer of the actuator itself.

Symptom:

Fine-grained motions may fail, with jitter or dead zones near zero velocity. **3.3.3.** Power Electronics. Between the low-level controller commands and the input signal to the motors, there is an additional layer: the power electronics. Motor drivers, inverters, and Electronic Speed Controllers close their inner loops, introducing a latency of hundreds of microseconds. Additionally, finite PWM resolution quantizes the commands produced by the actuator, reducing accuracy and introducing additional dead zones near zero crossings. Finally, most drivers come with protection logic, which enforces hard current and voltage caps. These effects are absent in simulations and can largely widen the actuation gap.

3.4. System Design

In addition to the gaps induced by the different modules of the robotics stack, the system design and the choices it entails can influence the reality gap.

3.4.1. Communication. Although communication in simulation is nearly perfect, communication in real-world robotic systems can face multiple challenges, including delays and

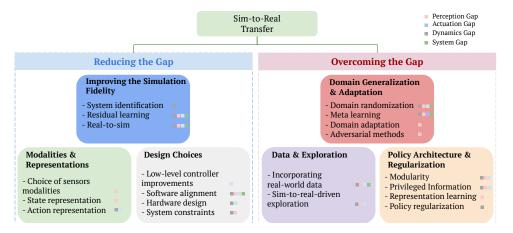


Figure 3

A taxonomy of sim-to-real transfer methods, distinguishing clearly between approaches and techniques that are designed to reduce the reality gap and those designed to overcome it. The colored squares indicate which reality gap each method addresses.

packet loss. In the real world, several mechanisms can be implemented to address these challenges. For instance, it is common to introduce some kind of control attenuation mechanism to gradually bring the robot to a safe stop in case of prolonged communication packet loss. Such mechanisms and behaviors are almost never modeled in simulation.

3.4.2. Safety Mechanisms. Real-world robotic experiments present several safety challenges that are not of concern in simulation. It is very common to implement safety mechanisms such as virtual walls in the real world but not in simulation. Such mechanisms change the behavior of the robot and can further widen the reality gap.

3.4.3. POMDP Formulation. Simulated POMDPs often feature unrealistic information and behaviors. For example, rewards and termination criteria may rely on privileged data, such as exact collisions, not accessible in the real world, leading to different rewards in simulation and reality. This mismatch is particularly problematic for model-based control methods that depend on reward signals at inference time. Similarly, environment resets may use infeasible behaviors, such as placing objects at precise positions with preset velocities, creating stateaction distributions unlike those in real deployment, where such resets are impossible.

3.4.4. Implementation Details. Many implementation details often differ between real-world and simulated environments. Simulation commonly involves the discretization of naturally continuous processes such as image formation and system dynamics. The granularity of these discretizations can significantly impact the reality gap. Low-level control implementation can also involve computational integration and differentiation steps, which can also end up running at different frequencies in simulation and the real world.

www.annualreviews.org • The Reality Gap in Robotics

Symptom:

Communication delays may cause freezing, slowdown, or jerky fallback motions outside the trained policy.

Symptom: A policy may fail to react to safety mechanisms unseen during training in simulation.

Symptom:

Real-world evaluations might strongly differ from the ones in simulation due to differences in the state-action distribution and the reward.

Symptom: A policy may become unstable due to mismatched control frequencies or numerical methods.

4. Existing Solutions

Successful sim-to-real transfer requires careful consideration of the reality gap. Methods addressing this challenge typically target one or more of the previously introduced sources of the reality gap (Section 3). We mainly distinguish between approaches that attempt to reduce the qap by improving different aspects of the simulation (i.e., eliminating sources of the reality gap) and approaches that attempt to overcome the qap by making the system agnostic, adaptive, and/or passive to certain simulation aspects that cannot be accurately modeled. Figure 3 summarizes the different methods that can be used for reducing and overcoming the different reality gaps. A general recipe for successful sim-to-real transfer includes the following steps.

Sim-to-Real Recipe

- 1. Design a simulation that represents all variables relevant for the target appli-
- 2. Attempt to reduce the different components of the reality gap as much as possible.
- 3. Design training methods that can help overcome the remaining gap.
- 4. Train policies in simulation (ideally under massive parallelization).
- 5. Evaluate policies in the target real-world environment.

control frequency, and actuation delay (4, 44, 6, 21).

6. Adjust simulation parameters based on real performance and retrain.

4.1. Reducing the Gap

To reduce the gap, we need to make system modifications that either i) improve the simulation's fidelity (see section 4.1.1), ii) select input and output modalities and representations that have a smaller reality gap (see section 4.1.2), or iii) design systems with components and constraints that induce a smaller gap (see section 4.1.3).

4.1.1. Improving Simulation. It is possible to improve different aspects of the simulated environment, such as its physical fidelity, sensor models, and many other components that influence the gaps introduced in section 3.

- System Identification. has been repeatedly shown to be a crucial aspect for successful sim-to-real transfer in multiple domains such as navigation (8), locomotion (4, 44), and manipulation (45, 46). One recent trend is to do online and iterative system identification with data collected at different iterations of real-world evaluations (47, 48, 49, 50). In addition to physical parameters, it is common to identify and match system-level parameters representing different aspects of a physical environment, such as its latency,
- Learned Residual Models. When discrepancies between the simulated parametric model and the real-world are large, for example when the simulated physics model makes fundamentally-inaccurate assumptions (e.g., rigid-body dynamics for modeling compliant bodies), system identification may be ineffective. Residual simulation, on the other hand, proposes a compelling solution: learn a model that modifies an imperfect simulator, such that the composite dynamics model more accurately reflects the real world. Residual simulation approaches often use learning to modify the outputs of a simulator, namely
- Sys-ID helps reduce the dynamics gap.

Learned residual models can reduce several sources of $reality\ gap.$

the predicted states, in order to better reflect real-world observations. Golemo et al. (51) first trained an Long-Short Term Memory neural network on differences between simulated trajectories and real-world trajectories to improve policy transfer. Ajay et al. (52) trained a stochastic Recurrent Neural Network to augment a deterministic simulator to capture uncertainty in contact dynamics. Gruenstein et al. (53) trained a network to augment an analytical model to capture unmodeled compliance and friction. Bauersfeld et al. (54) trained a neural network to augment a blade-element-momentum aerodynamic model, capturing residual forces and torques for high-speed quadrotor flight. Gao et al. (55) extended residual simulation to soft robotics, training a network to apply corrective forces to a finite-element model of a pneumatic arm. Other residual simulation efforts have proposed a learning technique to modify the *inputs* of a simulator, namely the applied actions, so that the resulting states more closely reflect the real world. For instance, Christiano et al. (56) trained policies for a manipulator in simulation, and afterwards learned an inverse dynamics model that maps a simulated action to a corrected real-world action that produces the state transition intended by the simulator. Hanna et al. (57) took a similar approach, but also augmented the simulator with the action transformation model before training the policy.

• Real-to-Sim First. A variety of techniques have aimed to use real-world data to construct visually and dynamically accurate simulation environments on the fly at test time (58, 59, 60, 61, 62, 46, 63, 64, 65, 66, 67, 68). Real-to-sim environment creation involves creating environments in simulation from data collected in the real world \(\mathcal{D}_{real} = \{(z, a, z')_i\}_{i=1}^N \). This "inverse" problem has multiple facets – constructing accurate geometry of the environment, constructing accurate kinematics and dynamics, and lastly constructing accurate visuals and rendering.

For geometry identification, techniques for real-to-sim often build on 3D reconstruction (69, 70) and novel-view synthesis methods for reconstructing geometry from multi-view data. Alternatively, a more recent class of monocular-3D methods has used learning-based priors to generate entire 3D meshes from single RGB images of objects (71).

In terms of identifying kinematics and dynamics, there is a range of techniques that have been proposed to take constructed geometries and imbue them with physical properties needed for manipulation. Kinematic identification involves accurately inferring object and scene articulation, and degrees of freedom from real data. This is particularly challenging when each of these DoFs is not directly activated during data collection (67). A variety of recent methods (66, 72, 68) have attempted to infer articulation from video or image sequences using foundation models. While they provide a reasonable starting point, there is still a considerable gap with respect to the real world. Finally, to obtain accurate environment renderings, novel-view synthesis methods have been particularly effective (73, 74). These techniques enable high-fidelity neural rendering learned from purely real-world data, while enabling 3D, multiview image queries.

4.1.2. Choice of Modalities and Representations. For most robotic tasks, it is possible to define different observation and action spaces. This choice of interface defines the potential behaviors of the resulting system and can have strong implications on the reality gap (25, 75, 76, 77, 78, 79, 80, 81).

• Choice of Modalities. Different sensors and actuators can yield different gaps (82, 83). For instance, using depth observations, or point clouds yields a smaller reality gap than images, and this approach is a common practice in the field (84, 75, 79, 80, 81). RGB

Real-to-sim can reduce various kinds of *reality gap*. Choice of modalities & representations affects the perception gap.

Choice of action representations affects the *actuation* gap.

Low-level controller improvements can reduce the actuation gap.

Software stack alignment can reduce both reality gap and performance gap.

Hardware design can affect the of dynamics gaps and the actuation gap. images are hard to render photo-realistically, and often include artifacts and features that are very complex to model, such as the ones discussed in section 3.2.1.

- State Representations. Similar to sensor modalities, the choice of state representations can strongly influence the reality gap (85, 86, 87, 88, 89). While using inferred depth maps can be a popular choice in the literature, other representations such as keypoint detections (87), visual feature tracking (90) or foundation model embeddings (89) have also been proved to be good abstractions for reducing the sensory reality gap.
- Action Representations. The action space plays a crucial role in reducing the simto-real gap as demonstrated across robotics domains including navigation (91), locomotion (92), and manipulation (25). For instance, for manipulation, previous studies have shown the advantages of using joint velocity control spaces for sim-to-real transfer (25).

4.1.3. Design Choices. Another important factor to reduce the reality gap is the system design and implementations. When designing a robotic system, multiple choices need to be made depending on the task requirements. This design, however, strongly influences the transferability of any behavior from simulation to the real world. Hence, it is quite important to consider the reality gap when designing the system, including its hardware, the controller's implementations, constraints, and even the software stack.

- Low-level Controller Improvements. While high-level policy learning typically takes place in simulation, the final interaction with the real world is governed by low-level controllers. Improving the robustness and fidelity of these controllers is therefore crucial for overcoming the reality gap. One effective approach is to increase the control frequency or bandwidth, allowing the system to respond more quickly to discrepancies and better handle latency, noise, or unmodeled dynamics. For example, Zhang et al. (93) show that high-frequency impedance controllers can significantly improve sim-to-real transfer in manipulation tasks by stabilizing behavior under real-world disturbances. Such improvements on the low-level controllers can make the system more resilient to imperfections in both the simulator and the real-world environment.
- Software Stack Alignment. Beyond physical realism, achieving successful sim-to-real transfer also requires consistency in the software stack between simulation and real-world deployment. Even small discrepancies, such as mismatched control rates, missing filters, or unmodeled safety checks, can lead to unexpected behavior and degraded performance. Replicating consistent software components ensures that the policy is exposed to the same control dynamics during both training and real-world execution.
- Hardware Design. The design of the robot itself can also significantly influence the sim-to-real gap. Hardware choices that simplify modeling or improve physical robustness can make it easier to simulate the system accurately and improve robustness to discrepancies. For example, using actuators with low latency and consistent torque output, or sensors with well-characterized noise profiles, can improve the fidelity of simulation and reduce the need for complex system identification. Additionally, hardware with passive stability, compliant actuators, or simple kinematics can better tolerate control and perception errors, which could further improve the chances of successful transfer. These principles align with recent advances in real-world imitation learning, such as (94), where careful co-design of hardware and learning pipelines enables effective policy learning.
- Constraining System Dynamics. Another practical strategy for reducing the reality gap is to initially limit the robot's dynamic complexity during deployment. By operating at lower speeds or avoiding aggressive maneuvers, the system becomes less sensitive to

modeling inaccuracies, actuator delays, and perception noise. Such non-dynamic behaviors reduce the burden on both the controller and simulator, improving robustness under real-world uncertainties. For example, Chen et al. (95) show that constraining motion to quasi-static regimes can lead to more reliable sim-to-real transfer in manipulation tasks.

Constraining system dynamics can reduce the dynamics gap and the perception gap.

4.2. Overcoming the Gap

In the previous section, we discussed different paradigms to reduce the reality gap with different measures to bring the simulated and real-world environments closer together. An orthogonal family of methods assumes a fixed reality gap and attempts to overcome it by either making the policies agnostic to the choice of models and parameters or making the policy capable of detecting and reacting to different physical and systematic parameters.

4.2.1. Domain Generalization and Adaptation. One of the most common approaches to overcome the reality gap is to expose the policy to a large variation of the system and dynamics parameters during training. As a result, the policy should be robust to different instantiations of these parameters, including the ones representing the real world.

- Domain randomization (DR). Among the most popular and widely-used approaches to achieve this is DR. In DR, we train the policy in simulation using a broad distribution of simulated environments spanning different simulation parameters such as visual (more generally sensory) parameters like texture and lighting, observation and action noise, physics parameters such as mass and inertia, system delays, object properties, and others. Consequently, the policy can learn a behavior that is applicable to such a large distribution of environments. DR mitigates the need for painstaking system identification and simulator calibration (96). One of the earliest applications of DR was to drone navigation (11). The paradigm has gained significant popularity ever since and has been used to produce some of the most impressive robotics milestones, including dexterous manipulation (3), quadruped locomotion (4, 44), and champion-level drone racing (8). To further improve traditional DR, multiple approaches have been proposed to automate the choice of parameter distributions. For instance, Akkaya et al. (3) proposed automatic DR to progressively expand the range of random dynamics as the policy becomes successful. Tiboni et al. (97) propose to use an offline real-world dataset to estimate the optimal DR ranges allowing to compensate for unmodeled effects. Similarly, Chebotar et al. (48) proposed using real-world data to infer simulation parameters for online training. The lowest-return dynamics from an ensemble effectively acts like adversarial samples for training. For a comprehensive review of DR methods, see (98).
- Adversarial Training systematically generates data to enforce robustness and enhance model resilience beyond standard training. Pinto et al. (99) propose robust adversarial reinforcement learning, where in addition to the main agent, an adversary agent is trained to apply disturbances to destabilize the main agent's policy.
- Meta Learning is another paradigm for domain adaptation. For instance, in (100), the authors propose to use meta-learning for policy learning in simulation under a large distribution of simulation parameters, allowing the policy to internally recognize variations of parameters and act according to its internal understanding. This family of methods leverages meta reinforcement learning to adapt the policy to variations of simulation parameters (101). While domain-randomization-based methods aim to learn a policy that can generalize to a large distribution of simulation parameters, methods such as meta-

Domain randomization can reduce all kinds of reality gaps.

Adversarial training can reduce the actuation gap, perception gap, and dynamics gap.

Meta learning adapts the simulation parameters to reduce the reality qap.

learning aim to adapt the policy as inspired by adaptive control methods (102, 103). One famous example of such methods beyond explicit meta learning, is rapid motor adaptation (RMA) (104). RMA learns an encoder that infers latent representations of the environment's dynamics using privileged information in simulation. Given such an encoder, the policy can adapt its actions based on the inferred latent environment representations.

Domain adaptation can reduce the perception gap.

• Domain Adaptation can similarly be used to enhance the adaptiveness of the policy to variations in environment variables and dynamics (10, 105, 48, 106). Compared to meta learning and RMA, domain adaptation methods for sim-to-real transfer aim to make the policy more robust to the sim-to-real distribution shift. In other words, the main focus of these methods is to enable observation adaptation in the policy and not necessarily adaptation to the changing dynamics.

4.2.2. Data Selection and Exploration. Another important category of methods to overcome the reality gap is to carefully select and curate the training data and exploration strategies used in simulation. These methods focus on generating training data that most closely resemble the target real-world data or data from worst-case behaviors.

- Real world data can be used to reduce all kinds of reality gaps.
- Incorporating real-world data to inform simulation training can greatly improve transfer. For instance, Niu et al. (107) proposed integrating limited real-world data with simulated experiences, while adaptively penalizing learning from simulated state-action pairs that exhibit significant discrepancies from real-world dynamics. Torne et al. (58) proposed a real-to-sim-to-real pipeline that first uses real-world data to train a vision-based policy that is later used to train a teacher policy in simulation. The teacher is then used to train a vision-based student policy in simulation using a mixture of RL with simulated rollouts and behavior cloning using real-world data. This work demonstrated the benefits of co-training with real-world data versus training with only simulation data. Ankile et al. (108) proposed a residual RL scheme to combine real-world and simulated data for vision-based manipulation. Maddukuri et al. (109) demonstrated the effectiveness of co-training vision-based manipulation policies with simulation and real-world data. Co-training has also been shown to be beneficial for training robotic foundation models (110, 111).

Sim-to-real-driven exploration can reduce the perception gap, dynamics gap, and actuation gap. • Sim-to-real-driven exploration methods aim to explore the state-action space in ways that expose the policy to interactions likely to enhance its transfer from simulation to the real world. For instance, Liang et al. (112) proposed learning exploration policies that are executed in the real world to identify critical system parameters. Given this improved model of the environment, they perform trajectory optimization to solve the downstream tasks in the real-world environment. Another approach is to leverage the simulation to learn exploration policies that are transferred to the real world for the sole purpose of learning and fine-tuning policies with real-world interactions (113, 46).

4.2.3. Policy Architecture and Regularization. In addition to the data a policy is exposed to at training time, the policy's architecture and constraints can play a crucial role in the transfer to the real world. In this section, we describe different ways to structure the policy and regularize it in a way that helps sim-to-real transfer.

• Modularity of the system and policy architecture has also been shown to be beneficial for sim-to-real transfer. Clavera et al. (114) proposed decomposing the system into

distinct modules, where vision and low-level control are managed independently from the policy. Their work demonstrated robust transfer of pushing policies. Mueller et al. (115) proposed a similar architecture for autonomous driving. Similarly, Zhang et al. (116) propose learning perception and control networks with different losses and fine-tuning the end-to-end policy network combining perception and control modules using a weighted loss. Julian et al. (117) proposed a hierarchical architecture, learning several low-level skills and a high-level policy coordinating them. By decomposing the policy into multiple skills, they reduce the amount of data needed to generalize to a new domain and environment, such as the real world.

- Privileged information is available to the agent at training time in simulation but not during deployment in the real world. Such information can boost learning efficiency and enable larger-scale training in simulation (118, 119, 120). One popular example of such method is the previously discussed RMA algorithm (104). RMA uses privileged information to train a latent space online as part of a system identification module. Pinto et al. (121) propose an asymmetric actor-critic architecture, where the critic is conditioned on privileged information, while the actor policy is conditioned on true observations. Radosavovic et al. (7) trained a teacher policy for humanoid locomotion with privileged information and then distilled the teacher behavior into a student agent conditioned on the observations. To overcome student-teacher asymmetry, they propose a loss combining RL and teacher distillation, similar to the approach previously proposed in (122) for block picking. The main difference between these two methods is how they combine RL and imitation learning losses using either fixed coefficients (122) or a schedule (7). More recently, Krinner et al. (123) proposed using privileged state information in state-space world models to improve the training efficiency of a model-based RL agent. The proposed approach was successfully demonstrated on vision-based high-speed drone racing.
- Representation learning leverages self-supervised objectives to learn representations that can be better suited for policy search and sim-to-real transfer. By carefully designing the representation learning loss, we can enforce the learned representation properties that are desirable to overcome certain sources of the reality gap. While domain randomization and adaptation methods improve domain robustness via exposing the policy to a wider set of data with the hope of out-of-distribution generalization, representation learning methods leverage loss functions to learn robust features. For instance, Tanwani et al. (124) proposed an approach that leverages real-world data to align the distributions of the state representations in simulation and the real world, making the features more robust to the corresponding domain shift. Yoneda et al. (125) propose to use adversarial training to adapt the observation encoder learned in simulation once it encounters real-world samples. This process is combined with a dynamics consistency loss to ensure that latent transitions in the target domain remain faithful to the dynamics learned in the source domain, thereby preserving policy effectiveness despite visual discrepancies. Xing et al. (126) proposed a contrastive learning approach to learn background-agnostic and task-relevant representations, ensuring effective feature learning for the downstream task while ignoring irrelevant, noisy background information.
- Policy regularization can change the behavior of the policy in a manner that helps overcome the reality gap for sim-to-real transfer. Regularization typically occurs through loss functions or elaborate reward designs for RL agents. For instance, it has become a common practice to penalize the magnitude of actions and consecutive action differences through reward terms or penalties (44, 21, 127, 128, 25). Some works even penalize

Modularity can reduce the perception gap, dynamics gap, and actuation gap.

Privileged information addresses the perception gap, dynamics gap, and actuation gap.

Representation learning can help reduce the perception gap.

Policy regularization can reduce the *dynamics qap*.

the magnitudes of explicit measures such as velocity and accelerations (128, 44), as well as power loss (129). While all of these works rely on RL penalties to enforce desirable properties such as smoothness on the learned policies, such policies can also be imposed using loss functions. For instance, Mysore et al. (130) introduced smoothness losses to the policy training algorithm to minimize the temporal and spatial Lipschitz constraints of a quadrotor control policy function. Their work demonstrated a clear advantage of loss function regularization over reward engineering. Chen et al. (131) presented similar findings for humanoids locomotion. While these methods enforce properties on the policy to overcome the reality gap, Niu et al. (107) proposed incorporating real-world data during training to penalize the action-value function in simulation when interactions exhibit significant discrepancies in dynamics between the simulated and real environments.

5. Evaluation Metrics

Research on sim-to-real transfer has leveraged various metrics for evaluation. In this section, we survey such metrics while making a clear distinction between metrics evaluating the reality gap itself and metrics that evaluate the sim-to-real transfer performance.

5.1. Assessing the Reality Gap

A well-designed evaluation framework for assessing the gap between simulation and the real world is critical for reducing the reality gap, as well as for understanding, diagnosing, and ultimately improving the transferability of learned policies.

• Sim-to-real Correlation Coefficient. One key question in sim-to-real transfer is whether performance improvement observed in simulation reliably leads to better performance in the real world. To address this, Kadian, et al. (132) introduced the sim-to-real Correlation Coefficient (SRCC). SRCC is defined as the Pearson correlation coefficient between the performance metrics of the agents in simulation and the real world,

$$SRCC = \frac{\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2}},$$
6.

where x_i and y_i denote the task evaluation performance (e.g., success rates) of the i-th policy in simulation and the real world, respectively, and \bar{x} , \bar{y} are their corresponding means. A coefficient close to +1 indicates a strong positive correlation, suggesting that the simulation performance is a good predictor of real-world performance. In contrast, values near 0 imply poor correlation, meaning simulation results offer little guidance for real-world evaluation. Importantly, a simulator with high average real-world performance but low SRCC is still problematic. Without a reliable correlation, improvements in simulation may have unpredictable effects in the real world, making it difficult to make informed design decisions. Most of the changes need to be validated on a physical robot, which defeats the purpose of using simulation to accelerate development.

• Offline Replay Error. When direct real-world policy deployment is not feasible, offline replay error provides a practical alternative for evaluating sim-to-real transfer. This situation often arises during early prototyping stages or when deployment is limited by cost, safety concerns, or hardware availability. Offline replay error compares state

trajectories between a real-world trajectory and the equivalent simulation trajectory when replaying the real-world actions in an open-loop fashion (25). Formally, given a trajectory of real-world policy rollout states $\{\mathbf{s}_t^{\text{real}}\}$ and corresponding actions $\{a_t\}$, the offline replay error is defined as

$$\mathcal{E}_{\text{replay}} = \frac{1}{T} \sum_{t=1}^{T} \|\mathbf{s}_{t}^{\text{sim}} - \mathbf{s}_{t}^{\text{real}}\|^{2},$$
 7.

where T is the length of the trajectory, and $\mathbf{s}_t^{\text{sim}}$ is obtained by rolling out the real-world actions in an open-loop fashion in simulation. This metric is appealing due to its simplicity and low cost: no real-time interaction is required, and evaluation can be done offline using logged data. It provides a quick diagnostic of how well the simulation represents the target real-world environment under the distribution induced by the policy. A policy having a high offline replay error is a strong indicator of a large reality gap.

• Visual Fidelity Analysis. Assessing the perceptual domain gap between simulated and real-world environments is particularly relevant for visuomotor policies that map visual observations directly to control commands, as well as visual state representations methods. To quantify the visual similarity, a variety of metrics have been proposed that operate at either the pixel level or in the space of learned image embeddings. These include both distribution-level metrics, which assess the global characteristics of image sets, and single-image metrics, which compare individual simulated images to corresponding real ones (133). For distribution-level evaluation, commonly used metrics include Inception Score (IS) (134), Fréchet Inception Distance (FID) (135), and Kernel Inception Distance (KID) (136), TSNE dimensionality reduction (137, 138, 126), all of which operate on feature representations extracted from pretrained networks. For single-image evaluation, metrics such as the Structural Similarity Index (SSIM) and Peak Signal-to-Noise Ratio (PSNR) (134), Instance Performance Difference (IPD) (139) are used to measure differences in luminance, contrast, and structural details.

5.2. Assessing Sim-to-Real Transfer

While evaluating the reality gap is very valuable to analyze the limitations of the simulation and further improve it, metrics that evaluate the sim-to-real transfer performance under a fixed reality gap are also necessary. This category includes metrics such as success rate, cumulative reward (in reinforcement learning), and task-specific indicators tailored to different domains. In this section, we will discuss in detail the evaluation metrics commonly used to quantify sim-to-real transfer performance.

• Success rate measures the proportion of trials in which a policy successfully completes the intended task, written as Success Rate = $\frac{N_{\rm success}}{N_{\rm total}}$. It is widely used as a simple and reliable indicator of sim-to-real transfer effectiveness. This measure is most informative when proper randomization is performed on all involved variables, such as the initial environment state and simulation seeds, among many others. Success rates are commonly reported in diverse domains, including manipulation (140, 23, 26, 14, 77, 141, 142, 143), navigation (144, 145, 132, 146), autonomous racing (8, 147), legged locomotion (104, 44, 104). High task success rates in the real world often indicate that a policy has successfully transferred key competencies learned in simulation. However, such metrics are often binary or aggregate and do not reveal where or why the transfer might have

- failed. For example, two policies with the same success rate might differ significantly in robustness, and some of the failure modes can be critical in real-world deployment.
- Cumulative reward. For policies learned using reinforcement learning (RL) or with a predefined reward structure, the cumulative reward measures how well a policy achieves long-term objectives by summing the reward signal over time: $R = \sum_{t=0}^{T} r_t$, where r_t is the reward at timestep t and T is the total duration of the episode. Unlike binary success metrics, it can capture the progress and efficiency of the policy rollouts, offering a more fine-grained view of sim-to-real performance. However, its interpretability depends on consistent and well-designed reward functions across simulation and real-world domains (148, 113, 48). This can make the comparison of different sim-to-real techniques under varying RL configurations difficult, especially when reward formulations differ across tasks or platforms.
- Task-specific metrics. In addition to general success or reward-based metrics, most of the successful sim-to-real studies evaluate performance using task-specific criteria tailored to the target domain. For example, navigation tasks may use path efficiency or time-to-goal (145), while manipulation tasks typically use object-centric metrics, such as object-distance-to-goal for object pushing (25). These metrics offer fine-grained insights into policy behavior and failure modes that are not captured by success rate alone. However, for real-world deployment, it will be challenging to provide standardization across benchmarks, which can hinder fair comparison between methods.

6. Discussion and Open Problems

Despite the reality gap and the challenges it creates, sim-to-real transfer has been a very successful and popular paradigm in robotics. However, it is unclear whether simulation will remain a major tool for robotics development as many challenges persist. We believe that there is still untapped potential for simulation in robotics and discuss some of the challenges and opportunities next.

6.1. Wrong Models, Better Controllers

As briefly mentioned in section 2.3, it is more important in practice to reduce the sim-to-real performance gap than to reduce the reality gap. This motivates the question of whether accurate physics modeling is needed to perform reliable model-based control. Model-based approaches, such as model-based RL and model predictive control (MPC) tend to be more data efficient than model-free RL and thus leveraging them in sim-to-real contexts can decrease the scale of data generation necessary for sufficiently reliable control. Particularly, researchers have examined how training for model-based RL can be adapted to not focus on accurately capturing the underlying physics, but instead learning a dynamics model that improves control performance (149, 150). Results show that focusing on learning a model accurately near high-return areas outperforms the alternative of trying to learn a model with uniform accuracy across the state-action space (149). Guzman et al. (150) examine the use of Bayesian optimization for MPC, where they sample different physical parameters in simulation for tuning the MPC controller. Results show that optimizing for control performance outperforms planning with the most likely model estimate. As such, an open question remains as to how a robot can most efficiently use a simulator with incorrect parametrization to learn either a policy or stochastic world model for use in generating robust, real-world performance.

6.2. Differentiable Simulators.

Differentiable simulators offer the computation of gradients of quantities involved in the simulation which allows them to be integrated in gradient-based optimization workflows (151). While traditional simulators, such as MuJoCo (19) and Bullet (43) can often be useful in such applications by leveraging finite differences, a differentiable simulator typically offers fast and analytically correct gradients. The methods to compute such derivatives typically fall into two categories: automatic differentiation leveraging the chain rule of differentiation, and manual computed analytical gradients. The most common form of Autodiff, reverse-mode automatic differentiation, accumulates gradients for the inputs from each operation, given the output gradients. Autodiff can be realized through a tape to record the operations of the computation path which allows the (reverse) playback in the backward pass. Frameworks, such as NVIDIA Warp (152), Taichi (153), JAX (154), and PyTorch's compilation option (155) allow code to be generated for the backward pass, which is critical for performance. As such, an open question remains as to how to best leverage differentiable simulation and augment it with learning-based dynamics model.

6.3. Video and World Models

Video models are primarily designed to process, generate, or predict sequences of visual frames. Their core objective is to model temporal dynamics and evolving visual content within video data. These models address a wide range of tasks, including future frame prediction based on observed sequences and conditional video generation from text, actions, or other modalities (156). World models pursue a broader goal: learning internal representations of an environment that enable simulation and prediction of future states in response to agent actions or external events (157, 158, 159). These models seek to encode the causal structure of the world to support planning, imagination, and counterfactual reasoning (160). Despite recent progress, key challenges remain with significant opportunities for future research. Video models, while effective at generating dynamic visual content, struggle with maintaining temporal consistency, physical plausibility, fine-grained controllability, and computational efficiency—issues that hinder their reliability in robotics and simulation. World models offer a promising alternative to white-box simulators and can be learned using real-world data, potentially creating a smaller reality gap. However, they face problems such as compounding errors in long-term predictions, poor generalization to new environments, difficulty balancing abstraction with realism, and high data requirements. Addressing these challenges will require innovations in model architecture, integration of physical priors, and improved training strategies. In future iterations, the paradigms of world modeling and simulation may not be so distinct, with simulators naturally providing data to bootstrap world models for deployment in reality.

6.4. Simulation-Based Inference

Simulation-based inference (161) is a statistical technique to approximate the posterior distribution of simulation parameters θ . Given a *prior* distribution $p(\theta)$, simulation dynamics model $\mathcal{T}_s(s_{t+1}|s_t, a_t)$, a set of simulation observations $\{\mathbf{s}_i^s\}_{i=1}^S$ and a set of real state observations $\{\mathbf{s}_i^s\}_{i=1}^R$, the posterior can be computed following the Bayes' rule as

 $p(\theta|\{\mathbf{s}_i^{\mathrm{r}}\}_{i=1}^{\mathrm{R}}, \{\mathbf{s}_i^{\mathrm{s}}\}_{i=1}^{\mathrm{S}}) \propto p(\{\mathbf{s}_i^{\mathrm{r}}\}_{i=1}^{\mathrm{R}}|\theta)p(\theta)$. The main challenge is that the likelihood term is generally intractable. If we consider simulation as a generative process, $\mathbf{s}^{s} = g(P_{s}, \theta)$, where s^{s} are simulated states or observations, the likelihood term can be defined as $p({\bf s}_i^{\rm r})_{i=1}^{\rm R}|\theta) = \int p({\bf s}_i^{\rm r})_{i=1}^{\rm R}, {\bf s}^{\rm s}|\theta) d{\bf s}^{\rm s}$. To compute this explicitly would require integrating over all possible trajectories the simulator could generate for a specific parametrization θ . This is not feasible for the majority of robotics simulators that involve complex physics models (including contact models), observation models, and numerical solvers. To address this issue, approximate inference methods have been proposed based on Monte Carlo techniques such as Approximate Bayesian Computation (162), and variational inference (163). Also, note that computing the posterior is an ill-posed inverse problem, as there are possibly many simulation parameters that can generate the same trajectories. For example, in a pushing task where a manipulator pushes a block on a table, the resistance of the motion can be attributed to higher mass, higher friction, or both. This leads to posterior distributions that are highly multimodal. To account for generic posteriors and sidestep the approximation of the likelihood term directly, recent works in robotics have used techniques where a neural network is trained to directly output the posterior distribution $p(\theta | \{\mathbf{s}_i^r\}_{i=1}^R, \{\mathbf{s}_i^s\}_{i=1}^S)$. This is known as neural posterior estimation (164, 165) and has been used in robotics recently (166, 167, 168, 169, 170, 171, 172). The posterior can then be used as the randomization distribution for domain randomization. For a review of simulation-based inference methods applied to domain randomization, see (98). We believe simulation-based inference methods will continue to be a popular research topic in robotics.

6.5. Simulation for Large Robotics Models

In recent years, there have been substantial efforts to collect real-world data to train large models for robotics with techniques such as imitation learning (173, 174, 158). However, realistically, real-world data collection of action-labeled datasets is limited by several factors such as human efforts and hardware resources. Simulation represents a great opportunity to augment such efforts with massive-scale synthetic data generation. Many recent works explored this avenue and proposed novel pipelines to procedurally generate simulated data based on real-world demonstrations (175, 176). As with RL, simulated data would need to have a small reality gap to be valuable in real-world deployments. An open question is whether the methods and assumptions needed to reduce these gaps would themselves be limiting factors to the scale of data that can be collected in simulation (under these assumptions and using these methods).

Another recent trend is to leverage simulation to evaluate real-world policies in a systematic and reproducible manner (177, 178). In this context, a proper calibration of the reality gap is crucial for simulation to be a good proxy for the real world. While methods to reduce this gap are very similar to the ones used for sim-to-real transfer, an open question is how to overcome the reality gap for the specific downstream purpose of model evaluations such that the performance of a policy in simulation matches the performance in the real environment.

7. Conclusion

This survey is an attempt at dissecting the very complex yet very important problem of the reality gap in robotics. We discussed the sources of these gaps, the problems they create and solutions to alleviate them, together with evaluation metrics and opportunities for future research. By better understanding the problem, we can more clearly situate ourselves as a community to face future research challenges. Leveraging simulation to the best of our capabilities will enable a cost-effective alternative for the development of the future generation of robotics systems.

DISCLOSURE STATEMENT

The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

AUTHOR CONTRIBUTIONS

EA and FR defined the structure of the survey and its focus. EA wrote the abstract, sections 1, 4, and 7 and contributed subsections to sections 2, 3, and 6. He also revised and updated many paragraphs throughout the text. EA and JX made the illustrations. JX wrote sections 3.2, section 5, and many paragraphs throughout the text. AR wrote most of section 2, section 3.1, and section 3.3. IA wrote section 6.3. CG provided general comments and suggestions and updated section 1. EH wrote section 6.2. AG contributed to different parts of section 4. TH wrote section 6.1. YN contributed to section 4.1. DF and DS provided feedback, corrections and suggestions on the content of the article. FR conceptualized the work with EA, wrote section 6.4, and revised and updated several parts of the article.

ACKNOWLEDGMENTS

This work was supported by the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

LITERATURE CITED

- Kirillov A, Mintun E, Ravi N, Mao H, Rolland C, et al. 2023. Segment anything. In Proceedings of the IEEE/CVF international conference on computer vision, pp. 4015–4026
- Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, et al. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774
- 3. Akkaya I, Andrychowicz M, Chociej M, Litwin M, McGrew B, et al. 2019. Solving rubik's cube with a robot hand. $arXiv\ preprint\ arXiv:1910.07113$
- 4. Tan J, Zhang T, Coumans E, Iscen A, Bai Y, et al. 2018. Sim-to-real: Learning agile locomotion for quadruped robots. Robotics: Science and Systems XIV
- Tang B, Lin MA, Akinola IA, Handa A, Sukhatme GS, et al. 2023. IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality. In Proceedings of Robotics: Science and Systems. Daegu, Republic of Korea
- Lee J, Hwangbo J, Wellhausen L, Koltun V, Hutter M. 2020. Learning quadrupedal locomotion over challenging terrain. Science robotics 5(47):eabc5986
- Radosavovic I, Xiao T, Zhang B, Darrell T, Malik J, Sreenath K. 2024. Real-world humanoid locomotion with reinforcement learning. Science Robotics 9(89):eadi9579
- Kaufmann E, Bauersfeld L, Loquercio A, Müller M, Koltun V, Scaramuzza D. 2023. Championlevel drone racing using deep reinforcement learning. Nature 620(7976):982–987

- 9. Muratore F, Gienger M, Peters J. 2019. Assessing transferability from simulation to reality for reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*
- Rusu AA, Večerík M, Rothörl T, Heess N, Pascanu R, Hadsell R. 2017. Sim-to-real robot learning from pixels with progressive nets. In CoRL, pp. 262–270. PMLR
- 11. Sadeghi F, Levine S. 2017. Cad2rl: Real single-image flight without a single real image. Robotics: Science and Systems
- Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 23–30. IEEE
- Nagabandi A, Clavera I, Liu S, Fearing RS, Abbeel P, et al. 2018. Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning. In International Conference on Learning Representations
- Andrychowicz OM, Baker B, Chociej M, Jozefowicz R, McGrew B, et al. 2020. Learning dexterous in-hand manipulation. The International Journal of Robotics Research 39(1):3–20
- Zhuang Z, Fu Z, Wang J, Atkeson C, Schwertfeger S, et al. 2023. Robot Parkour Learning. In Conference on Robot Learning (CoRL)
- Mittal M, Yu C, Yu Q, Liu J, Rudin N, et al. 2023. Orbit: A unified simulation framework for interactive robot learning environments. IEEE Robotics and Automation Letters 8(6)
- 17. Makoviychuk V, Wawrzyniak L, Guo Y, Lu M, Storey K, et al. 2021. Isaac gym: High performance gpu-based physics simulation for robot learning. arXiv preprint arXiv:2108.10470
- Liang J, Makoviychuk V, Handa A, Chentanez N, Macklin M, Fox D. 2018. Gpu-accelerated robotic simulation for distributed reinforcement learning. In CoRL. PMLR
- 19. Todorov E, Erez T, Tassa Y. 2012. Mujoco: A physics engine for model-based control. $IEEE/RSJ\ IROS$
- Rudin N, Hoeller D, Reist P, Hutter M. 2022. Learning to walk in minutes using massively parallel deep reinforcement learning. In Conference on Robot Learning, pp. 91–100. PMLR
- Song Y, Romero A, Müller M, Koltun V, Scaramuzza D. 2023. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. Science Robotics 8(82):eadg1462
- Zhang M, Ma Y, Miki T, Hutter M. 2024. Learning to Open and Traverse Doors with a Legged Manipulator. In 8th Annual Conference on Robot Learning
- 23. Fu Z, Cheng X, Pathak D. 2023. Deep whole-body control: learning a unified policy for manipulation and locomotion. In Conference on Robot Learning, pp. 138–149. PMLR
- 24. Grandia R, Knoop E, Hopkins MA, Wiedebach G, Bishop J, et al. 2024. Design and Control of a Bipedal Robotic Character. In Proceedings of Robotics: Science and Systems
- 25. Aljalbout E, Frank F, Karl M, van der Smagt P. 2024. On the role of the action space in robot manipulation learning and sim-to-real transfer. *IEEE Robotics and Automation Letters* 9(6)
- Alles M, Aljalbout E. 2022. Learning to centralize dual-arm assembly. Frontiers in Robotics and AI 9:830007
- 27. D'Ambrosio DB, Jaitly N, Sindhwani V, Oslund K, Xu P, et al. 2023. Robotic Table Tennis: A Case Study into a High Speed Learning System. In Robotics: Science and Systems
- 28. NVIDIA. 2023. Isaac sim documentation. https://docs.omniverse.nvidia.com/isaacsim/latest/. Accessed: 2025-05-26
- 29. Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V. 2017. CARLA: An open urban driving simulator. In Conference on robot learning, pp. 1–16. PMLR
- Shah S, Dey D, Lovett C, Kapoor A. 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In Field and Service Robotics: Results of the 11th International Conference, pp. 621–635. Springer
- 31. Savva M, Kadian A, Maksymets O, Zhao Y, Wijmans E, et al. 2019. *Habitat: A platform for embodied ai research*. In *IEEE/CVF international conference on computer vision*
- 32. Koenig N, Howard A. 2004. Design and use paradigms for Gazebo, an open-source multi-robot

- simulator. In IEEE/RSJ IROS
- 33. Catmull EE. 1974. A subdivision algorithm for computer display of curved surfaces. The University of Utah
- 34. Tremblay Jea. 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In CVPR Workshops
- 35. Nguyen CV, Izadi S, Lovell D. 2012. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In 2012 second international conference on 3D imaging, modeling, processing, visualization & transmission, pp. 524–530. IEEE
- Huang J, Valentin J, et al. 2019. DeepFusion: Real-Time Dense 3D Reconstruction for Monocular SLAM Using Single-View Depth and Gradient Predictions. In CVPR
- Laconte J, Deschênes SP, Labussière M, Pomerleau F. 2018. Lidar measurement bias estimation via return waveform modelling in a context of 3d mapping. arXiv preprint arXiv:1810.01619
- Susperregi L, Sierra B, Castrillón M, Lorenzo J, Martínez-Otzeta JM, Lazkano E. 2013. On the use of a low-cost thermal sensor to improve kinect people detection in a mobile robot. Sensors 13(11):14687–14713
- White J, Proakis JG. 2008. Digital Signal Processing: Principles, Algorithms, and Applications. Prentice Hall
- Barshan B, Durrant-Whyte H. 1995. Noise characterization of an ultrasonic sensor. IEEE Transactions on Instrumentation and Measurement 44(3):764-768
- 41. Peng XB, Coumans E, Zhang T, Lee TWE, Tan J. 2020. Learning Agile Robotic Locomotion Skills by Imitating Animals. In Robotics: Science and Systems (RSS)
- 42. Arm P, Mittal M, Kolvenbach H, Hutter M. 2024. Pedipulate: Enabling manipulation skills using a quadruped robot's leg. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 5717–5723. IEEE
- 43. Coumans E, Bai Y. 2016–2021. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org
- 44. Hwangbo J, Lee J, Dosovitskiy A, Bellicoso D, Tsounis V, et al. 2019. Learning agile and dynamic motor skills for legged robots. *Science Robotics* 4(26):eaau5872
- 45. Golemo F, Taiga AA, Courville A, Oudeyer PY. 2018. Sim-to-real transfer with neural-augmented robot simulation. In Conference on Robot Learning, pp. 817–828. PMLR
- Memmel M, Wagenmaker A, Zhu C, Fox D, Gupta A. 2024. ASID: Active Exploration for System Identification in Robotic Manipulation. In The Twelfth International Conference on Learning Representations
- 47. Farchy A, Barrett S, MacAlpine P, Stone P. 2013. Humanoid robots learning to walk faster: From the real world to simulation and back. In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, pp. 39–46
- Chebotar Y, Handa A, Makoviychuk V, Macklin M, Issac J, et al. 2019. Closing the sim-toreal loop: Adapting simulation randomization with real world experience. In 2019 International Conference on Robotics and Automation (ICRA), pp. 8973–8979. IEEE
- Allevato AD, Schaertl Short E, Pryor M, Thomaz AL. 2020. Iterative residual tuning for system identification and sim-to-real robot learning. Autonomous Robots 44(7):1167–1182
- 50. Du Y, Watkins O, Darrell T, Abbeel P, Pathak D. 2021. Auto-tuned sim-to-real transfer. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 1290–1296
- Golemo F, Martín-Martín R, Pal C, Meger D. 2018. Sim-to-Real Transfer with Neural-Augmented Robot Simulation. In Conference on Robot Learning (CoRL)
- 52. Ajay A, Wu J, Fazeli N, Bauza M, Kaelbling LP, et al. 2018. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3066–3073. IEEE
- Gruenstein J, Chen T, Doshi N, Agrawal P. 2021. Residual model learning for microrobot control. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pp.

- 7219-7226. IEEE
- Bauersfeld L, Kaufmann E, Foehn P, Sun S, Scaramuzza D. 2021. Neurobem: Hybrid aerodynamic quadrotor model. Robotics Science and Systems
- 55. Gao J, Shi Y, Garg A. 2024. Sim-to-real of soft robots with learned residual physics. *IEEE Robotics and Automation Letters (RA-L)*
- Christiano P, Shah Z, Mordatch I, Schneider J, Blackwell T, et al. 2016. Transfer from simulation to real world through learning deep inverse dynamics model. arXiv preprint arXiv:1610.03518
- Hanna J, Stone P. 2017. Grounded Action Transformation for Robot Learning in Simulation.
 In AAAI Conference on Artificial Intelligence
- Torne M, Simeonov A, Li Z, Chan A, Chen T, et al. 2024. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. arXiv preprint arXiv:2403.03949
- 59. Torne M, Jain A, Yuan J, Macha V, Ankile L, et al. 2024. Robot learning with super-linear scaling
- Dan P, Kedia K, Chao A, Duan EW, Pace MA, et al. 2025. X-sim: Cross-embodiment learning via real-to-sim-to-real
- Jiang H, Hsu HY, Zhang K, Yu HN, Wang S, Li Y. 2025. Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos
- 62. Patel S, Yin X, Huang W, Garg S, Nayyeri H, et al. 2025. A real-to-sim-to-real approach to robotic manipulation with vlm-generated iterative keypoint rewards. arXiv preprint
- 63. Huang P, Zhang X, Cao Z, Liu S, Xu M, et al. 2023. What Went Wrong? Closing the Sim-to-Real Gap via Differentiable Causal Discovery. In Conference on Robot Learning (CoRL)
- 64. Pfaff N, Fu E, Binagia J, Isola P, Tedrake R. 2025. Scalable real2sim: Physics-aware asset generation via robotic pick-and-place setups
- 65. Chen Y, Li X, Guo S, Ng XY, Ang M. 2022. Real2sim or sim2real: Robotics visual insertion using deep reinforcement learning and real2sim policy adaptation
- 66. Chen Z, Walsman A, Memmel M, Mo K, Fang A, et al. 2024. URDFormer: A Pipeline for Constructing Articulated Simulation Environments from Real-World Images. In Robotics: Science and Systems (RSS)
- 67. Hsu CC, Jiang Z, Zhu Y. 2023. Ditto in the House: Building Articulation Models of Indoor Scenes through Interactive Perception. In IEEE International Conference on Robotics and Automation (ICRA). Code available at https://github.com/UT-Austin-RPL/HouseDitto
- 68. Zhao M, Weng Y, Bauer D, Song S. 2024. Real2code: Reconstruct articulated objects via code generation. arXiv preprint arXiv:2406.08474
- 69. Schönberger JL, Frahm JM. 2016. Structure-from-Motion Revisited. In Conference on Computer Vision and Pattern Recognition (CVPR)
- 70. Schönberger JL, Zheng E, Pollefeys M, Frahm JM. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In European Conference on Computer Vision (ECCV)
- 71. Wang N, Zhang Y, Li Z, Fu Y, Liu W, Jiang YG. 2018. Pixel2mesh: Generating 3d mesh models from single rgb images. In Proceedings of the European conference on computer vision
- Le L, Xie J, Liang W, Wang HJ, Yang Y, et al. 2025. Articulate-Anything: Automatic Modeling of Articulated Objects via a Vision-Language Foundation Model. In International Conference on Learning Representations (ICLR). Poster
- 73. Kerbl B, Kopanas G, Leimkühler T, Drettakis G. 2023. 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. 42(4):139–1
- 74. Mildenhall B, Srinivasan PP, Tancik M, Barron JT, Ramamoorthi R, Ng R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*
- 75. Mahler J, Liang J, Niyaz S, Laskey M, Doan R, et al. 2017. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *Robotics: Science and Systems XIII*
- 76. Tai L, Paolo G, Liu M. 2017. Virtual-to-real deep reinforcement learning: Continuous con-

- trol of mobile robots for mapless navigation. In 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 31–36. IEEE
- 77. Ding Z, Tsai YY, Lee WW, Huang B. 2021. Sim-to-real transfer for robotic manipulation with tactile sensory. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6778–6785. IEEE
- Kollar T, Laskey M, Stone K, Thananjeyan B, Tjersland M. 2022. Simnet: Enabling robust unknown object manipulation from pure synthetic data via stereo. In Conference on Robot Learning, pp. 938–948. PMLR
- Miki T, Lee J, Hwangbo J, Wellhausen L, Koltun V, Hutter M. 2022. Learning robust perceptive locomotion for quadrupedal robots in the wild. Science robotics 7(62):eabk2822
- Agarwal A, Kumar A, Malik J, Pathak D. 2023. Legged locomotion in challenging terrains using egocentric vision. In Conference on robot learning, pp. 403–415. PMLR
- 81. Zhu H, Wang Y, Huang D, Ye W, Ouyang W, He T. 2024. Point cloud matters: Rethinking the impact of different observation spaces on robot learning. *Advances in Neural Information Processing Systems* 37:77799–77830
- Zhou B, Krähenbühl P, Koltun V. 2019. Does computer vision matter for action? Science Robotics 4(30):eaaw6661
- 83. Chen B, Sax A, Lewis F, Armeni I, Savarese S, et al. 2021. Robust Policies via Mid-Level Visual Representations: An Experimental Study in Manipulation and Navigation. In Proceedings of the 2020 Conference on Robot Learning, ed. J Kober, F Ramos, C Tomlin, pp. 2328–2346, vol. 155 of Proceedings of Machine Learning Research, pp. 2328–2346
- 84. Loquercio A, Kaufmann E, Ranftl R, Müller M, Koltun V, Scaramuzza D. 2021. Learning high-speed flight in the wild. *Science Robotics* 6(59):eabg5810
- 85. James S, Wohlhart P, Kalakrishnan M, Kalashnikov D, Irpan A, et al. 2019. Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)
- So J, Xie A, Jung S, Edlund J, Thakker R, et al. 2022. Sim-to-Real via Sim-to-Seg: End-to-end Off-road Autonomous Driving Without Real Data. In Conference on Robot Learning
- 87. Zhang Y, Zhang Z, Ke L, Srinivasa S, Gupta A. 2024. ATK: Automatic Task-driven Keypoint selection for Policy Transfer from Simulation to Real World. In CoRL Workshop on Learning Robot Fine and Dexterous Manipulation: Perception and Control
- 88. Ho D, Rao K, Xu Z, Jang E, Khansari M, Bai Y. 2021. Retinagan: An object-aware approach to sim-to-real transfer. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 10920–10926. IEEE
- Silwal S, Yadav K, Wu T, Vakil J, Majumdar A, et al. 2024. What do we learn from a large-scale study of pre-trained visual representations in sim and real environments? In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 17515–17521. IEEE
- 90. Kaufmann E, Loquercio A, Ranftl R, Müller M, Koltun V, Scaramuzza D. 2020. Deep Drone Acrobatics. In Proceedings of Robotics: Science and Systems. Corvalis, Oregon, USA
- Kaufmann E, Bauersfeld L, Scaramuzza D. 2022. A benchmark comparison of learned control policies for agile quadrotor flight. In 2022 International Conference on Robotics and Automation (ICRA), pp. 10504–10510. IEEE
- 92. Kim D, Berseth G, Schwartz M, Park J. 2023. Torque-based deep reinforcement learning for task-and-robot agnostic learning on bipedal robots using sim-to-real transfer. *IEEE Robotics and Automation Letters* 8(10):6251–6258
- Zhang X, Tomizuka M, Li H. 2024. Bridging the sim-to-real gap with dynamic compliance tuning for industrial insertion. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 4356–4363. IEEE
- 94. Chi C, Xu Z, Pan C, Cousineau E, Burchfiel B, et al. 2024. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. arXiv preprint arXiv:2402.10329

- Chen T, He Z, Ciocarlie M. 2021. Hardware as Policy: Mechanical and Computational Co-Optimization using Deep Reinforcement Learning. In Conference on Robot Learning
- Peng XB, Andrychowicz M, Zaremba W, Abbeel P. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In 2018 IEEE international conference on robotics and automation (ICRA), pp. 3803–3810. IEEE
- 97. Tiboni G, Arndt K, Kyrki V. 2023. Dropo: Sim-to-real transfer with offline domain randomization. Robotics and Autonomous Systems: 104432
- 98. Muratore F, Ramos F, Turk G, Yu W, Gienger M, Peters J. 2022. Robot learning from randomized simulations: A review. Frontiers in Robotics and AI 9:799893
- Pinto L, Davidson J, Sukthankar R, Gupta A. 2017. Robust Adversarial Reinforcement Learning. In Proceedings of the 34th International Conference on Machine Learning, ed. D Precup, YW Teh, pp. 2817–2826, vol. 70 of Proceedings of Machine Learning Research, pp. 2817–2826
- Arndt K, Hazara M, Ghadirzadeh A, Kyrki V. 2020. Meta reinforcement learning for sim-toreal domain adaptation. In 2020 IEEE international conference on robotics and automation (ICRA), pp. 2725–2731. IEEE
- Ren AZ, Dai H, Burchfiel B, Majumdar A. 2023. AdaptSim: Task-Driven Simulation Adaptation for Sim-to-Real Transfer. In 7th Annual Conference on Robot Learning
- 102. Åström KJ. 1995. Adaptive control. In Mathematical System Theory: The Influence of RE Kalman. Springer
- Sastry S, Isidori A. 1989. Adaptive control of linearizable systems. IEEE Transactions on Automatic Control 34(11):1123–1131
- 104. Kumar A, Fu Z, Pathak D, Malik J. 2021. Rma: Rapid motor adaptation for legged robots. Robotics: Science and Systems
- 105. Bousmalis K, Irpan A, Wohlhart P, Bai Y, Kelcey M, et al. 2018. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In 2018 IEEE international conference on robotics and automation (ICRA), pp. 4243–4250. IEEE
- Zhang F, Leitner J, Ge Z, Milford M, Corke P. 2019. Adversarial discriminative sim-to-real transfer of visuo-motor policies. The International Journal of Robotics Research 38
- 107. Niu H, Qiu Y, Li M, Zhou G, Hu J, et al. 2022. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. Advances in Neural Information Processing Systems 35:36599–36612
- 108. Ankile L, Simeonov A, Shenfeld I, Torne M, Agrawal P. 2024. From imitation to refinement–residual rl for precise assembly. arXiv preprint arXiv:2407.16677
- 109. Maddukuri A, Jiang Z, Chen LY, Nasiriany S, Xie Y, et al. 2025. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. arXiv preprint arXiv:2503.24361
- 110. Nasiriany S, Maddukuri A, Zhang L, Parikh A, Lo A, et al. 2024. RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots. In Robotics: Science and Systems (RSS)
- 111. Bjorck J, Castañeda F, Cherniadev N, Da X, Ding R, et al. 2025. Gr00t n1: An open foundation model for generalist humanoid robots. arXiv preprint arXiv:2503.14734
- Liang J, Saxena S, Kroemer O. 2020. Learning Active Task-Oriented Exploration Policies for Bridging the Sim-to-Real Gap. In Robotics: Science and Systems
- 113. Wagenmaker A, Huang K, Ke L, Jamieson K, Gupta A. 2024. Overcoming the sim-to-real gap: Leveraging simulation to learn to explore for real-world rl. *Advances in Neural Information Processing Systems* 37:78715–78765
- 114. Clavera I, Held D, Abbeel P. 2017. Policy transfer via modularity and reward guiding. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE
- Mueller M, Dosovitskiy A, Ghanem B, Koltun V. 2018. Driving Policy Transfer via Modularity and Abstraction. In Conference on Robot Learning, pp. 1–15. PMLR
- Zhang F, Leitner J, Milford M, Corke P. 2017. Modular deep Q networks for sim-to-real transfer of visuo-motor policies. In Australasian Conference on Robotics and Automation 2017, pp. 62–71. Australian Robotics and Automation Association (ARAA)

- 117. Julian R, Heiden E, He Z, Zhang H, Schaal S, et al. 2020. Scaling simulation-to-real transfer by learning composable robot skills. In Proceedings of the 2018 International Symposium on Experimental Robotics, pp. 267–279. Springer
- 118. Chen D, Zhou B, Koltun V, Krähenbühl P. 2020. Learning by cheating. In Conference on robot learning, pp. 66–75. PMLR
- Hu ES, Springer J, Rybkin O, Jayaraman D. 2024. Privileged Sensing Scaffolds Reinforcement Learning. In The Twelfth International Conference on Learning Representations
- Messikommer N, Xing J, Aljalbout E, Scaramuzza D. 2025. Student-Informed Teacher Training. In The Thirteenth International Conference on Learning Representations
- 121. Pinto L, Andrychowicz M, Welinder P, Zaremba W, Abbeel P. 2018. Asymmetric actor critic for image-based robot learning. *Robotics: Science and Systems*
- 122. Nguyen HH, Baisero A, Wang D, Amato C, Platt R. 2022. Leveraging Fully Observable Policies for Learning under Partial Observability. In 6th Annual Conference on Robot Learning
- Krinner M, Aljalbout E, Romero A, Scaramuzza D. 2025. Accelerating model-based reinforcement learning with state-space world models. arXiv preprint arXiv:2502.20168
- 124. Tanwani A. 2021. DIRL: Domain-invariant representation learning for sim-to-real transfer. In Conference on Robot Learning, pp. 1558–1571. PMLR
- 125. Yoneda T, Yang G, Walter MR, Stadie BC. 2022. Invariance Through Latent Alignment. In Proceedings of Robotics: Science and Systems (RSS)
- Xing J, Bauersfeld L, Song Y, Xing C, Scaramuzza D. 2024. Contrastive learning for enhancing robust scene transfer in vision-based agile flight. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 5330-5337. IEEE
- 127. Xing J, Geles I, Song Y, Aljalbout E, Scaramuzza D. 2024. Multi-task reinforcement learning for quadrotors. *IEEE Robotics and Automation Letters*
- 128. Handa A, Allshire A, Makoviychuk V, Petrenko A, Singh R, et al. 2023. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 5977–5984. IEEE
- 129. Aractingi M, Léziart PA, Flayols T, Perez J, Silander T, Souères P. 2023. Controlling the solo12 quadruped robot with deep reinforcement learning. *scientific Reports* 13(1):11945
- Mysore S, Mabsout B, Mancuso R, Saenko K. 2021. Regularizing action policies for smooth control with reinforcement learning. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 1810–1816. IEEE
- 131. Chen Z, He X, Wang YJ, Liao Q, Ze Y, et al. 2024. Learning smooth humanoid locomotion through lipschitz-constrained policies. arxiv Preprint arXiv:2410.11825
- 132. Kadian A, Truong J, Gokaslan A, Clegg A, Wijmans E, et al. 2020. Sim2real predictivity: Does evaluation in simulation predict real-world performance? IEEE RA-L 5(4):6670-6677
- Lambertenghi SC, Stocco A. 2024. Assessing quality metrics for neural reality gap input mitigation in autonomous driving testing. In 2024 IEEE Conference on Software Testing, Verification and Validation (ICST), pp. 173–184. IEEE
- 134. Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X. 2016. Improved techniques for training gans. Advances in neural information processing systems 29
- 135. Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS 2017*
- 136. Yu J, Xu X, Gao F, Shi S, Wang M, et al. 2020. Toward realistic face photo–sketch synthesis via composition-aided gans. *IEEE transactions on cybernetics* 51(9):4350–4362
- Van der Maaten L, Hinton G. 2008. Visualizing data using t-sne. Journal of machine learning research 9(11)
- 138. Biruduganti S, Yardi Y, Ankile L. 2025. Bridging the sim2real gap: Vision encoder pre-training for visuomotor policy transfer. arXiv preprint arXiv:2501.16389
- Chen BH, Negrut D. 2024. Instance performance difference: A metric to measure the sim-toreal gap in camera simulation. arXiv preprint arXiv:2411.07375

- 140. James S, Wohlhart P, Kalakrishnan M, Kalashnikov D, Irpan A, et al. 2019. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition
- Lin T, Sachdev K, Fan L, Malik J, Zhu Y. 2025. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids. arXiv preprint arXiv:2502.20396
- 142. Dalal M, Liu M, Talbott W, Chen C, Pathak D, et al. 2024. Local policies enable zero-shot long-horizon manipulation. arXiv preprint arXiv:2410.22332
- 143. Singh R, Allshire A, Handa A, Ratliff N, Van Wyk K. 2024. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. arXiv preprint arXiv:2412.01791
- 144. Zeng KH, Zhang Z, Ehsani K, Hendrix R, Salvador J, et al. 2024. Poliformer: Scaling on-policy rl with transformers results in masterful navigators. 8th Annual Conference on Robot Learning
- Gervet T, Chintala S, Batra D, Malik J, Chaplot DS. 2023. Navigating to objects in the real world. Science Robotics 8(79):eadf6991
- 146. Chang M, Gervet T, Khanna M, Yenamandra S, Shah D, et al. 2024. GOAT: GO to Any Thing. In Proceedings of Robotics: Science and Systems (RSS)
- Xing J, Romero A, Bauersfeld L, Scaramuzza D. 2024. Bootstrapping reinforcement learning with imitation for vision-based agile flight. arXiv preprint arXiv:2403.12203
- 148. Sandha SS, Garcia L, Balaji B, Anwar F, Srivastava M. 2021. Sim2real transfer for deep reinforcement learning with stochastic state transition delays. In Conference on robot learning
- 149. Lambert N, Amos B, Yadan O, Calandra R. 2018. Objective Mismatch in Model-based Reinforcement Learning. In Learning for Dynamics and Control
- Guzman R, Oliveira R, Ramos F. 2022. Bayesian Optimisation for Robust Model Predictive Control under Model Parameter Uncertainty. In International Conference on Robotics and Automation (ICRA), pp. 5539–5545
- 151. Newbury R, Collins J, He K, Pan J, Posner I, et al. 2024. A review of differentiable simulators. $IEEE\ Access$
- 152. NVIDIA Corporation. 2024. Nvidia warp: Differentiable spatial computing for python. https://developer.nvidia.com/warp-python. Version 1.5
- 153. Hu Y, Anderson L, Li TM, Sun Q, Carr N, et al. 2020. Difftaichi: Differentiable programming for physical simulation. *International Conference on Learning Representations (ICLR)*
- 154. Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, et al. 2018. JAX: Composable transformations of Python+NumPy programs. http://github.com/google/jax
- 155. Ansel J, You K, Team P. 2023. Torchdynamo and torchinductor: Pytorch compilation stack. https://pytorch.org/docs/stable/torch.compiler.html. Accessed 2025-06-27
- 156. Ming R, Huang Z, Ju Z, Hu J, Peng L, Zhou S. 2024. A survey on future frame synthesis: Bridging deterministic and generative approaches. arXiv preprint arXiv:2401.14718
- 157. Bruce J, Dennis MD, Edwards A, Parker-Holder J, Shi Y, et al. 2024. Genie: Generative Interactive Environments. In Forty-first International Conference on Machine Learning
- 158. Hassan M, Stapf S, Rahimi A, Rezende P, Haghighi Y, et al. 2025. Gem: A generalizable ego-vision multimodal world model for fine-grained ego-motion, object dynamics, and scene composition control. In Conference on Computer Vision and Pattern Recognition (CVPR)
- 159. Agarwal N, Ali A, Bala M, Balaji Y, Barker E, et al. 2025. Cosmos world foundation model platform for physical ai. arXiv preprint arXiv:2501.03575
- 160. Ding J, Zhang Y, Shang Y, Zhang Y, Zong Z, et al. 2024. Understanding world or predicting future? a comprehensive survey of world models. arXiv preprint arXiv:2411.14499
- Cranmer K, Brehmer J, Louppe G. 2020. The frontier of simulation-based inference. Proc. Natl. Acad. Sci. 117(48):30055–30062
- 162. Sisson SA, Fan Y, Beaumont M, ed. 2018. *Handbook of Approximate Bayesian Computation*. New York: Chapman and Hall/CRC
- Glöckler M, Deistler M, Macke JH. 2022. Variational methods for simulation-based inference. arXiv preprint arXiv:2203.04176

- 164. Papamakarios G, Murray I. 2016. Fast ε-free Inference of Simulation Models with Bayesian Conditional Density Estimation. In Advances in Neural Information Processing Systems
- Radev ST, Mertens UK, Voss A, Ardizzone L, Köthe U. 2022. Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks* and Learning Systems 34(7):3694–3708
- 166. Ramos F, Possas R, Fox D. 2019. BayesSim: Adaptive Domain Randomization Via Probabilistic Inference for Robotics Simulators. In Proceedings of Robotics: Science and Systems. Freiburg im Breisgau, Germany
- 167. Barcelos L, Oliveira R, Possas R, Ott L, Ramos F. 2020. DISCO: Double Likelihood-free Inference Stochastic Control. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). IEEE
- 168. Matl C, Narang Y, Bajcsy R, Ramos F, Fox D. 2020. Inferring the Material Properties of Granular Media for Robotic Tasks. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2770–2777. IEEE
- 169. Possas R, Barcelos L, Oliveira R, Fox D, Ramos F. 2020. Online BayesSim for Combined Simulator Parameter Inference and Policy Improvement. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8005–8012. IEEE
- 170. Antonova R, Yang J, Sundaresan P, Fox D, Ramos F, Bohg J. 2021. A bayesian treatment of real-to-sim for deformable object manipulation. arXiv preprint arXiv:2112.05068
- 171. Muratore F, Gruner T, Wiese F, Belousov B, Gienger M, Peters J. 2022. Neural Posterior Domain Randomization. In Proceedings of the 5th Conference on Robot Learning
- 172. Matl C, Narang Y, Fox D, Bajcsy R, Ramos F. 2021. STReSSD: Sim-To-Real from Sound for Stochastic Dynamics. In Proceedings of the Conference on Robot Learning (CoRL), vol. 155
- 173. O'Neill A, Rehman A, Maddukuri A, Gupta A, Padalkar A, et al. 2024. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 6892–6903. IEEE
- 174. Khazatsky A, Pertsch K, Nair S, Balakrishna A, Dasari S, et al. 2024. Droid: A large-scale in-the-wild robot manipulation dataset. arXiv preprint arXiv:2403.12945
- 175. Liu G, Deng Y, Zhao R, Zhou H, Chen J, et al. 2025. DexScale: Automating Data Scaling for Sim2Real Generalizable Robot Control. In International Conference on Machine Learning
- 176. Mandlekar A, Nasiriany S, Wen B, Akinola I, Narang Y, et al. 2023. MimicGen: A Data Generation System for Scalable Robot Learning using Human Demonstrations. In Conference on Robot Learning, pp. 1820–1864. PMLR
- 177. Li X, Hsu K, Gu J, Pertsch K, Mees O, et al. 2024. Evaluating real-world robot manipulation policies in simulation. 8th Anal Conference on Robot Learning
- 178. Barreiros J, Beaulieu A, Bhat A, Cory R, Cousineau E, et al. 2025. A careful examination of large behavior models for multitask dexterous manipulation. arXiv preprint arXiv:2507.05331