Research Thesis

Sim-to-Real Transfer for Locomotion Tasks on Legged Robots: A Survey

Luis Scheuch

14 July 2025

Supervisor Mike Wesselhöft, M. Sc.
Examiner Prof. Dr.-Ing. Carlos Jahn

This thesis is typeset using typst [1], a modern alternative to LaTeX [2]. Several large language model tools were used to assist in the writing process, including: OpenAI's ChatGPT [3], Google's Gemini [4], Anthropic's Claude [5], DeepSeek's DeepSeek [6], and DeepL's Write [7]. These tools were employed not for direct text generation, but as aids for ideation, summarization, and grammatical refinement.

Hereby I declare that I produced the present work myself only with the help of the indicated aids and sources.

Hamburg, 14 July 2025 Luis Scheuch

Abstract

The sim-to-real gap between simulated and real-world environments remains a fundamental challenge in deploying Reinforcement Learning policies for robotic locomotion. This thesis presents a comprehensive survey of sim-to-real transfer techniques for legged robots, with additional emphasis on bipedal systems and the NAO platform used in the RoboCup.

Through systematic analysis of over 250 papers, this thesis identifies and evaluates existing techniques for reducing the sim-to-real gap and assesses their applicability to humanoid robotics. My findings reveal that domain randomization emerges as the dominant approach, appearing in over 80 surveyed implementations. Successful sim-to-real transfer typically requires combining multiple techniques rather than relying on a single method. Hardware constraints, particularly relevant for resource-limited platforms like the NAO, heavily influence technique selection. This survey contributes a comprehensive seven-component pipeline for sim-to-real transfer, integrating system identification, observation space design with I/O history, comprehensive domain randomization, curriculum learning, simulation grounding, online system identification, and predictive control.

This work provides researchers and practitioners with a structured roadmap for implementing robust sim-to-real transfer for robotic locomotion tasks, advancing toward the RoboCup 2050 goal of creating autonomous humanoid robots capable of competing with human soccer champions.

Contents

List of Figures

Acronyms Index

A3C	Asynchronous Advantage Actor-Critic
ACGD	Adaptive Curriculum Generation from Demonstrations
AI	Artificial Intelligence
ANN	Artificial Neural Network
AcDR	Active Domain Randomization
AuDR	Automatic Domain Randomization
BO	Bayesian Optimization
BayRn	Bayesian Domain Randomization
CDR	Continual Domain Randomization
CMDP	Constrained Markov Decision Process
CPG	Central Pattern Generator
CPU	Central Processing Unit
DAgger	Dataset Aggregation
DDPG	Deep Deterministic Policy Gradient
DORAEMON	DOmain RAndomization via Entropy MaximizatiON
DR	Domain Randomization
DRL	Deep Reinforcement Learning
DROID	Domain Randomization Optimization IDentification
DROPO	Domain Randomization Off-Policy Optimization
EAP	Error-Aware Policy
GAN	Generative Adversarial Network
GAT	Grounded Action Transformation
GPU	Graphics Processing Unit
GSL	Grounded Simulation Learning
I/O	Input/Output
IL	Imitation Learning
IMU	Inertial Measurement Unit
KNN	K-Nearest-Neighbor
MDP	Markov Decision Process
ML	Machine Learning
MPC	Model Predictive Control
NN	Neural Network
NPDR	Neural Posterior Domain Randomization
OG	Optimality Gap

PINN	Physics-Informed Neural Network
PNN	Progressive Neural Network
POMDP	Partially Observable Markov Decision Process
RGAILfO	Robust Generative Adversarial Imitation Learning from Observation
RL	Reinforcement Learning
RMA	Rapid Motor Adaptation
SAC	Soft Actor-Critic
SGAT	Stochastic Grounded Action Transformation
SIL	software-in-the-loop
SNN	Spiking Neural Network
SOB	Simulation Optimization Bias
SOTA	state-of-the-art
SPL	Standard Platform League
SPOTA	Simulation-based Policy Optimization with Transferability Assessment
SimTwin	Simulation Twin
TL	Transfer Learning
TRC	Trust Region Conditional Value at Risk
UCBOG	Upper Confidence Bound on the Optimality Gap
UKF	Unscented Kalman Filter
VAE	Variational Auto Encoder
VPE	Variational Policy Embedding

1. Introduction

Artificial intelligence and robotics are two of the most promising fields to shape the future of manufacturing and automation technologies [8], [9]. As robotic systems become increasingly deployed in diverse environments - from industrial assembly lines to service robotics and autonomous vehicles - the need for robust and adaptive motion control algorithms has become paramount.

Traditional approaches to robotic motion control often rely on classical algorithms or hand-tuned parameters, such as keyframe animations for specific motions or carefully calibrated control loops. Tuning these algorithms is an error-prone and time-consuming task. There are many high-dimensional parameters to tune, there is always the risk of overfitting to specific hardware configurations, and significant domain knowledge and experience are required to achieve good results. These challenges are particularly pronounced for complex systems like legged robots, where the dynamics involve intricate balance control, ground contact forces, and coordination of multiple actuators.

Methods from Reinforcement Learning (RL) offer a promising solution to these challenges. Various works demonstrate the potential of RL for robotic motion control across different platforms and applications [10], [11], [12]. For example, Haarnoja et al. [10] train custom soccer robots using Deep Reinforcement Learning (DRL) to autonomously play soccer, i.e., kicking a moving ball, blocking a shot, turning, getting up after falling, strategic defense and recovery from pushes. Compared to a scripted baseline, they achieved a walk-speed increase of 181 %, a turn speed increase of 302 %, and a standup time reduction of 63 %.

While these results demonstrate the potential of sim-to-real transfer, significant challenges remain in systematically addressing the sim-to-real gap across diverse robotic platforms, in particular for multi-legged and bipedal robots.

This thesis provides a comprehensive survey of available techniques to reduce the sim-to-real gap, also known as reality gap, with a focus on motion-related robotics tasks. Through systematic analysis of over 250 relevant publications, this work identifies Domain Randomization (DR) as the most prevalent approach, appearing in more than 80 practical implementations across diverse robotic platforms. The findings reveal that successful sim-to-real transfer in challenging environments typically requires combining multiple complementary techniques rather than relying on a single method, with the selection heavily influenced by specific hardware constraints - particularly relevant for resource-limited platforms like the NAO humanoid bipedal robot with its 83 Hz control frequency and significant communication latencies. The research demonstrates that bidirectional learning approaches consistently outperform unidirectional methods for challenging locomotion tasks, despite their implementation complexity. This thesis provides a structured overview of existing techniques and gives recommendations for their application. Additionally, it presents an exemplary seven-component sim-to-real pipeline specifically designed for the NAO robot, integrating system identification, observation space design with Input/Output (I/O) history, comprehensive DR, curriculum learning, simulation grounding, online system identification,

and predictive control to address the unique challenges of humanoid robotics in constrained environments.

1.1. Research Questions

This thesis addresses the following research questions:

RQ1. What are existing techniques to reduce the sim-to-real gap in robotics for motion-related tasks?

RQ2. Which of these techniques are applicable to humanoid robotics and appropriate for the NAO robot as a special case?

1.2. Structure

This thesis is structured as follows: Chapter 2 summarizes the required background knowledge and essential concepts. Chapter 3 explains the research methodology and writing process of this thesis. Chapter 4 describes possible techniques to set up a robotics RL task. This chapter provides a solid baseline for RL-based motion control, on which additional sim-to-real techniques can be added. Chapter 5 will then introduce the concept of sim-to-real transfer, highlight its significance in the context of robotics and discuss available techniques tailored to reduce the sim-to-real gap. Chapter 6 synthesizes the key findings of the literature survey, examines their implications for robotic locomotion, and identifies critical gaps that remain to be addressed. Lastly, Chapter 7 concludes the thesis, answers the research questions, and provides an outlook on possible future work.

2. Preliminaries

This chapter provides an overview of important topics and concepts, which will be relevant for the following chapters. In the first section, mathematical frameworks, including Markov Decision Processes (MDPs) and their variants for modeling uncertainty, are introduced, followed by a detailed overview of RL concepts, algorithms, and their formalism. The chapter concludes with a discussion of common RL learning strategies, such as knowledge distillation and curriculum learning.

2.1. Markov Decision Processes

An MDP is a mathematical framework for modeling sequential decision making for a stochastic, discrete-time process. In simplified form, it is defined as a tuple $M = \{S, A, f, r\}$, where S is a set of states the robot can be in, A is the set of actions, $f : S \times A \to S'$ is a transition function describing the probability of reaching state s' from state s by taking action a, and $r : S \times A \times S' \to \mathbb{R}$ is a reward function describing the expected reward for taking action a in state s and reaching state s' [13].

2.1.1. Markov Property

The Markov property states that the future state of a process only depends on the current state and action, not on any past states or actions. This means that the process is *memoryless*, making it a *Markovian* process [13].

2.1.2. Partially Observable Markov Decision Process

A Partially Observable Markov Decision Process (POMDP) is a generalization of an MDP that accommodates partial observability of the state, thereby enabling "principled decision making under conditions of uncertain sensing" [14]. It was specifically introduced for RL in uncertain environments. POMDPs extend the MDP by adding an observation function $O(o \mid s, a, s')$, which maps the current state s, the chosen action a and the next state s' to a distribution over observations o. In some algorithms, the observation function is simplified to $O(o \mid s, s')$ [14].

2.1.3. Constrained Markov Decision Process

A Constrained Markov Decision Process (CMDP) is a generalization of an MDP that is, for example, used in safe RL. It extends the MDP by adding a set of constraints $\mathcal{C} = \{(C_i, b_i)\}_{i=1}^{m}$, where C_i is a cost function, b_i is the safety constraint bound and m is the total number of constraints [15]. The cost function C_i maps transition tuples $\left(s^{(t)}, a^{(t)}, s^{(t+1)}\right)$ to an associated cost $c_i \in \mathbb{R}$ and the safety constraint bound $b_i \in \mathbb{R}$ is a threshold that the cost function must not exceed [16].

2.2. Reinforcement Learning

RL is a subfield of machine learning. Inspired by human and animal learning, an agent learns by interacting with an environment. There are three main differences from supervised and unsupervised learning, the two other subfields in machine learning [17]:

1. RL is closed-loop: The agent's actions influence the environment's state, which in turn affects future observations and rewards, creating a feedback loop.
2. No direct supervision: Unlike supervised learning, RL does not need labeled input-output pairs to be presented, and does not need sub-optimal actions to be explicitly corrected.
3. Long-term consequences: Actions have long-term effects on future rewards, requiring the agent to reason about delayed consequences.

2.2.1. Terms and Concepts

The *agent* is learner and decision-maker that interacts with the *environment* E by choosing an *action* a. The agent's algorithm that selects actions is the *policy*, usually denoted by μ in the deterministic case or by π in the stochastic case. A trivial example of a stochastic policy is a random policy, which, for example, chooses actions uniformly at random from the action space A.

The environment is described by a *state* s, which is typically not directly accessible to the agent. The agent observes the environment through an *observation* o, which is usually a partial description of the environment state.

Multiple consecutive actions, states, and immediate rewards are grouped into a *trajectory*, also known as a *rollout* τ. For example, a trajectory can be defined as $\tau = \big(\big(s^{(0)}, a^{(0)}, r^{(0)}\big), \big(s^{(1)}, a^{(1)}, r^{(1)}\big), ..., \big(s^{(T)}, a^{(T)}, r^{(T)}\big)\big)$, where $s^{(t)}$ is the state at time t, $a^{(t)}$ is the action taken at time t, and $r^{(t)}$ is the reward received at time t.

As feedback, the agent receives a *reward* $r^{(t)} = R\big(s^{(t)}, a^{(t)}, s^{(t+1)}\big)$, depending on the current state $s^{(t)}$, chosen action $a^{(t)}$, and next state $s^{(t+1)}$. In some algorithms, the reward function R is also simplified to $r^{(t)} = R\big(s^{(t)}, a^{(t)}\big)$ or simply $r^{(t)} = R\big(a^{(t)}\big)$.

The *return* is the reward for multiple states and actions, i.e., the reward of a trajectory τ. It can, e.g., be *finite-horizon undiscounted* ($R(\tau) = \sum_{t=0}^{T} r^{(t)}$) or *infinite-horizon discounted* ($R(\tau) = \sum_{t=0}^{\infty} \gamma^t r^{(t)}$, $\gamma \in (0, 1)$). The return of a specific policy π is denoted by R_π.

The goal of RL is to find the optimal π^*, which always chooses the optimal action, i.e., an action that maximizes the expected return:

$$\begin{aligned} \pi^* &= \arg\max_{\pi} \sum_{t=0}^{\infty} \gamma^t r\big(s^{(t)}, \pi\big(o^{(t)}\big)\big) \\ &= \arg\max_{\pi} R_\pi\big(s^{(0)}\big). \end{aligned} \tag{1}$$

R_π is the *infinite-horizon discounted reward* starting in initial state $s^{(0)}$ and following policy π [17], [18].

2.2.2. Formalism

MDPs, POMDPs, and CMDPs are frequently used to model robotic RL problems. The goal of this formulation is to find a mapping from state s to action a that maximizes the return $R(\tau)$. Now, the *total reward goal* can be written as

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{a \sim \pi(\cdot \mid s)}[R(\tau)] \tag{2}$$

where the solution is the optimal policy π^* [19].

In order to better describe and distinguish different RL algorithms and sim-to-real approaches, this thesis adopts a notation similar to Salvato et al. [20] by using a control-systems-oriented formalism.

A *dynamical discrete-time system* Ω is a tuple (S, A, O, f, g), where S is the state space, A is the action space, O is the observation space, f the state transition function $S \times A \to S$, and g the observation function $S \to O$. Two state laws describe how the system behaves when it starts in an initial state $s^{(0)}$ and is subject to an action sequence $a^{(0)}, a^{(1)}, ...$:

$$s^{(t+1)} = f\big(s^{(t)}, a^{(t)}\big) \tag{3}$$

$$o^{(t+1)} = g\big(s^{(t+1)}\big) = g\big(f\big(s^{(t)}, a^{(t)}\big)\big) \tag{4}$$

An *environment* E is a dynamical discrete-time system Ω, but with an additional reward function $r : S \times A \to \mathbb{R}$. The environment evolves according to the state laws of Ω.

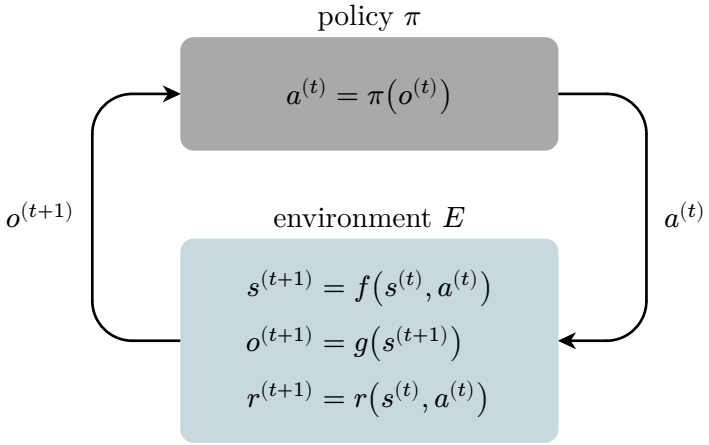


Figure 1: Relation between policy and environment.

Distinguishing between Simulation and Real World

Previously defined terms can be used to formally distinguish between simulation and real-world environments.

Figure 2: Distinguishing between simulation and real-world environments.

Symbols with a prime ($'$) are used to denote the simulation environment, while symbols without a prime are used to denote the real-world environment; i.e., E refers to the real-world environment and E' refers to the simulation environment.

2.2.3. Model-free vs. Model-based

There are two main classes of RL algorithms: *model-free* and *model-based*. In model-free algorithms, the agent has no access to a model of the environment E; rather, it learns a direct policy mapping between states s and actions a. In model-based algorithms, the agent has access to f and g, and thus knows how the state s will evolve and what the agent will observe o [19], [20]. Model-free methods require more data and experience to learn a task, but usually produce better results than model-based methods [19]. There are also hybrid approaches, which try to combine the advantages of model-free and model-based methods [19].

2.2.4. Robust Reinforcement Learning

Robust RL is a term introduced by Mankowitz et al. [21] to explicitly take environmental uncertainties into account in the RL formulation. It is based on the MDP formulation, with the difference that the transition function $f : S \times \mathcal{M}(S)$ maps to the *uncertainty set* $\mathcal{M}(S)$, which is a set of probability measures over the next states $s' \in S$. Using this, the RL goal can be reformulated as finding a policy π^* that optimizes for the worst-case expected return [21], similar to an adversarial optimization problem [22].

2.2.5. Training in Simulation vs. Training on Real Hardware

For online policy interaction by an agent, a simulator with a model of the system is usually the only viable option. Simulation can help to bypass many challenges of learning on real systems, such as data efficiency and safety [23]. Simulated data is provided at low cost, but involves a mismatch with the real-world setting [22]. This poses a problem, since for example even small errors can quickly accumulate and cause the model to deviate from real behavior in dynamic tasks [12].

2.3. Learning Strategies

This section introduces common methods for training RL policies or general machine learning models with a focus on robotic tasks.

2.3.1. Knowledge Distillation

Knowledge distillation or *teacher-student* learning is a method for transferring knowledge from a teacher network to a student network. It was first introduced by Hinton et al. [24] in 2015 and has since been used in many different applications. Knowledge distillation is based on the idea that one network can learn from another network or a collection of multiple networks, while keeping the same performance and possibly decreasing size and computational cost [25].

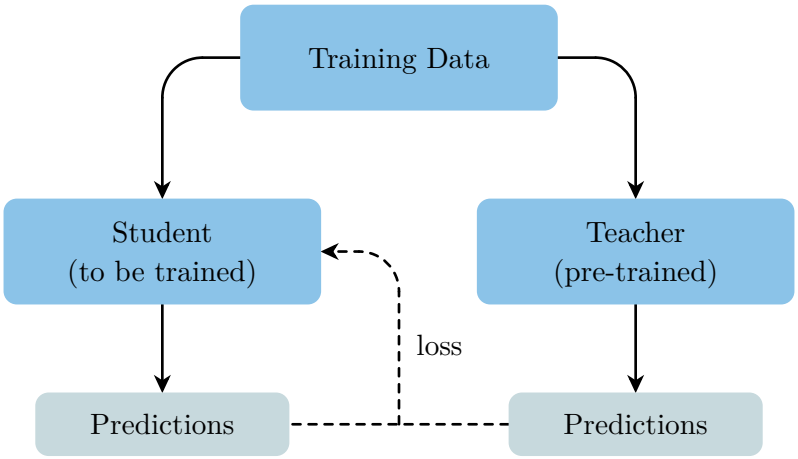


Figure 3: Teacher-student learning framework.

Teacher-student learning can also be used to enable the learning of complex tasks by first training a teacher model with access to privileged information, such as ground truth data, and then transferring the knowledge to a student model that does not have access to this information [25]. Chen et al. [26] introduced this as *Learning by Cheating*.

2.3.2. Curriculum Learning

Curriculum learning is a method in machine learning to improve generalization, speed of convergence and quality of the learned minimum by presenting training data in a meaningful order and thus improving sampling efficiency [25], [27]. In the context of bipedal walking, this could mean starting with simple tasks, such as standing, then gradually increasing the walk speed and finally learning to walk on uneven terrain.

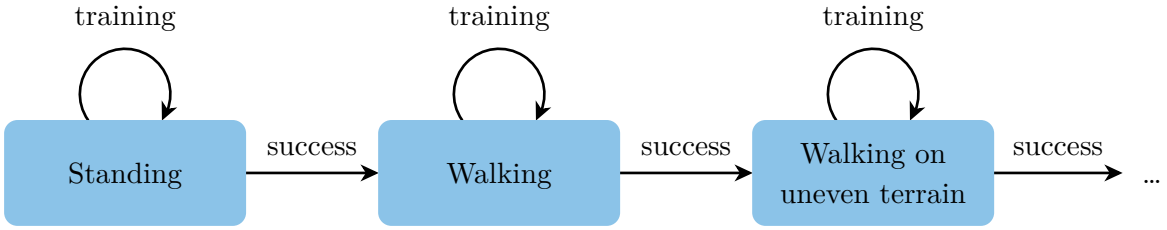


Figure 4: Curriculum learning example.

2.3.3. Imitation Learning

Imitation Learning (IL) is a machine learning paradigm in which an agent acquires skills or behaviors by observing demonstrations from a teacher, typically a human expert or another proficient agent. Rather than relying on explicit programming or carefully crafted reward functions, IL enables agents to learn complex tasks by mapping observed states to actions, making it valuable for scenarios where specifying the desired behavior is difficult [28].

2.3.4. Meta-Learning

Meta-learning is the idea of giving systems the ability to learn how to learn [29]. In contrast to most common machine learning problem formulations, the goal is to train models that are good and fast at adapting to new tasks (fine-tuning), instead of training models that are optimized for solving one specific task out of the box [30], [31].

3. Methodology

To achieve a comprehensive overview of the current state of sim-to-real in robotics, this work is a structured literature review [32] with a preliminary snowballing [33] approach.

The goal of this literature review is to identify state-of-the-art (SOTA) techniques to reduce the sim-to-real gap in robotics, with a focus on motion-related tasks, i.e., mostly omitting vision-related tasks. It updates and extends existing surveys, such as [20], [22], but focuses only on techniques to reduce the sim-to-real gap, without going into much detail about the general setting of RL for robotics, such as [12], [34], [35]. Chapter 4 only provides setup steps relevant for sim-to-real transfer, such as system identification, or the design of the observation and action space. Chapter 5 then summarizes the results of the literature review and provides an overview of relevant techniques.

Following the article "Producing a systematic review" by Denyer and Tranfield [32], the literature review process is split into five steps, which are detailed in the following sections:

1. Question formulation
2. Locating studies
3. Study selection and evaluation
4. Analysis and synthesis
5. Reporting and using the results

3.1. Question Formulation

The first step in a systematic review is to define the research questions, which should be as specific and clear as possible.

> A good systematic review is based on a well-formulated, answerable question. The question guides the review by defining which studies will be included, what the search strategy to identify the relevant primary studies should be, and which data need to be extracted from each study. Ask a poor question and you will get a poor review.
>
> — C. E. Counsell [36]

The research questions were developed through an iterative refinement process involving consultation with domain experts, including academic supervisors and practitioners from the RoboCup community, particularly members of Team HULKs [37].

3.2. Locating Studies

In step two, the aim is to find as much literature relevant to the research questions as possible. All primary searches were executed in the Scopus database by Elsevier [38] since it offers a wide range of scientific papers, including conference papers and journal articles from different publishers, as well as good search tools. Other databases such as IEEE Xplore [39] and Springer Link [40] were not directly used in this step, since this would have exceeded

the scope of this work and the search in Scopus already provided 197 papers from different publishers and databases, such as IEEE Xplore [39], Springer Link [40], MDPI [41], Sage Journals [42], and others.

The search term used for the literature review is: ("sim-to-real" OR "sim to real" OR "reality gap") AND "simulation" AND "robotics" AND "reinforcement learning".

Some additional filters were used to narrow down the search results:
- The search strings were used in the title, abstract, and keywords of the papers.
- The subject area was limited to "Computer Science", "Engineering", and "Mathematics", while other topics were excluded.
- To include only topical papers, the publication date is limited to the last ten years, i.e., 2015-2025.
- Only papers in English are considered.

3.3. Study Selection and Evaluation

In the first step, a short literature review was conducted in a snowballing approach [33] to get a preliminary overview of the available literature to help formulate the research questions and the search string. This included roughly ten papers, including "Robot Learning from Randomized Simulations: A Review" [25] and "How to Train Your Robot with Deep Reinforcement Learning – Lessons We've Learned" [23].

In the second step, the search string was then used to find relevant papers in the Scopus database. This resulted in a collection of 197 papers, of which roughly 130 were found to be relevant to the research questions.

To check each paper's relevance to the research questions, the title, abstract, and conclusion were read. If the title, abstract, or conclusion contained relevant information, the remaining parts of the paper were skimmed, and relevant sections were read in detail. If other important works were cited in the paper, they were also checked for relevance to ensure that relevant papers not found by the search string, as well as foundational works often cited in the literature, are included in the survey.

Notes about important aspects of the paper were taken during the reading process and documented in an additional list.

3.4. Analysis and Synthesis

Relevant content was extracted, summarized, and structured into different categories to give a clear overview of the results. During the synthesis phase, papers were prioritized based on three key indicators of scientific impact and validity: citation count as a proxy for influence within the research community, the existence of follow-up work indicating sustained research interest, and practical demonstrations or real-world implementations that validated the theoretical approaches. Additionally, peer-reviewed papers were preferred over preprints or non-peer-reviewed articles.

To better differentiate different sim-to-real approaches, a mathematical formulation inspired by Salvato et al. [20] is used. Techniques are additionally split into two main categories: individual techniques in Section 5.3 and complete frameworks in Section 5.4.

3.5. Reporting and Using the Results

Chapter 5 summarizes the results of the literature review and provides an overview of relevant techniques. The techniques are ordered within Section 5.3 and Section 5.4 by popularity and relevance. To document the usage of DR as the most popular technique, a table was created that lists all relevant papers using DR as further reference.

Chapter 6 provides critical analysis and synthesis of the findings, examining the implications of technique selection for different hardware constraints and identifying patterns in successful sim-to-real implementations. This chapter moves beyond mere description to interpret why certain techniques dominate the field, what limitations persist despite their popularity, and how the combination of multiple techniques often proves necessary for robust transfer. The discussion particularly emphasizes the unique challenges faced by resource-constrained platforms like the NAO robot, bridging the gap between general sim-to-real principles and specific implementation requirements.

4. Reinforcement Learning Setup

This chapter examines key techniques for setting up robotics RL tasks, specifically focusing on system identification, observation/action space design, control architectures, and safety regularization methods that enhance real-world transferability. The goal of this chapter is to provide a solid baseline for further sim-to-real techniques, which will be discussed in Chapter 5 to reduce the remaining sim-to-real gap.

4.1. System Identification

System identification or model learning is the process of determining physical parameters of a given system that best fit observed data [43] and is typically the first step in sim-to-real transfer, if no model of the real system is available. System identification is typically done by minimizing the mean-squared error between observed data and model predictions given a known control signal [25]. Different techniques exist, with the most recent ones being deep learning-enhanced approaches such as Newton-Euler-based forward dynamics models [44], [45], and residual learning with Artificial Neural Networks (ANNs) that augment simulators to improve one-step prediction accuracy [46]. Additionally, classification-based system identification using simulated versus real sample discrimination [47], [48], and episodic RL formulations that treat system mismatch as a cost function [49] have been proposed. Lee et al. [43] provide a comprehensive review of recent system identification methods.

Xie et al. [50] emphasize that an accurate replication of the robot's mass and inertia is critical to their sim-to-real success in their experiments, and they demonstrate this by training and deploying a neural-network controller on the full-scale bipedal Cassie robot. They also observe that a residual target with an underlying reference motion works better than directly learning the target motion. Using knowledge distillation to train a direct policy works in simulation, but fails to transfer to the robot.

Haarnoja et al. [10] show that it does not always make sense to directly learn the physical, controlled parameters of a robot. In their study, they perform system identification for a custom bipedal robot with direct current control actuators. They find that using a simplified position controlled actuator model with torque feedback results in successful sim-to-real transfer, while their experiments with direct current control (which would more accurately model the actual servo operation) fail due to a larger sim-to-real gap.

4.1.1. Model Complexity

Different publications show that the complexity of the model used for training has a significant impact on the transferability and trainability of the policy. Generally, more realistic models and simulation lead to better sim-to-real transfer [22], but also lead to higher computational costs, i.e., slower training, and might lead the policy to overfit to simulation [20].

Diprasetya et al. [51] propose to use curriculum learning to gradually increase the complexity of the model used for training. Their approach starts training with a simple model and

gradually increases its complexity as the policy learns, to speed up training in the beginning and to allow for good sim-to-real transfer when training is finished.

While traditional approaches focus on physics-based parameter optimization, recent work explores using RL itself for system identification.

4.1.2. RL-Based System Identification

Jiang et al. [47] introduce an RL-based system identification method that offers an interesting perspective on the problem. They combine a traditional physics-based simulator with a state-action-dependent adaptation function. This adaptation function is trained using RL to minimize the difference between the state-action pairs predicted by the simulator and the observed state-action pairs, which are discriminated by a Generative Adversarial Network (GAN) [52]. To collect data, a suboptimal baseline policy is used beforehand. One challenge of this approach is that enough diverse data needs to be collected to train the adaptation function. In real-world tests, the proposed method outperforms five out of six domain adaptation experiments.

4.2. Design of Observation and Action Space

Tan et al. [53] observe that the design of the observation space plays an important role in the transferability of a policy. A high-dimensional observation space can lead to overfitting to simulated data, which makes transfer to the real robot difficult. In their work, they empirically show that their 4-dimensional observation space, consisting only of the robot's Inertial Measurement Unit (IMU), performs better in reality than a 12-dimensional observation space that includes the robot's motor angles. A smaller observation space leaves less room for overfitting [53] and can facilitate exploration [30], [54] as well as transferability to other domains [55], but might also make learning more difficult [53].

Aljalbout et al. [56] underline that the action parameterization is at least as critical for sim-to-real transfer as the observation design. Their large-scale comparison of 13 control spaces shows that velocity-based commands preserve a high success rate once deployed on the physical robot, while position commands transfer poorly. In addition, delta formulations outperform their "absolute" counterparts: defining the command as a change w.r.t. the current feedback yields higher accuracy, requires less hyper-parameter tuning, and exhibits lower tracking error.

4.3. Control Architecture

Different control architectures can have a significant impact on the success of sim-to-real transfer. In general, it can be said that open-loop control architectures are less suitable for sim-to-real transfer than closed-loop architectures [10], [57], [58]. This is because open-loop architectures are more sensitive to model inaccuracies and disturbances in the environment, since there is no feedback mechanism to correct for these errors.

Model Predictive Control (MPC) is a very popular (non-RL) control architecture for bipedal locomotion. For example, Boston Dynamics' Atlas robot uses an MPC algorithm at its core [59]. A disadvantage of MPC is that it is quite limited in terms of generalization capabilities and adaptability to dynamic environments [10], [60].

Bao et al. [60] distinguish two main categories of control architectures for DRL-based bipedal locomotion: end-to-end and hierarchical control.

4.3.1. End-to-End Control

End-to-end control architectures, where robot states are directly mapped to joint-level control actions, are one possible approach to bipedal locomotion. Haarnoja et al. [10], for example, successfully train soccer robots in an end-to-end fashion using RL and achieve remarkable performance in walk speed, turning, kicking, and getting up. In a similar fashion, Siekmann et al. [61] use RL to obtain a single policy capable of several gaits on a bipedal Cassie robot. There are many more examples of successful applications: [62], [63], [64], [65].

4.3.2. Hierarchical Control

Hierarchical control architectures decompose the control problem into multiple levels. Typically, a high-level policy might, for example, control navigation and path planning, while a low-level controller handles the fundamental locomotion and actuator control tasks [60].

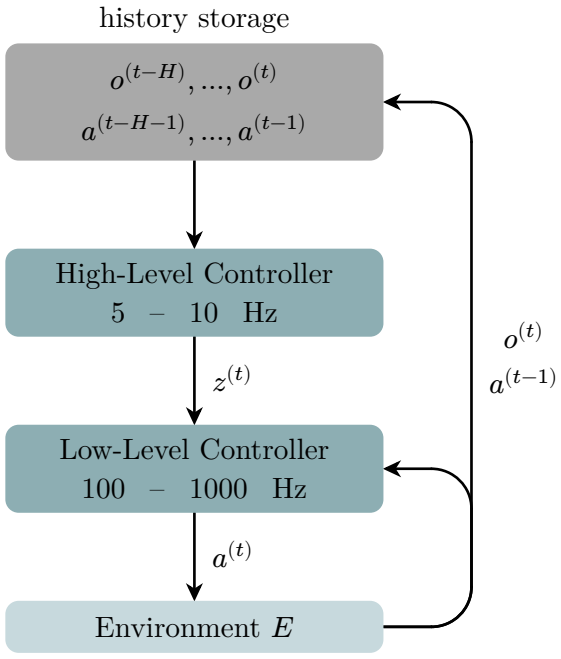


Figure 5: Example of an hierarchical control architecture Rapid Motor Adaptation (RMA) [66].

One advantage of hierarchical control architectures is that they can be run asynchronously, i.e., at different control frequencies or even on different hardware. Tan et al. [67] employ the high-level policy on a dedicated host, while the low-level controller runs on the robot's onboard computer.

In existing works, the high-level policy typically runs at a 4–70 times lower frequency than the low-level controller [57], [62], [64], [65], [66].

Central-Pattern-Based-RL

A specific type of hierarchical control that has been very popular in recent years and warrants mention here is the use of Central Pattern Generators (CPGs). CPGs are biologically inspired control architectures that map low-dimensional inputs into high-dimensional rhythmic motor outputs [68]. CPGs have been applied to various bipedal, quadrupedal, and other robotic locomotion tasks, first demonstrated for bipedal locomotion by Taga et al. [69] in 1991.

There exist multiple works that combine CPGs with RL [70], [71], [72], [73], one of the latest is CPG-RL, a method that combines CPG with RL to learn inputs to the CPG that result in desired locomotion behaviors, in contrast to manually tuning inputs [72].

4.4. Regularization for Safe Behavior

To prevent hardware damage, some precautions have to be taken when training policies in simulation that are intended to be transferred to a real robot. For example, limiting the maximum torque by capping the time integral of torque peaks can help extend motor and joint lifetimes [10]. Limiting changes in joint angles and velocities can help protect motor boards from excessive current spikes, system failures, and help with sim-to-real transferability [65], [74]. The latter is particularly important for the NAO robot[1].

[1]Dutch NAO Team [75] experienced this first-hand when they blew the fuses on two of their NAO robots' chest boards while attempting to transfer a previously in simulation learned policy (G. de Jong, personal communication, 2025-01-19)

5. Crossing the Sim-to-Real Gap

This chapter explores approaches for bridging the sim-to-real gap between simulated and real-world environments. It examines techniques such as DR, online system identification, meta-learning, and comprehensive frameworks that enable zero-shot or few-shot transfer of policies trained in simulation to physical robots. The goal of this chapter is to present SOTA methods that address the challenges of sim-to-real transfer, including physics discrepancies, sensor noise, and asynchronous control, building upon the foundations from Chapter 4 to achieve robust real-world performance after training in simulation.

Sim-to-Real (also referred to as *Sim2Real*) is the process of transferring a policy learned in simulation to the real world, to real hardware. Using standard RL algorithms and simple reward functions can be enough to train a quadruped robot to walk in simulation within 2 to 3 hours. However, transferring such a policy zero-shot usually does not work well [23].

In this context, the sim-to-real gap, also called reality gap [20], [23], is the fundamental problem of the sim-to-real process. The sim-to-real gap is the gap between the simulated environment E' and the real environment E. This gap includes

- inaccurate physics simulation,
- deviations in modeled motor dynamics,
- difficulties in modeling contact forces and material properties such as slip behavior,
- wear and tear of joints and gears,
- changing operating conditions, such as temperature or humidity,
- changing battery behavior, such as changing voltage.

Closing or minimizing this gap is usually difficult, as the above-mentioned factors are hard to model accurately, either due to a lack of knowledge or due to a lack of computational power [20], [76].

A challenge for training in simulation induced by the sim-to-real gap is the Simulation Optimization Bias (SOB) [77], a concept related to the Optimality Gap (OG) that is used in the optimization community. The SOB describes the problem of overfitting to and exploiting small details in simulation, which leads to non-transferable policies.

5.1. Challenges of Real-World Learning

Given these challenges, why is simulation still widely used in practice?

According to Ibarz et al. [23], there are three main challenges for real-world learning of locomotion skills. The first is sample efficiency. DRL often needs tens of millions of data points to learn performant locomotion gaits, which can take months of data collection on a real robot.

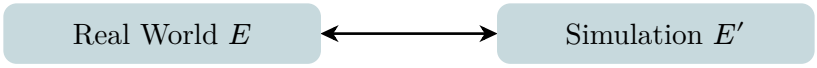
Another challenge is robot safety. During the exploration stage of learning, where new behaviors and states are explored, the robot often tries to execute noisy actuation patterns that cause jerky motions and severely increase wear and tear [10], [23].

The third challenge is *asynchronous control*. Usually, sensor measurements, neural-network inference, and action execution happen simultaneously and asynchronously on a real robot. Thus, the observation that is needed for training may not be the latest, which breaks the fundamental synchronicity assumption of the MDP, which many DRL algorithms rely on [12], [23]. This problem also arises when artificially introducing latencies in simulation, a common practice in DR to reduce the sim-to-real gap [20].

5.2. Information Exchange

Before delving into specific techniques for bridging the sim-to-real gap, it is important to understand the fundamental ways information can be exchanged between simulation E' and the real world E.

One can categorize different approaches to how information is exchanged. Some algorithms work *unidirectionally*, i.e., a policy is trained in simulation and then transferred to the real robot. There, the policy is left as is or is fine-tuned in another training step [20]. This is also called Transfer Learning (TL), as data collected in one domain is transferred to another domain [78], [79]. Fine-tuning is one specific form of TL, where the policy is adapted to the new domain by training it further on data collected in the new domain [80]. In contrast to unidirectional algorithms, some algorithms work *bidirectionally*, i.e., they use data from the real robot to improve the simulation and vice versa in an iterative process [20].



Bidirectional learning is usually more effective and can cope with more complex tasks, but is also more complex to implement and to carry out [20].

For instance, unidirectional approaches are common in DR, where a policy trained in simulation is deployed directly, i.e. zero-shot, on hardware without real-world feedback. In contrast, bidirectional methods, such as Grounded Simulation Learning (GSL) [81], iteratively refine the simulation using real-world data to enhance transferability. There are also bidirectional methods in DR, such as Neural Posterior Domain Randomization (NPDR). Both approaches are discussed in more detail in the following sections.

5.3. Individual Techniques

Having established the fundamental challenges of sim-to-real transfer and the different paradigms for information exchange between simulation and reality, we now turn our attention to the practical techniques that researchers have developed to bridge this gap. If it is not possible to reduce the gap between E' and E by improving the simulation, then the policy has to be more robust to model errors [20]. This section describes the main techniques that are currently most frequently used in literature to achieve this goal.

Regarding hardware, there are two fundamental aspects that need to be addressed: differences in sensing and differences in actuation [22]. The former includes sensors such as cameras, IMU, or joint position sensors, while the latter includes actuators such as motors and servos. This also maps nicely to the formulation of the RL-locomotion problem as a dynamical discrete-time system, where the observation o is the sensor data and the action a is the actuator command, which both need to be adapted to the real world.

Some authors also include other aspects in the idea of sim-to-real, such as the need to perform new tasks that were not trained in simulation [22]. This is not the focus of this work, where the focus is solely on the transfer of a policy learned in simulation to real hardware and to cross the gap between E' and E.

5.3.1. Domain Randomization

DR is based on the idea of training a policy that is robust to a wide range of variations in the environment, by randomizing the environment parameters d during training or by perturbing applied actions a [25], [53], [82]. It is arguably the most popular method and was first introduced in 1995 by Jakobi et al. [83], where different levels of noise are added in simulation to increase the performance of a robot in the real world.

DR facilitates the development of adaptable policies that are effective across a diverse range of environments, rather than being overly tailored to a specific problem instance. The goal of the RL problem then becomes to maximize the expected return for a distribution of environments, rather than maximizing it for one specific environment [84]. During training in simulation, DR usually does not improve performance, but is critical for successful transfer afterwards [85]. It can be seen as a form of regularization to reduce the SOB [77].

Which Parameters to Randomize?

It is important to choose the right parameters to randomize, and randomizing unnecessary parameters can lead to suboptimal performance if applied trivially [86]. For example, parameters that are not relevant to the stability of the task do not need to be randomized.

Relevant parameters can be divided into four groups: sensors, simulation parameters, mechanical properties of the robot, and external factors. Common examples in the literature (not limited to RL-locomotion but excluding vision-related parameters) are:

Category	Parameter	References
Observation noise		[25], [47], [53], [62], [71], [77], [83], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100], [101]
	IMU noise	[53], [62], [64]
	IMU bias	[53]
Simulation Environment		
	Friction	[10], [25], [30], [47], [53], [61], [62], [66], [82], [89], [91], [92], [94], [95], [97], [98], [101], [102], [103], [104], [105], [106], [107], [108], [109]
	Gravity	[82], [94], [95], [98], [101], [104], [105]
	Simulator timestep	[94], [99]
Mechanical properties		
	Mass	[10], [30], [47], [50], [53], [61], [62], [64], [71], [86], [92], [94], [95], [98], [99], [103], [104], [105], [106], [108], [109], [110]
	Center of mass	[10], [62], [64], [94], [97], [102]
	Latencies	[10], [47], [53], [62], [64], [86], [89], [95], [97], [103], [104], [105], [111]
	Inertia	[47], [50], [53], [62], [86], [94], [95], [99], [103], [105], [110]
	Motor strength	[53], [62], [71], [95], [101], [103], [112]
	Motor friction	[101], [103], [104], [106]
	Motor damping	[47], [61], [62], [64], [92], [93], [95], [97], [98], [101], [106], [108], [112]
	Motor response	[86], [95], [99], [104], [106]
	Action noise	[47], [62], [64], [87], [88], [89], [93], [94], [98], [99], [100], [101], [106], [109], [113], [114], [115]
	Offsets	[10], [61], [71], [89], [92]
	Battery voltage	[10], [53]
	Control frequency	[53], [61]
	Scale	[47], [104]

Category	Parameter	References
External disturbances		[10], [62], [64], [92], [95], [101], [104], [113], [116]
	Wind	[25]
	Goal positions	[55], [90], [113], [116]
	Initial states	[30], [54], [55], [87], [90], [91], [96], [100], [101], [107], [108], [110], [113], [115], [116], [117], [118]
	Payload	[10], [62], [66]
	Ground slope	[61], [62], [119]

Table 1: Possible DR parameters to reduce the sim-to-real gap.

This table provides a non-exhaustive collection of parameters that may be randomized. The usefulness of each parameter, however, depends highly on the specific task and the robotic platform. According to the surveyed literature, there is no unified theoretical framework for determining optimal randomization parameters, and the choice of parameters is often based on empirical results.

An important caveat: care must be taken when randomizing latencies, as this introduces asynchronous control, which might break the Markov assumption upon which many RL algorithms are built [12], [23].

Solutions to Asynchronous Control

Ibarz et al. [23] test two SOTA algorithms, Soft Actor-Critic (SAC) [120] and QT-Opt [121], on two robotic control tasks with different latencies. Both algorithms learn efficiently with no latency, but fail when latency is introduced.

One possible solution to this problem is to include previous actions $a^{(t-1)}, a^{(t-2)}, ...$ or previous actions and observations $o^{(t-1)}, o^{(t-2)}, ...$ in the observation space [23], [62], [122], [123], [124], which shows successful results for SAC in practical experiments by Ou and Tavakoli [122]. This approach can be viewed as implicitly constructing an augmented state space that recovers the Markov property, effectively transforming a POMDP into a standard MDP by incorporating sufficient historical information to approximate the true state.

When to Randomize?

Parameters d should be sampled as they could occur in reality. That means that parameters that might change from time step to time step in the real world, such as camera noise, control frequency or random perturbation forces should be sampled every step. Other parameters, such as mass or inertia, should only be sampled every episode [10], [25].

Event-triggered randomization, i.e., randomizing parameters when a specific event occurs, is mentioned as idea in [25], but is not explored further in the surveyed literature.

It can make sense to start with fixed parameters and no randomization and then, for example, linearly increase the amount of randomization during training, acting like a curriculum learning approach. This can aid the policy during the beginning of training, and prevent the adoption of excessively conservative policies [64].

How to Randomize?

Simply randomizing parameters during training can waste modeling power and decrease performance, as it increases the complexity of the task and might even pose a problem for which no single solution exists [20], [30], [86], [125], [126]. This problem naturally grows with the number of parameters that are randomized [127]. With DR, the algorithm has to model the arbitrary perturbations in the input space as well as the dynamics of the underlying world, which are the interesting parts to learn [128]. Moreover, there is evidence that external randomization can destabilize some algorithms, such as Deep Deterministic Policy Gradient (DDPG) [129] and Asynchronous Advantage Actor-Critic (A3C) [91], [128], [130].

Josifovski et al. propose Continual Domain Randomization (CDR) [127], a method that leverages continual learning [131] to reduce the impact of randomizing many different parameters at once. CDR sequentially trains subsets of parameters, indirectly integrating learned knowledge between iterations to mitigate catastrophic forgetting [131].

Muratore et al. [25] distinguish three different ways to randomize, which are *static*, *adaptive* and *adversarial*. With static parameters, the distribution of the parameters is fixed; with dynamic parameters, the distribution is updated during training by data from the target domain. Adversarial means that the distribution is updated such that the policy's performance is minimized but without stopping training completely. Most DR approaches use static randomization, which requires prior knowledge and may lead to suboptimal performance [132], but is usually easier to implement than dynamic or adversarial randomization [133].

Dynamic Approaches

One example of a dynamic approach is Bayesian Domain Randomization (BayRn) [133]. BayRn works by optimizing the source domain distribution, i.e., the randomized parameters, based on returns from the target domain, i.e., the performance of the policy on the real robot. This is as a form of indirect system identification, as the Bayesian Optimization (BO) will converge to sampling from regions with high real-world return where the sampled parameters are close to the "true" parameters.

Another example for a dynamic approach is NPDR [84]. NPDR modifies the distribution of randomized parameters based on real-world data, starting with a user-defined prior, which is then updated iteratively each training round, very similar to BayesSim [134], which follows the same idea but takes some shortcuts, or to SimOpt [49]. A disadvantage of these approaches is that they require real-world rollouts in between. Domain Randomization Off-Policy Optimization (DROPO) [135] fixes this by allowing the posterior to be optimized

based on pre-collected real-world data, which seems most promising for use cases in the RoboCup where large datasets of real-world trajectories are available.

Adversarial Approaches

Automatic Domain Randomization (AuDR) introduced by OpenAI et al. [136] in 2019 is an example of an adversarial approach. AuDR automatically randomizes parameters during training but without real-world feedback by gradually increasing the variance of the distributions, acting like a curriculum learning scheme. This approach demonstrates success in multiple publications [87], [92], [137]. DOmain RAndomization via Entropy MaximizatiON (DORAEMON) [137], a further development, gradually increases the randomized parameter variance as long as the probability of success is sufficiently high.

Mehta et al. [132] introduce an approach similar to AuDR, Active Domain Randomization (AcDR), where parameters are employed that are currently the most difficult for the policy. AcDR requires online feedback from a real robot, making it less feasible for bipedal locomotion tasks.

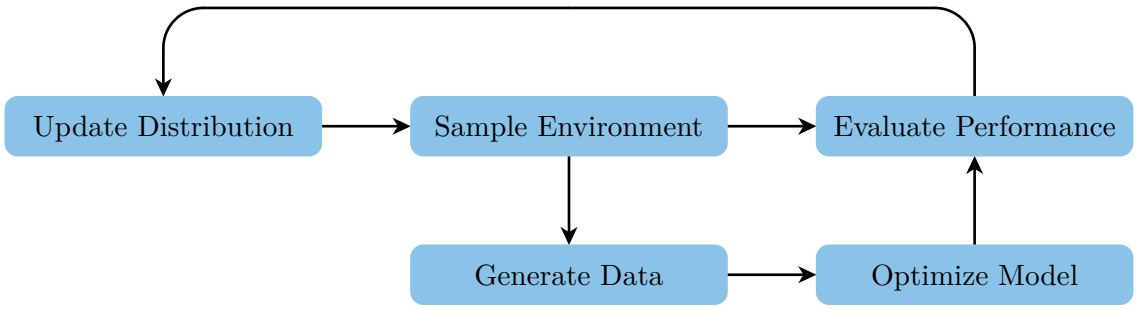


Figure 6: AuDR overview [136].

Another method for automatic DR is Adaptive Curriculum Generation from Demonstrations (ACGD) [138]. As the name suggests, ACGD automatically samples from recorded demonstration trajectories and gradually increases the amount of DR [138]. Also based on demonstrations, Domain Randomization Optimization IDentification (DROID) [139] identifies fitting randomization parameters by iteratively tuning a multivariate normal distribution over simulator dynamics so that torques from a replayed human demonstration match their real-world counterparts [139].

However, whether these last two techniques work for bipedal locomotion remains unclear, as ACGD is designed for vision-based arm control and both ACGD and DROID have thus far only been validated on robotic manipulation tasks.

Physical Plausibility

After randomization, physical parameters should remain realistic. For instance, assigning negative values to mass or inertia can cause numerical instabilities, and there is no evidence

suggesting any benefit to this approach [126], except in vision tasks, where pseudo-random color patterns can increase robustness of object detection algorithms [140], [141].

Online System Identification

As previously discussed, when using DR, the policy π has to learn to handle the perturbations in the input space as well as the underlying environmental dynamics. One way to ease this task is to include the randomized parameters d in the input of π, such that $o' = (o, d)$. This way, the policy π has the advantage of knowing the current parameters d and can adapt its behavior accordingly.

During inference on the real robot, d is not available and has to be inferred online or pre-computed offline. This could be done using a *teacher-student* approach, where the teacher's input is the extended $o' = (o, d)$ and the student's input is o as well as a history of previous observations. The student policy thus implicitly performs online system identification by inferring d from the state history $o^{(t)}, ..., o^{(t-H)}$ [126].

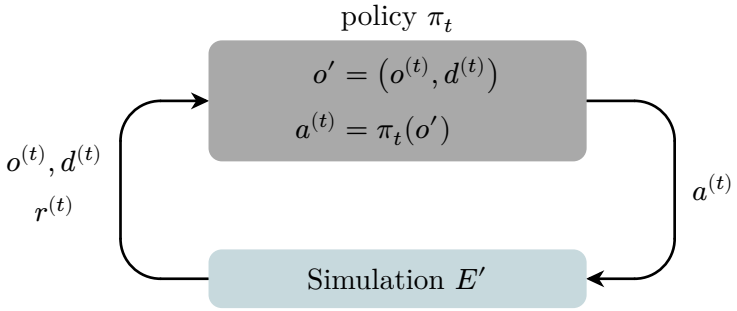


Figure 7: Training of teacher policy π_t.

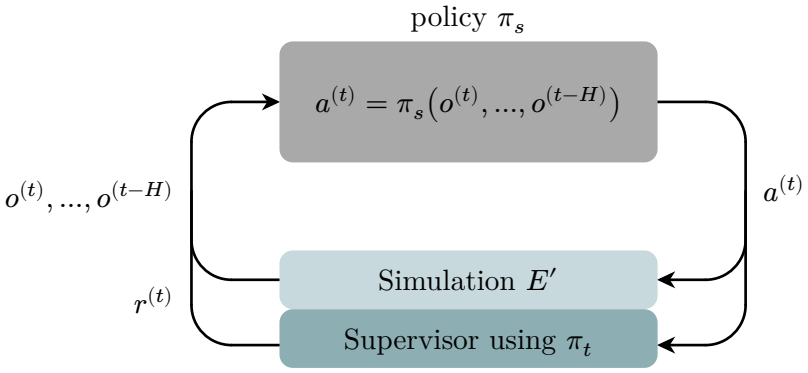


Figure 8: Training of student policy π_s. H denotes the history length. The supervisor provides the training data, for example using Dataset Aggregation (DAgger) [142].

Another idea is to also include the previous actions $a^{(t-1)}, ..., a^{(t-H)}$ in the input. When formulating the problem of dynamic locomotion for a bipedal robot in an uncertain environment as a POMDP, solving the POMDP can be formulated as finding the optimal policy

π^* that maps the process history, which in this control context is the robot's I/O history, to the optimal action [62].

Using a history-dependent policy is generally beneficial with DR, as Chen et al. [124] mathematically show.

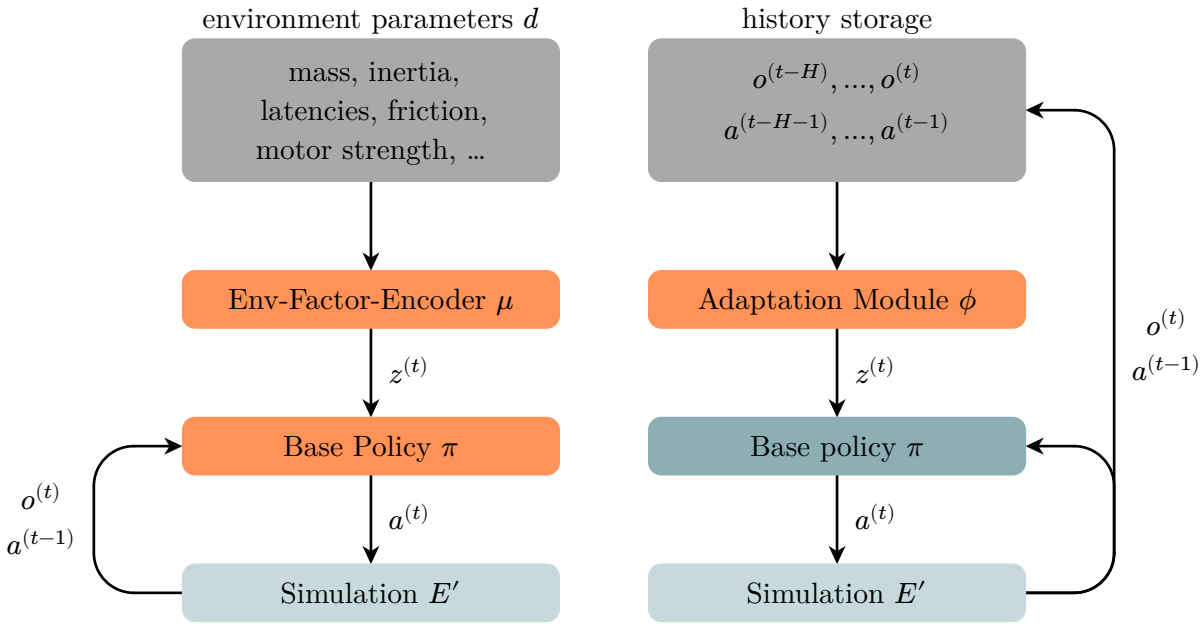


Figure 9: RMA training [66], phase 1 on the left, phase 2 on the right. Trainable modules in orange. History length H.

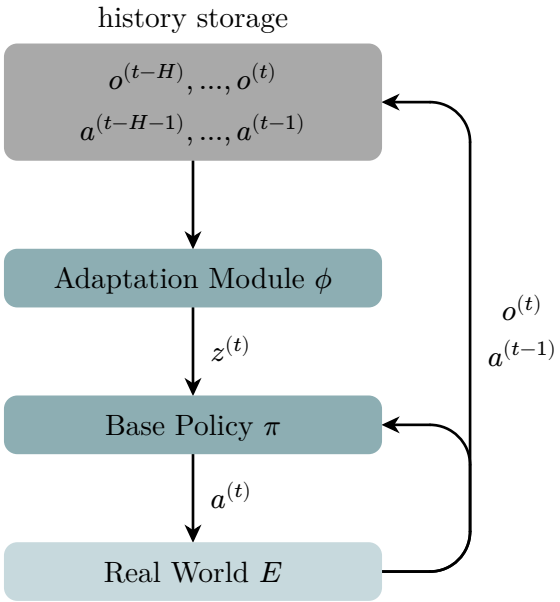


Figure 10: RMA deployment [66].

A very similar approach is RMA [66], with the difference that it does the parameter identification in a separate *Adaptation Module* and that it uses a discrete intermediate latent space z for the representation of the environment parameters d. This offers the advantage of reduced computational overhead, since the adaptation module can be run asynchronously and at a lower frequency than the main policy, with the drawback that it is theoretically less expressive [62].

Zhang et al. [143] propose to use a Transformer-based architecture [144] for online system identification. The transformer performs in-context learning, i.e., in simplified terms, it learns to remember the history of previous environment parameters, actions, and observations to predict future environment parameters.

In contrast to inferring the environment parameters d online, they also can be estimated once beforehand, for example using BO [102], [145], though this can cause the policy to overfit to that specific parameter set. To counteract overfitting, recent work proposes supplying the policy only with a compact, low-dimensional latent encoding of the parameters, thereby introducing an artificial information bottleneck. This bottleneck offers a balance between generalization across different parameters (online) and the possibility to adapt the policy to specific ones (offline) [103].

Meta-Learning-Inspired Online System Identification

In a similar fashion but with influence from *meta-learning*, Variational Policy Embedding (VPE) [145] learns an adaptable master policy over a family of MDPs by embedding them into a low-dimensional latent space, allowing for efficient adaptation to new environments by searching within this learned latent space using BO.

Julian et al. [146] propose an algorithm that learns a latent space of robotic skills. During inference, this algorithm can adapt to new environments by interpolating in the latent space to find an appropriate policy by using live sensor data and online simulation.

Directly estimating parameters is called a *direct* approach, while using a latent encoding or some other form of discrete parameterization is called an *indirect* approach [62]. Both ideas of online system identification show promising results on real hardware, although the former (direct) is more adaptive due to its design, but is also computationally more demanding.

5.3.2. Domain Adaptation

Domain adaptation is a technique that is used when there is a lot of data in one domain, the *source domain* (training data X and test data Y), but data in a different domain is needed, the *target domain* (training data $\widetilde{X}$ and test data $\tilde{Y}$) [147]. To enable domain adaptation, it is important that the characteristics of both domains are similar, i.e., $P(Y|X) \approx P\left(\tilde{Y}|\widetilde{X}\right)$ [148].

The literature review reveals no work that uses domain adaptation for sim-to-real transfer in robotics for motion-related tasks that do not involve vision. For vision tasks, there exist several approaches.

One possibility to perform domain adaptation is to extract the common features into some shared feature space [149]. For vision tasks, this is often done using GANs [52]; in this context, the process is called *image-to-image translation* [150]. This could, for example, be the translation of simulated images from a simulated environment to a real environment, or to transfer both real images and simulated images to a common feature space, where, for example, noise, lighting and texture are omitted [151], [152]. An alternative, for example, would be to directly use depth images, where the sim-to-real fidelity gap is smaller due to the inherently simpler structure of depth images [153], [154]. A disadvantage of GANs is that they require large amounts of real-world training data [155] and further measures need to be taken to avoid collapse and hallucinated objects in translated images, such as a special loss function like the *RL-scene consistency loss* [156].

Another idea to reduce the sim-to-real gap for visual tasks is to go the other way around and do a *real-to-sim* transfer, instead of a *sim-to-real* approach [150], [157], [158]. During training, the policy learns directly on simulated images and during deployment, the camera inputs are translated to look simulated. This direction can be easier, as reducing fidelity and photorealism of real images is simpler than increasing it in simulation. Additionally, the computational overhead during training is minimized [150].

5.3.3. Meta-Learning

Meta-Learning maps nicely to the problem of sim-to-real transfer, as here the goal is to learn a policy that can adapt to the real world and does not only overfit to the simulation environment [29].

This section will present some recent publications that use meta-learning, but without going into too much detail, as this would go beyond the scope of this thesis.

Arndt et al. [30] propose a meta-learning approach, where a *meta policy* that is optimized for fast adaptability is trained to predict latent actions. Working in a latent action space results in faster training and safer on-policy domain adaptation. In the next step, this meta policy is used to train multiple adapted policies for different environments. The results of these adapted policies are then backpropagated to the original meta policy, which is now explicitly optimized to be optimal after a fixed number of adaptation steps. Their experimental results show that this approach can lead to higher success rates after only a few adaptation steps in the real world, compared to fine-tuning a standard RL baseline approach.

One method of adaptation in the real world is to only retrain parts of the policy, such as the last layer, while freezing previous layers. Lončarević et al. [159] show that this can even lead to better results when using a special combination of RL and *Lazy Learning* than retraining the whole policy.

Lee et al. [105] specifically learn an intermediate *sim-to-real policy*, which is trained in simulation in a teacher-student approach. This policy is used to collect data in the real world, which is then used to train a new policy in an offline RL step. They observe that using data from a scripted agent is worse than zero-shot sim-to-real transfer, but that using their sim-to-real policy achieves better results. The reason for this seems to be that the scripted agent

was not able to explore the environment sufficiently, while the sim-to-real policy was able to collect more useful data.

5.4. Frameworks to Facilitate Sim-to-Real Transfer

Besides individual techniques, there are also publications that propose complete frameworks or algorithms to facilitate sim-to-real transfer that are not clearly divisible into common sim-to-real categories. The following presents the most notable approaches. The last section presents other methods that may not be as relevant or applicable to bipedal locomotion, but that could nevertheless be interesting for further research.

5.4.1. Grounded Simulation Learning and Grounded Action Transformation

One example of a complete framework is GSL [81], which describes an iterative approach of repeatedly training a policy in simulation, checking the residuals in the real world, and then updating the simulation parameters such that the residuals between simulation and reality are minimized using some Machine Learning (ML) approach. It follows the principles of *grounding* and *guiding*. Grounding refers to matching the behavior of the robot in simulation to the behavior of the robot in reality. Guiding refers to helping the learning algorithm focus on the important parameters, to enable fast and efficient learning.

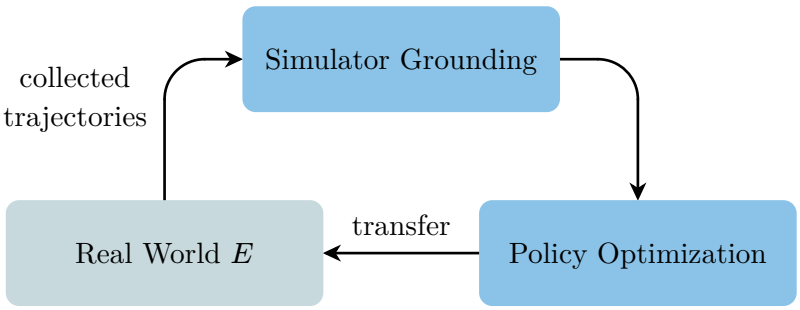


Figure 11: GSL framework overview [81].

Grounded Action Transformation (GAT) [114] and Stochastic Grounded Action Transformation (SGAT) [114] are further developments of GSL. GAT works by directly modifying the agent's actions in simulation rather than the simulator's parameters. SGAT is a generalization of GAT, which works better in environments with noisy state transitions. One benefit of modifying actions is that it allows the problem to be formulated as a simple supervised learning task, without needing to understand the simulator, which can be treated as a non-modifiable black box.

GAT was used to train bipedal walking on a NAO and has shown very promising results that are still, to date, in theory very close to the fastest manually tuned walking algorithms. Hanna et al. [114] achieved a velocity of $279.7\,\mathrm{mm}\,s^{-1}$ in 2021. The fastest manual walking algorithm in the RoboCup SPL by team B-Human achieves $270-300\,\mathrm{mm}\,s^{-1}$ in 2024 [160].

Comparing these two speeds, however, is difficult, since other factors such as stability, changes in walking direction, and energy consumption are not considered.

5.4.2. Simulation Twin

Simulation Twin (SimTwin) [161] is a framework developed for zero-shot sim-to-real transfer of robotic manipulation tasks. Nevertheless, it introduces an interesting approach for direct sim-to-real transfer. SimTwin uses online simulation during inference.

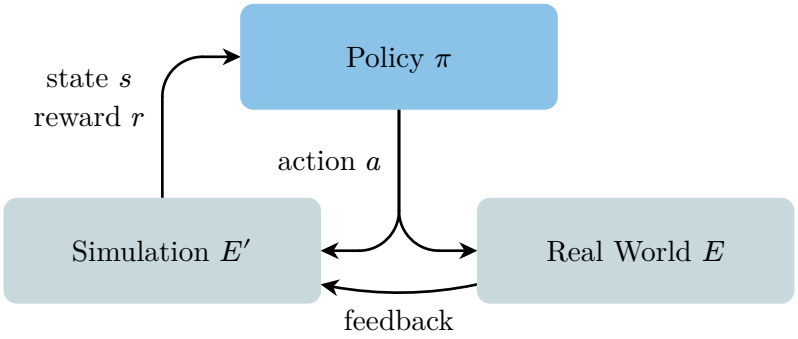


Figure 12: SimTwin framework overview [161].

During inference, the computed actions are applied to the real robot and the simulation. Feedback from the real robot is used to update the domain parameters of the simulation, and the simulation returns the updated state and reward to the policy.

5.4.3. Sim-to-Real Pipelines

Recent publications propose pipelines that combine several sim-to-real techniques to achieve better results.

Valdivia et al. [15] propose a sim-to-real pipeline that combines safe RL, DR and TL. First, a safe RL agent that is based on a CMDP is designed using the Trust Region Conditional Value at Risk (TRC) algorithm [162]. The agent is then trained in simulation with DR, deployed to the real robot, and fine-tuned using real-world data.

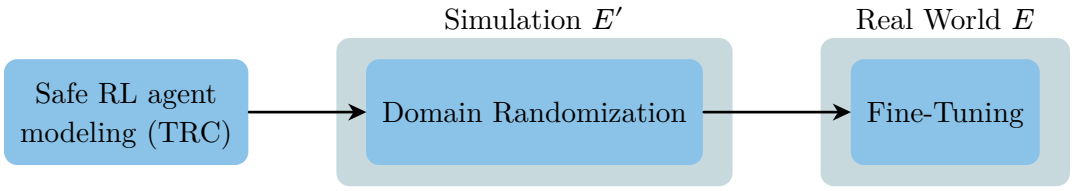


Figure 13: Safe RL, DR, TL pipeline [15].

Their results demonstrate that this integrated solution achieves superior performance in accomplishing tasks, especially in previously unencountered situations, when compared to using these sim-to-real approaches separately [15].

Section 4.1.1 describes a curriculum learning-inspired approach to sim-to-real transfer, where the model complexity is gradually increased during training. Neary et al. [163] propose a similar idea that starts with a low-fidelity simulation, which only implements the dynamics that are most fundamental to the decision-making problem. The initial training in the low-fidelity simulation is then followed by testing the policy in a high-fidelity software-in-the-loop (SIL) simulation, which exposes the policy to the robot's full software stack and more realistic operational delays and conditions. Insights that are gained from this SIL testing are then used to iteratively refine the policy, often by returning to the low-fidelity simulator for further training, to ensure robustness before deployment on the physical hardware.

5.4.4. One-Step Predictive Control

There also exist individual algorithms or ideas to facilitate sim-to-real transfer. Huang et al. [164] propose a one-step predictive control approach to minimize the impact of latencies in robotic systems. Since the execution of the policy π is time-consuming, the actions that are produced by π will be based on outdated sensor data. To mitigate this effect, they propose to predict future observations and operating π directly based on these predictions [164].

The idea of reducing the impact of latencies by predicting future joint angles (even for multiple steps) is not new to the RoboCup community and is already used in the RoboCup SPL [165], [166]. Using this idea in the context of sim-to-real transfer is thus very promising.

5.4.5. Learning Residuals

Using residuals between simulation and real-world trajectories is a common approach in sim-to-real transfer. We have already seen this in the context of GSL [81] and GAT [114].

Schperberg et al. [167] propose a related approach in which residuals between simulation and the real world are predicted using a learning-based Unscented Kalman Filter (UKF). According to their own evaluation, their algorithm is not yet applicable to more complex tasks such as bipedal locomotion, as their approach scales exponentially with the size of the parameter space.

Kaufmann et al. [168] use residuals to adapt the simulation environment for drone racing tasks. They collect a small amount of real-world flight data, i.e., onboard sensory observations alongside pose estimates from a motion-capture system. Dynamics residuals are then predicted using a K-Nearest-Neighbor (KNN) regressor and are used to adapt the simulation environment.

Additional contributions to this field can be found in [169], [170].

5.4.6. Simulation-Based Policy Optimization with Transferability Assessment

Muratore et al. [77] present Simulation-based Policy Optimization with Transferability Assessment (SPOTA), an algorithm that takes the difference between unseen target domains and the set of source domains into account during training. SPOTA optimizes the expected

discounted return with randomized physics parameters, but also provides an approximate probabilistic guarantee on the suboptimality when the policy is applied to a different domain. To achieve this, SPOTA uses the Upper Confidence Bound on the Optimality Gap (UCBOG) as a stopping criterion during training.

5.4.7. Leveraging Progressive Neural Networks

Progressive Neural Networks (PNNs) were introduced to solve complex sequences of tasks, while avoiding catastrophic forgetting and leveraging transfer of previously learned tasks [171]. They thus seem to be a natural choice for sim-to-real tasks.

Meng et al. [172] propose MetaPNN, a meta-learning approach that uses a PNN as its underlying architecture. During the initial phase, a meta-policy is trained on a set of simulated meta-tasks. This pre-trained meta-policy then initializes the first column of the PNN used for transfer to the real-world task. After every task change, the previously learned PNN column is frozen and a new column is added to the PNN. The PNN, with the transferred meta-policy as its initial column and a new column adapted for the real-world task, forms the final target policy for that specific task.

Güitta-López et al. [173] propose a sim-to-real approach leveraging PNNs [171] and DR. However, their paper lacks real-world experiments, and no direct follow-up work on their idea seems to have been conducted.

Using PNNs for sim-to-real transfer is an interesting idea, but according to the surveyed literature it has not been applied to locomotion tasks yet, instead focusing on less complex manipulation tasks.

5.4.8. Other Methods

Akl et al. [174] propose an RL algorithm-agnostic Spiking Neural Network (SNN)-based [175] neuromorphic [176] sim-to-real approach. Their paper focuses on the application to neuromorphic hardware, which is not the focus of this work.

Rizzardo et al. [177] demonstrate a sim-to-real adaptation method in a robotic manipulation task, where a Variational Auto Encoder (VAE) [178], [179] is used to encode high-dimensional observations, such as images, and low-dimensional observations, such as motor positions, into a latent state z. This latent observation z is the only input to the policy $\pi\big(z^{(t)}\big)$. The advantage of this method is that, to fine-tune it in the real world, only the VAE needs to be fine-tuned, and no RL needs to be carried out [177]. However, this approach does not seem applicable to tasks that do not involve vision inputs, as the "low-dimensional" observations are not fed through the VAE and are only concatenated to the latent z.

Robust Generative Adversarial Imitation Learning from Observation (RGAILfO) [180] is an algorithm that builds on robust RL to enable IL for zero-shot sim-to-real transfer. Typically, IL requires the target domain to be the same as the source domain. As IL is not a focus of this work, it will not be discussed further here.

Kumar et al. [181] propose Error-Aware Policy (EAP) learning to inform the policy with an explicit notion of model error already during training in simulation. An auxiliary network predicts the future state deviation that would arise if the action were executed in a domain with different dynamics parameters. The current observation, the known parameters, and the predicted error are combined and fed to the control policy, allowing it to pre-compensate for mismatches in a single forward pass. This approach seems promising, but has not yet been tested in real-world experiments and has not attracted much attention in the community.

6. Discussion

The comprehensive survey of sim-to-real transfer techniques presented in the preceding chapters reveals a constantly evolving field with both significant achievements and persistent challenges. This discussion synthesizes the key findings, examines their implications for robotic locomotion, and identifies critical gaps that remain to be addressed.

6.1. The Dominance and Limitations of Static Domain Randomization

Static DR emerges as the most prevalent technique in the surveyed literature, appearing in the majority of the approximately 80 implementations examined. This dominance can be attributed to several factors: its conceptual simplicity, ease of implementation, and demonstrated effectiveness across diverse robotic platforms and tasks. However, this widespread adoption also masks important limitations that practitioners must consider.

First, the effectiveness of static DR is heavily dependent on the selection and range of randomized parameters [86]. As shown in Table 1, researchers randomize many different parameters, often without much theoretical justification for their choices. This ad-hoc approach suggests that the field lacks a unified theoretical framework for determining optimal randomization strategies.

Second, the task complexity when using DR scales poorly with the number of randomized parameters and the required diversity of training environments [20], [30], [86], [125], [126], [127], since the policy must not only learn the task, but also how to adapt to the constantly changing environment.

Dynamic DR approaches such as BayRn [133], NPDR [84], DROPO [135] and adversarial DR approaches like AuDR [136], AcDR [132], or DORAEMON [137] promise addressing these limitations by adapting the randomization process during training, either by real-world feedback or by learning from the training process itself. Also Online System Identification approaches like RMA [66] can help to mitigate the limitations of static DR by informing the policy about the current state of the system including the current domain parameters.

6.2. The Multi-Technique Imperative

A crucial finding from this survey is that successful sim-to-real transfer in challenging environments rarely relies on a single technique. The complexity of integrating multiple techniques creates a combinatorial explosion of design choices. Each component must be carefully tuned not only for individual performance but also for synergistic interaction with other components. For instance, the choice of observation space design directly impacts the effectiveness of online system identification approaches like RMA [66]. Similarly, curriculum learning schedules must be coordinated with the progression of DR parameters to avoid overwhelming the learning algorithm. This interconnectedness suggests that the field would benefit from more holistic evaluation methodologies. Current research typically evaluates

techniques in limited combinations, making it difficult to understand their relative contributions and interaction effects. Valdivia et al. [15] propose a pipeline combinging safe RL, DR, and fine-tuning, but in the surveyed literature, this is next to [163] the only example of a systematic approach combining multiple techniques for sim-to-real transfer.

The lack of standardized benchmarks for bipedal locomotion further complicates direct comparisons between approaches. There exist benchmarks for measuring the sim-to-real gap [25], [77], [182] which could be used to systematically evaluate the effectiveness of different technique combinations.

6.3. Hardware Constraints as Design Drivers

The influence of hardware constraints on technique selection emerges as a critical theme throughout this survey. Many successful sim-to-real demonstrations utilize high-end platforms with powerful onboard GPUs or offboard computation, without considering the limitations of resource-constrained or edge-platforms. These constraints fundamentally shape which sim-to-real techniques are viable. For example, employing attention-based [144] architectures [183] or large language models [143] for online system identification may be impractical on resource-limited platforms.

Asynchronous control, identified as a major challenge in Section 5.1, illustrates how hardware limitations can invalidate fundamental assumptions of RL algorithms. The standard MDP formulation assumes synchronous state transitions and immediate action execution - assumptions that break down in most real robotic systems. While solutions like I/O history inclusion show promise, they increase computational requirements, creating a tension between theoretical soundness and practical feasibility.

6.4. Bridging Theory and Practice

The gap between theoretical advances and practical implementations represents another challenge for the field. This theory-practice gap is particularly pronounced for bipedal locomotion, where the complexity of the task and the cost of failure create high barriers to experimental validation.

Notably, there exist many works that employ relatively simple techniques but achieve good results on real hardware [57], [62], [65]. Haarnoja et al. [10] for example, train custom-built soccer robots using DRL to autonomously play soccer, i.e., kicking a moving ball, blocking a shot, turning, getting up after falling, strategic defense and recovery from pushes. Compared to a scripted baseline, they achieved a walk-speed increase of 181 %, a turn speed increase of 302 %, and a standup time reduction of 63 %. In their work, they achieve SOTA results with relatively simple techniques: they perform system identification, apply static DR with perturbations, and use regularization for safe behaviors. Li et al. [64] train a bipedal Cassie robot for robust push recovery using a model-free RL approach, achieving good results while employing only static DR.

These empirical successes reveal an interesting pattern: despite the theoretical sophistication available in the literature, practical breakthroughs often emerge from carefully implemented fundamental approaches rather than complex algorithmic innovations. The success of relatively simple approaches like GAT [114] on the NAO platform (achieving $279.7\,\mathrm{mm}\,s^{-1}$ walking speed) compared to more sophisticated methods raises important questions about the trade-off between algorithmic complexity and practical performance. This pattern suggests that the field may benefit from greater emphasis on "simple but effective" solutions that can be reliably deployed on real hardware.

6.5. Case Study for the NAO robot

After surveying the landscape of sim-to-real transfer techniques and identifying key patterns in their application, this section presents a theoretical case study that synthesizes these findings into a concrete proposal. While the preceding chapters examine techniques in isolation or in limited combinations, real-world applications demand a holistic approach that carefully balances multiple competing constraints. The NAO robot operating within the RoboCup Standard Platform League (SPL) provides an ideal possibility for a theoretical case study for this synthesis, as it embodies many of the fundamental challenges identified throughout this survey: computational limitations, control latencies, restricted communication, and the requirement for robust real-time performance.

It is important to note that this pipeline represents a theoretical proposal based on the literature analysis rather than an implemented system. The recommendations draw from successful applications of individual techniques on various platforms, adapted to address the specific challenges of the NAO robot.

The following sections first establish the operational context through an examination of the RoboCup competition and the NAO platform's technical specifications. This foundation then informs the design of a seven-component pipeline that addresses each identified challenge while remaining computationally feasible.

6.5.1. RoboCup

The RoboCup [184] is an annual international robotics competition. It is composed of multiple different leagues, each with its own rules and research focus. Many of the leagues are about robot soccer - but there are also leagues about robots for search and rescue missions, and for industrial or residential use. This work will give additional focus to the SPL, one of the five robot soccer leagues of the RoboCup. In the SPL, teams are required to use the same robot platform, which currently is the NAOv6 by Aldebaran [185], a small humanoid robot. Teams are also not permitted to modify the robot hardware. Because of this restriction, research in the SPL focuses to a large part on algorithm design and their efficient implementation [186].

The SPL imposes strict operational constraints that significantly impact algorithm design. Most importantly, all computation must happen locally on each robot's onboard processor. Once a game starts, robots must operate fully autonomously without any external control

or computation from humans or external computers. This requirement means that all perception, decision-making, and motion control algorithms must run in real-time within the computational constraints. Communication between robots is only allowed in a very limited manner, currently only 1200 messages per team per match are allowed [187]. This forces teams to develop decentralized strategies where each robot must primarily rely on its own sensor data to make decisions, with only minimal coordination messages exchanged for basic team tactics such as role assignment or ball position sharing. These constraints - local computation only and limited communication, combined with the hardware limitations, create unique challenges for RL approaches in the SPL.

6.5.2. NAO

The NAO is a small humanoid robot designed and manufactured by the French robotics company Aldebaran Robotics. It has a height of 58 cm, has 25 degrees of freedom, and is equipped with two cameras, four microphones, two speakers, two pairs of sonar sensors and numerous touch and tactile sensors [188]. The latest version, NAOv6, is used by most teams in the SPL and is powered by an Intel Atom E3845 processor running at 1.91 GHz [189]. Many vital components are housed in the head of the robot, such as the Central Processing Unit (CPU), the cameras, microphones, and speakers. The chest of the NAO robot contains, among other components, a six-axis IMU, the sonar sensors, and the *chestboard*. The chestboard is a microcontroller-powered circuit board responsible for issuing commands to the motors and reading from the joint position encoders, among other tasks [190]. The main CPU in the head communicates with the chestboard over a USB 2.0 connection that runs through the robot's neck joint. This link has limited bandwidth, which is one reason for the low motor and sensor update rate of at most 83 Hz [186].

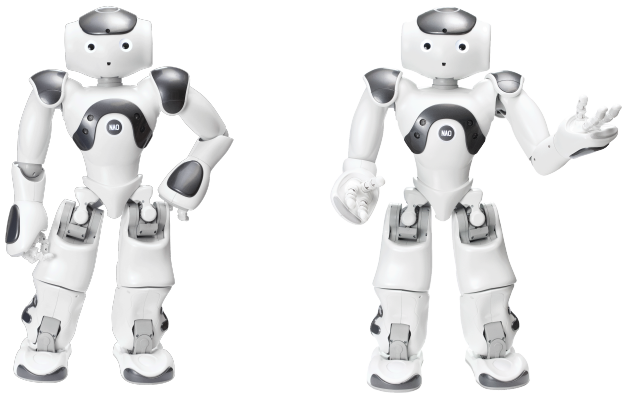


Figure 14: The NAO robot [191].

The NAO robot is very challenging for RL because of its limited hardware capabilities. High sensor and control delays violate the synchronicity assumptions underlying most algorithms, and complex, unknown motor dynamics make it hard to build or learn a good model of the robot. Additionally, the limited computational power of the NAO robot makes it impossible to run large ANNs in real-time, increasing the delay even further and increasing the problem of asynchronous control.

Unlike research platforms where computational resources can be augmented or latencies minimized through hardware modifications, the NAO forces researchers to work within fixed, severe constraints. This makes the NAO an excellent case study for sim-to-real transfer techniques on challenging hardware.

6.5.3. Proposed Pipeline for Sim-to-Real Transfer

The following pipeline represents a theoretical synthesis of techniques surveyed in this thesis, specifically tailored to address the unique challenges of the NAO robot in the RoboCup SPL. While individual components have been validated on various robotic platforms, this integrated approach remains a proposal based on literature analysis rather than empirical validation.

1. **System Identification (Section 4.1)**
   Since no accurate model of the NAOv6 is available, system identification is the first step to approach sim-to-real transfer. Since data from real robots is usually available in the RoboCup SPL due to abundant match recordings, a first step could be to use recorded data to identify a system model. Neural Network (NN)-based residual learning approaches seem promising to additionally improve the model, due to many hard-to-model effects in the real world. On the NAO robot, this includes effects like stuck motors, failing sensors, or overheating joints.
2. **Observation Space Design (Section 4.2)**
   To tackle the notoriously large delays on the NAO robot, I propose to include a limited I/O history of previous observations and actions in the observation space. This should also help to reduce the problem of asynchronous control [23] and enable the possibility to use algorithms like SAC [120] or QT-Opt [121].
3. **Domain Randomization (Section 5.3.1)**
   Used in over 80 studies surveyed, DR remains the most widely validated approach, with consistent success across diverse robotic platforms and domains. Parameters that definitely should be randomized include:
   - Masses
   - Center of mass
   - Latencies
   - Inertia
   - Motor damping
   - Motor strength
   - Friction
   - Observation noise
   - Action noise
   - External disturbances

Other parameters that have only been randomized in some studies but seem relevant for the NAO include:
  - Offsets
  - Control frequency

4. **Using a curriculum-learning approach**
   To facilitate the training process, I propose to use a curriculum-learning approach to gradually increase the difficulty of the tasks and the amount of DR similar to Li et al. [64].

5. **Simulation Grounding (Section 5.4.1)**
   The idea of simulation grounding as introduced by Farchy et al. in GSL [81] and extended by Hanna et al. in GAT [114] shows remarkable results on real hardware. Treating the simulation as a black box and using real-world data to minimize the sim-to-real gap is a promising approach without the need for a perfect simulator and without significant modifications to the training process.

6. **Online System Identification**
   To identify the state of the system during inference, I propose to use an online system identification approach like RMA [66]. RMA shows to be very effective in robotics motion control, offers great adaptability to changing environments, and is computationally tunable by varying the control frequency of Adaptation Module and Base Policy.

7. **Predictive Control (Section 5.4.4)**
   To reduce latencies and improve feedback speed, I propose to use a predictive control approach to predict future observations. This has already shown to be effective in the RoboCup SPL and there exists at least one open source implementation [165], [166].

Note

In many cases, only a few techniques are sufficient to reduce the sim-to-real gap [10], [23], [25]. This suggests starting with a minimal set of the above-mentioned techniques and progressively incorporating more techniques if the sim-to-real gap persists.

7. Conclusion

This survey examines SOTA techniques for bridging the sim-to-real gap in robotic locomotion tasks, with additional attention to bipedal systems like the NAO robot. Through systematic analysis of over 250 relevant papers, this thesis identifies DR as the dominant approach (appearing in more than 80 practical implementations), while also uncovering promising emerging techniques like Simulator Grounding and meta-learning, and drawing valuable parallels from related fields such as robotic manipulation and navigation.

My findings reveal three key insights: First, successful sim-to-real transfer in challenging environments typically requires combining multiple techniques rather than relying on a single approach. Second, the choice of techniques heavily depends on the specific hardware constraints - particularly relevant for resource-limited platforms like the NAO with its 83 Hz control frequency and significant latencies. Third, bidirectional learning approaches, though more complex to implement, consistently outperform unidirectional methods for challenging locomotion tasks.

7.1. Addressing the Research Questions

RQ1: Several primary categories of techniques to mitigate the sim-to-real gap are identified through systematic analysis. These include robust system identification (both offline parameter optimization and RL-based approaches), diverse forms of DR (static, dynamic, and adversarial variants), online system identification for real-time adaptation (including history-dependent policies and latent parameter encoding), meta-learning approaches for faster adaptation to new environments, simulation grounding techniques like GSL [81] and GAT [114] to iteratively align simulation and real-world dynamics, one-step predictive control to counteract system latencies, residual learning methods, and leveraging advanced NN architectures like PNNs for transfer learning. Beyond individual techniques, comprehensive frameworks such as SimTwin [161], SPOTA [77], and integrated sim-to-real pipelines [15], [163] emerge as important solutions that combine multiple approaches, reinforcing the key finding that successful transfer typically requires synergistic application of complementary techniques rather than relying on any single method - particularly for resource-constrained platforms where hardware limitations heavily influence technique selection.

RQ2: For humanoid robotics and the NAO specifically, techniques that explicitly handle asynchronous control (I/O history inclusion), predictive control for latency compensation, and grounded simulation learning demonstrate particular promise, with GAT [114] achieving competitive walking speeds of $279.7\,\mathrm{mm}\,s^{-1}$ on the NAO platform. The complete pipeline proposed in Section 6.5.3 is specifically designed with the NAO robot in mind, addressing its unique hardware constraints through each of its seven components. This NAO-centric design ensures that the pipeline is not merely a generic sim-to-real approach, but one tailored to succeed within the strict computational and communication constraints of the SPL environment.

7.2. Limitations

This survey is subject to certain limitations that warrant acknowledgment:

1. Comparison of different techniques is challenging: Many surveyed papers do not provide shared benchmarks or standardized evaluation metrics, making direct comparisons difficult.
2. Rapid field evolution: With the fast pace of robotics advancement, some findings may become outdated quickly, particularly regarding computational constraints.
3. Limited real-world validation: Multiple surveyed techniques show promise in simulation but lack extensive real-world testing on diverse hardware platforms.
4. Computational cost analysis: Most papers do not report specific computational costs or resource requirements, making it hard to assess feasibility for resource-constrained platforms like the NAO.

7.3. Future Work

Several promising research directions emerge from this survey that could further advance sim-to-real transfer in legged robotics:

7.3.1. Emerging AI/ML Techniques

Recent advances in large language models and foundation models open new possibilities for sim-to-real transfer:

1. In-context learning for system identification: Zhang et al. [143] demonstrate using Transformers for dynamics learning, suggesting that attention mechanisms could enable more adaptive online system identification without explicit parameter estimation. In a similar manner Qu et al. [183] use attention-augmented memory to inform the policy about changing dynamics.
2. Diffusion models for trajectory generation: While not yet applied to bipedal locomotion, diffusion models' success in other domains [192], [193] suggests potential for generating diverse, physically plausible motion primitives.

7.3.2. Standardization and Benchmarking

As discussed in Section 6.2, the field would benefit from the adoption of more standardized benchmarks such as [77], [182] for evaluating sim-to-real transfer in robotics locomotion. Current ad-hoc evaluations make it difficult to compare techniques or track progress over time. To compare the effectiveness of different approaches, benchmarks for measuring the sim-to-real gap could be systematically used to evaluate the effectiveness of different technique combinations.

7.3.3. Hardware-Specific Optimizations

For custom-built or modifiable hardware platforms, where computational resources can be adjusted or where communication with external systems is not artificially constrained, such as in the case of the NAO robot and the RoboCup SPL:

1. Neuromorphic computing: Leveraging event-based processing could reduce latency and enable real-time adaptation despite limited CPU resources.
2. Edge-cloud hybrid architectures: Offloading complex computations while maintaining low-latency control loops.

7.4. Closing Remarks

This survey provides a robust foundation for advancing sim-to-real transfer in legged robotics. The pipeline proposed in Section 6.5.3, drawing from a wide array of successful strategies, offers a pathway to developing more capable and adaptable robots that can be deployed in real-world environments.

The ultimate goal of RoboCup, creating humanoid robots capable of competing with human soccer champions by 2050, remains ambitious. This survey demonstrates that while significant progress has been made in sim-to-real transfer, the integration of these techniques into robust, deployable systems for dynamic tasks like soccer remains a challenge in closed or restricted systems. The path forward requires not just algorithmic innovation but also careful consideration of the complete pipeline from simulation to real-world deployment.

8. References

[1] L. Mädje, M. Haug, and The Typst Project Developers, "Typst (Version 0.13.1)." Accessed: Jun. 23, 2025. [Online]. Available: https://github.com/typst/typst

[2] "The LaTeX Project." Accessed: Jun. 23, 2025. [Online]. Available: https://www.latex-project.org/

[3] OpenAI *et al.*, "ChatGPT." Accessed: Jun. 23, 2025. [Online]. Available: https://openai.com/blog/chatgpt/

[4] Gemini Team *et al.*, "Gemini: A Family of Highly Capable Multimodal Models." Accessed: Jun. 23, 2025. [Online]. Available: https://arxiv.org/abs/2312.11805

[5] Anthropic, "Claude Opus." Accessed: Jun. 23, 2025. [Online]. Available: https://www.anthropic.com/claude/opus

[6] DeepSeek-AI *et al.*, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning." Accessed: Jun. 23, 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[7] DeepL SE, "DeepL Translator." [Online]. Available: https://www.deepl.com/write

[8] World Economic Forum, "The Future of Jobs Report 2023," 2023. Accessed: Jun. 23, 2025. [Online]. Available: https://www.weforum.org/publications/the-future-of-jobs-report-2023/

[9] A. S. Duggal *et al.*, "A sequential roadmap to Industry 6.0: Exploring future manufacturing trends," *IET Communications*, vol. 16, no. 5, pp. 521–531, 2022, doi: 10.1049/cmu2.12284.

[10] T. Haarnoja *et al.*, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *Science Robotics*, vol. 9, no. 89, Apr. 2024, doi: 10.1126/scirobotics.adi8022.

[11] B. v. Marum, A. Shrestha, H. Duan, P. Dugar, J. Dao, and A. Fern, "Revisiting Reward Design and Evaluation for Robust Humanoid Standing and Walking," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 11256–11263. doi: 10.1109/IROS58592.2024.10802680.

[12] J. Kober and J. Peters, "Reinforcement Learning in Robotics: A Survey," in *Reinforcement Learning: State-of-the-Art*, M. Wiering and M. van Otterlo, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 579–610. doi: 10.1007/978-3-642-27645-3_18.

[13] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[14] M. T. J. Spaan, "Partially Observable Markov Decision Processes," in *Reinforcement Learning: State-of-the-Art*, M. Wiering and M. van Otterlo, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 387–414. doi: 10.1007/978-3-642-27645-3_12.

[15] J. P. Valdivia, A. Hata, and A. Terra, "Safe and Robust Simulation-to-Reality Transfer for Mobile Robots," in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2024, pp. 1–8. doi: 10.1109/ETFA61755.2024.10711026.

[16] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained Policy Optimization," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., in Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 22–31. [Online]. Available: https://proceedings.mlr.press/v70/achiam17a.html

[17] R. Sutton and A. Barto, "Reinforcement Learning: An Introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, p. 1054, 1998, doi: 10.1109/TNN.1998.712192.

[18] J. Achiam and OpenAI, "Spinning Up in RL." Accessed: Jan. 16, 2025. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro.html

[19] A. Pinosky, I. Abraham, A. Broad, B. Argall, and T. D. Murphey, "Hybrid control for combining model-based and model-free reinforcement learning," *The International Journal of Robotics Research*, vol. 42, no. 6, pp. 337–355, 2023, doi: 10.1177/02783649221083331.

[20] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning," *IEEE Access*, vol. 9, no. , pp. 153171–153187, 2021, doi: 10.1109/ACCESS.2021.3126658.

[21] D. J. Mankowitz *et al.*, "Robust Reinforcement Learning for Continuous Control with Model Misspecification," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=HJgC60EtwB

[22] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," 2020, pp. 737–744. doi: 10.1109/SSCI47803.2020.9308468.

[23] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4–5, pp. 698–721, Jan. 2021, doi: 10.1177/0278364920987859.

[24] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," 2015, [Online]. Available: https://arxiv.org/abs/1503.02531

[25] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters, "Robot Learning From Randomized Simulations: A Review," *Frontiers in Robotics and AI*, vol. 9, p. 19, May 2022, doi: 10.26083/tuprints-00021227.

[26] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by Cheating," in *Proceedings of the Conference on Robot Learning*, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., in Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 66–75. [Online]. Available: https://proceedings.mlr.press/v100/chen20a.html

[27] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, in ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 41–48. doi: 10.1145/1553374.1553380.

[28] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation Learning: A Survey of Learning Methods," *ACM Comput. Surv.*, vol. 50, no. 2, Apr. 2017, doi: 10.1145/3054912.

[29] J. Schmidhuber, "Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook," 1987. [Online]. Available: http://www.idsia.ch/~juergen/diploma.html

[30] K. Arndt, M. Hazara, A. Ghadirzadeh, and V. Kyrki, "Meta Reinforcement Learning for Sim-to-real Domain Adaptation," in *2020 IEEE/ICRA International Conference on Robotics and Automation*, 2020, pp. 2725–2731. doi: 10.1109/ICRA40945.2020.9196540.

[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, in ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1126–1135.

[32] D. Denyer and D. Tranfield, "Producing a systematic review," in *The SAGE Handbook of Organizational Research Methods*, Thousand Oaks, CA: Sage Publications Ltd, 2009, pp. 671–689. [Online]. Available: https://archive.org/details/isbn_9781412931182/

[33] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, in EASE '14. London, England, United Kingdom: Association for Computing Machinery, 2014. doi: 10.1145/2601248.2601268.

[34] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017, doi: 10.1109/MSP.2017.2743240.

[35] A. S. Polydoros and L. Nalpantidis, "Survey of Model-Based Reinforcement Learning: Applications on Robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, May 2017, doi: 10.1007/s10846-017-0468-y.

[36] C. E. Counsell, "Formulating Questions and Locating Primary Studies for Inclusion in Systematic Reviews," *Annals of Internal Medicine*, vol. 127, pp. 380–387, 1997, [Online]. Available: https://api.semanticscholar.org/CorpusID:33532748

[37] "HULKs - RoboCup SPL Team." Accessed: Jun. 11, 2024. [Online]. Available: https://hulks.de/

[38] "Scopus." Accessed: Jun. 11, 2024. [Online]. Available: https://www.scopus.com/

[39] "IEEE Xplore." Accessed: Jun. 11, 2024. [Online]. Available: https://ieeexplore.ieee.org/

[40] "Springer Nature Link." Accessed: Jun. 11, 2024. [Online]. Available: https://link.springer.com/

[41] "Multidisciplinary Digital Publishing Institute." Accessed: Jun. 11, 2024. [Online]. Available: https://www.mdpi.com/

[42] "Sage Journals." Accessed: Jun. 11, 2024. [Online]. Available: https://journals.sagepub.com/

[43] T. Lee, J. Kwon, P. M. Wensing, and F. C. Park, "Robot Model Identification and Learning: A Modern Perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, no. Volume7, 2024, pp. 311–334, 2024, doi: 10.1146/annurev-control-061523-102310.

[44] G. Sutanto *et al.*, "Encoding Physical Constraints in Differentiable Newton-Euler Algorithm," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Eds., in Proceedings of Machine Learning Research, vol. 120. PMLR, 2020, pp. 804–813. [Online]. Available: https://proceedings.mlr.press/v120/sutanto20a.html

[45] M. Lutter, J. Silberbauer, J. Watson, and J. Peters, "Differentiable Physics Models for Real-world Offline Model-based Reinforcement Learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2021, pp. 4163–4170. doi: 10.1109/ICRA48506.2021.9561805.

[46] A. Allevato, E. S. Short, M. Pryor, and A. Thomaz, "TuneNet: One-Shot Residual Tuning for System Identification and Sim-to-Real Robot Task Transfer," in *Proceedings of the Conference on Robot Learning*, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., in Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 445–455. [Online]. Available: https://proceedings.mlr.press/v100/allevato20a.html

[47] Y. Jiang *et al.*, "SimGAN: Hybrid Simulator Identification for Domain Adaptation via Adversarial Reinforcement Learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 2884–2890. doi: 10.1109/ICRA48506.2021.9561731.

[48] Y. Du, O. Watkins, T. Darrell, P. Abbeel, and D. Pathak, "Auto-Tuned Sim-to-Real Transfer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1290–1296. doi: 10.1109/ICRA48506.2021.9562091.

[49] Y. Chebotar *et al.*, "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World

Experience," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8973–8979. doi: 10.1109/ICRA.2019.8793789.

[50] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Learning Locomotion Skills for Cassie: Iterative Design and Sim-to-Real," in *Proceedings of the Conference on Robot Learning*, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., in Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 317–329. [Online]. Available: https://proceedings.mlr.press/v100/xie20a.html

[51] M. R. Diprasetya, A. N. Pullani, and A. Schwung, "Sim-to-Real Transfer for Robotics Using Model-Free Curriculum Reinforcement Learning," in *2024 IEEE International Conference on Industrial Technology (ICIT)*, 2024, pp. 1–6. doi: 10.1109/ICIT58233.2024.10540995.

[52] I. J. Goodfellow *et al.*, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., 2014, pp. 2672–2680. doi: 10.1145/3422622.

[53] J. Tan *et al.*, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, Jun. 2018. doi: 10.15607/RSS.2018.XIV.010.

[54] A. Wasala, D. Byrne, P. Miesbauer, J. O'Hanlon, P. Heraty, and P. Barry, "Trajectory based lateral control: A Reinforcement Learning case study," *Engineering Applications of Artificial Intelligence*, vol. 94, p. 103799, 2020, doi: 10.1016/j.engappai.2020.103799.

[55] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang, "DexPoint: Generalizable Point Cloud Reinforcement Learning for Sim-to-Real Dexterous Manipulation," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., in Proceedings of Machine Learning Research, vol. 205. PMLR, 2023, pp. 594–605. [Online]. Available: https://proceedings.mlr.press/v205/qin23a.html

[56] E. Aljalbout, F. Frank, M. Karl, and P. van der Smagt, "On the Role of the Action Space in Robot Manipulation Learning and Sim-to-Real Transfer," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5895–5902, 2024, doi: 10.1109/LRA.2024.3398428.

[57] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Robust Feedback Motion Policy Design Using Reinforcement Learning on a 3D Digit Bipedal Robot," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic: IEEE Press, 2021, pp. 5136–5143. doi: 10.1109/IROS51168.2021.9636467.

[58] W. Zhang, L. Ott, M. Tognon, and R. Siegwart, "Learning Variable Impedance Control for Aerial Sliding on Uneven Heterogeneous Surfaces by Proprioceptive and Tactile Sensing," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11275–11282, 2022, doi: 10.1109/LRA.2022.3194315.

[59] R. Deits and T. Koolen, "Picking up momentum," *Blog post, Boston Dynamics*, Jan. 2023, [Online]. Available: https://www.bostondynamics.com/blog/picking-momentum

[60] L. Bao, J. Humphreys, T. Peng, and C. Zhou, "Deep Reinforcement Learning for Bipedal Locomotion: A Brief Survey." [Online]. Available: https://arxiv.org/abs/2404.17070

[61] J. Siekmann, K. R. Green, J. Warila, A. Fern, and J. Hurst, "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning ," in *Proceedings of Robotics: Science and Systems (RSS)*, May 2021. doi: 10.15607/RSS.2021.XVII.061.

[62] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *The International Journal of Robotics Research*, vol. 44, no. 5, pp. 840–888, 2025, doi: 10.1177/02783649241285161.

[63] Y. Yao *et al.*, "AnyBipe: An End-to-End Framework for Training and Deploying Bipedal Robots Guided by Large Language Models." [Online]. Available: https://arxiv.org/abs/2409.08904

[64] Z. Li *et al.*, "Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 2811–2817. doi: 10.1109/ICRA48506.2021.9560769.

[65] A. Duburcq, F. Schramm, G. Boéris, N. Bredeche, and Y. Chevaleyre, "Reactive Stepping for Humanoid Robots using Reinforcement Learning: Application to Standing Push Recovery on the Exoskeleton Atalante," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 9302–9309. doi: 10.1109/IROS47612.2022.9982234.

[66] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid Motor Adaptation for Legged Robots," in *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, D. A. Shell, M. Toussaint, and M. A. Hsieh, Eds., 2021. doi: 10.15607/RSS.2021.XVII.011.

[67] W. Tan *et al.*, "A Hierarchical Framework for Quadruped Omnidirectional Locomotion Based on Reinforcement Learning," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 4, pp. 5367–5378, 2024, doi: 10.1109/TASE.2023.3310945.

[68] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008, doi: 10.1016/j.neunet.2008.03.014.

[69] G. Taga, Y. Yamaguchi, and H. Shimizu, "Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment," *Biological Cybernetics*, vol. 65, no. 3, pp. 147–159, Jul. 1991, doi: 10.1007/BF00198086.

[70] T. Matsubara, J. Morimoto, J. Nakanishi, M. Sato, and K. Doya, "Learning CPG-based biped locomotion with a policy gradient method," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, 2005, pp. 208–213. doi: 10.1109/ICHR.2005.1573569.

[71] G. Bellegarda, M. Shafiee, and A. Ijspeert, "Visual CPG-RL: Learning Central Pattern Generators for Visually-Guided Quadruped Locomotion," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 1420–1427. doi: 10.1109/ICRA57147.2024.10611128.

[72] G. Bellegarda and A. Ijspeert, "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion ," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12547–12554, 2022, doi: 10.1109/LRA.2022.3218167.

[73] H. Kimura, Y. Fukuoka, Y. Hada, and K. Takase, "Adaptive Dynamic Walking of a Quadruped Robot on Irregular Terrain Using a Neural System Model," in *Robotics Research*, R. A. Jarvis and A. Zelinsky, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 147–160.

[74] J. Lee, L. Schroth, V. Klemm, M. Bjelonic, A. Reske, and M. Hutter, "Exploring Constrained Reinforcement Learning Algorithms for Quadrupedal Locomotion," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 11132–11138. doi: 10.1109/IROS58592.2024.10801341.

[75] Dutch NAO Team, Accessed: Jan. 19, 2025. [Online]. Available: https://www.dutchnaoteam.nl/

[76] M. Mozian, J. Camilo Gamboa Higuera, D. Meger, and G. Dudek, "Learning Domain Randomization Distributions for Training Robust Locomotion Policies," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 6112–6117. doi: 10.1109/IROS45743.2020.9341019.

[77] F. Muratore, M. Gienger, and J. Peters, " Assessing Transferability From Simulation to Reality for Reinforcement Learning ," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 43, no. 4, pp. 1172–1183, Apr. 2021, doi: 10.1109/TPAMI.2019.2952353.

[78] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010, doi: 10.1109/TKDE.2009.191.

[79] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning," *Sensors*, vol. 21, no. 4, 2021, doi: 10.3390/s21041278.

[80] A. Zhang, Z. Lipton, M. Li, and A. Smola, *Dive Into Deep Learning*. Cambridge University Press, 2023. [Online]. Available: https://d2l.ai/

[81] A. Farchy, S. Barrett, P. MacAlpine, and P. Stone, "Humanoid Robots Learning to Walk Faster: From the Real World to Simulation and Back," in *Proc. of 12th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2013. [Online]. Available: http://www.cs.utexas.edu/users/ai-lab?AAMAS13-Farchy

[82] J. v. Baar, A. Sullivan, R. Cordorel, D. Jha, D. Romeres, and D. Nikovski, "Sim-to-Real Transfer Learning using Robustified Controllers in Robotic Tasks involving Complex Dynamics," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6001–6007. doi: 10.1109/ICRA.2019.8793561.

[83] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Advances in Artificial Life*, F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 704–720.

[84] F. Muratore, T. Gruner, F. Wiese, B. Belousov, M. Gienger, and J. Peters, "Neural Posterior Domain Randomization," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., in Proceedings of Machine Learning Research, vol. 164. PMLR, 2022, pp. 1532–1542. [Online]. Available: https://proceedings.mlr.press/v164/muratore22a.html

[85] T. Chaffre, J. Moras, A. Chan-Hon-Tong, and J. Marzat, "Sim-to-Real Transfer with Incremental Environment Complexity for Reinforcement Learning of Depth-based Robot Navigation," in *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics - ICINCO*, SciTePress, 2020, pp. 314–323. doi: 10.5220/0009821603140323.

[86] Z. Xie, X. Da, M. van de Panne, B. Babich, and A. Garg, "Dynamics Randomization Revisited: A Case Study for Quadrupedal Locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4955–4961. doi: 10.1109/ICRA48506.2021.9560837.

[87] B. Balaji *et al.*, "DeepRacer: Autonomous Racing Platform for Experimentation with Sim2Real Reinforcement Learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2746–2754. doi: 10.1109/ICRA40945.2020.9197465.

[88] P. Egli and M. Hutter, "A General Approach for the Automation of Hydraulic Excavator Arms Using Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5679–5686, 2022, doi: 10.1109/LRA.2022.3152865.

[89] Z. Ding, Y.-Y. Tsai, W. W. Lee, and B. Huang, "Sim-to-Real Transfer for Robotic Manipulation with Tactile Sensory," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6778–6785. doi: 10.1109/IROS51168.2021.9636259.

[90] A. Iriondo, E. Lazkano, A. Ansuategi, A. Rivera, I. Lluvia, and C. Tubío, "Learning positioning policies for mobile manipulation operations with deep reinforcement learning," *International Journal of Machine*

Learning and Cybernetics, vol. 14, no. 9, pp. 3003–3023, Sep. 2023, doi: 10.1007/s13042-023-01815-8.

[91] A. Orsula, S. Bøgh, M. Olivares-Mendez, and C. Martinez, "Learning to Grasp on the Moon from 3D Octree Observations with Deep Reinforcement Learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 4112–4119. doi: 10.1109/IROS47612.2022.9981661.

[92] L. Leyendecker, M. Schmitz, H. A. Zhou, V. Samsonov, M. Rittstieg, and D. Lütticke, "Deep Reinforcement Learning for Robotic Control in High-Dexterity Assembly Tasks - A Reward Curriculum Approach," in *2021 Fifth IEEE International Conference on Robotic Computing (IRC)*, 2021, pp. 35–42. doi: 10.1109/IRC52146.2021.00012.

[93] R. Jitosho, T. G. W. Lum, A. Okamura, and K. Liu, "Reinforcement Learning Enables Real-Time Planning and Control of Agile Maneuvers for Soft Robot Arms," in *Proceedings of The 7th Conference on Robot Learning*, J. Tan, M. Toussaint, and K. Darvish, Eds., in Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 1131–1153. [Online]. Available: https://proceedings.mlr.press/v229/jitosho23a.html

[94] C. Alessi, D. Bianchi, G. Stano, M. Cianchetti, and E. Falotico, "Pushing with Soft Robotic Arms via Deep Reinforcement Learning," *Advanced Intelligent Systems*, vol. 6, no. 8, p. 2300899, 2024, doi: 10.1002/aisy.202300899.

[95] P. Klokowski *et al.*, "evoBOT – Design and Learning-Based Control of a Two-Wheeled Compound Inverted Pendulum Robot," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 10425–10432. doi: 10.1109/IROS55552.2023.10342128.

[96] X. Zhang, S. Jain, B. Huang, M. Tomizuka, and D. Romeres, "Learning Generalizable Pivoting Skills," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5865–5871. doi: 10.1109/ICRA48891.2023.10161271.

[97] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3803–3810. doi: 10.1109/ICRA.2018.8460528.

[98] T. G. W. Lum *et al.*, "DextrAH-G: Pixels-to-Action Dexterous Arm-Hand Grasping with Geometric Fabrics," in *Proceedings of the Conference on Robot Learning (CoRL)*, Sep. 2024. [Online]. Available: https://sites.google.com/view/dextrah-g

[99] L. Sacchetto, M. Korte, S. Gronauer, M. Kissel, and K. Diepold, "Offline Reinforcement Learning for Quadrotor Control: Overcoming the Ground Effect," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 7539–7544. doi: 10.1109/IROS55552.2023.10341599.

[100] M. Ranaweera and Q. Mahmoud, "Evaluation of Techniques for Sim2Real Reinforcement Learning," *The International FLAIRS Conference Proceedings*, vol. 36, no. 1, May 2023, doi: 10.32473/flairs.36.133317.

[101] A. Scaldaferri *et al.*, "Otto—Design and Control of an 8-DoF SEA-Driven Quadrupedal Robot," *IEEE Open Journal of the Industrial Electronics Society*, vol. 6, no. , pp. 820–839, 2025, doi: 10.1109/OJIES.2025.3567112.

[102] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, "Sim-to-Real Transfer for Biped Locomotion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China: IEEE Press, 2019, pp. 3503–3510. doi: 10.1109/IROS40897.2019.8968053.

[103] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning Agile Robotic Locomotion Skills by Imitating Animals," in *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*, M. Toussaint, A. Bicchi, and T. Hermans, Eds., 2020. doi: 10.15607/RSS.2020.XVI.064.

[104] A. Handa *et al.*, "DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5977–5984. doi: 10.1109/ICRA48891.2023.10160216.

[105] A. X. Lee *et al.*, "Beyond Pick-and-Place: Tackling Robotic Stacking of Diverse Shapes," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., in Proceedings of Machine Learning Research, vol. 164. PMLR, 2022, pp. 1089–1131. [Online]. Available: https://proceedings.mlr.press/v164/lee22b.html

[106] Q. Wang *et al.*, "Dexterous robotic manipulation using deep reinforcement learning and knowledge transfer for complex sparse reward-based tasks," *Expert Systems*, vol. 40, no. 6, p. e13205, 2023, doi: 10.1111/exsy.13205.

[107] W. Hu, B. Huang, W. W. Lee, S. Yang, Y. Zheng, and Z. Li, "Dexterous in-hand manipulation of slender cylindrical objects through deep reinforcement learning with tactile sensing," *Robotics and Autonomous Systems*, vol. 186, p. 104904, 2025, doi: 10.1016/j.robot.2024.104904.

[108] Y. Zhou, Y. Jin, P. Lu, S. Jiang, Z. Wang, and B. He, "T-TD3: A Reinforcement Learning Framework for Stable Grasping of Deformable Objects Using Tactile Prior," *IEEE Transactions on Automation Science and Engineering*, vol. 22, no. , pp. 6208–6222, 2025, doi: 10.1109/TASE.2024.3440047.

[109] J. J. Rothert, S. Lang, M. Seidel, and M. Hanses, "Sim-to-Real Transfer for a Robotics Task: Challenges and Lessons Learned," in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2024, pp. 1–8. doi: 10.1109/ETFA61755.2024.10711073.

[110] B. Habas and B. Cheng, "From Flies to Robots: Inverted Landing in Small Quadcopters With Dynamic Perching," *IEEE Transactions on Robotics*, vol. 41, no. , pp. 1773–1790, 2025, doi: 10.1109/TRO.2025.3543263.

[111] S. W. Abeyruwan *et al.*, "i-Sim2Real: Reinforcement Learning of Robotic Policies in Tight Human-Robot Interaction Loops," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., in Proceedings of Machine Learning Research, vol. 205. PMLR, 2023, pp. 212–224. [Online]. Available: https://proceedings.mlr.press/v205/abeyruwan23a.html

[112] A. Church, J. Lloyd, R. Hadsell, and N. F. Lepora, "Tactile Sim-to-Real Policy Transfer via Real-to-Sim Image Translation," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., in Proceedings of Machine Learning Research, vol. 164. PMLR, 2022, pp. 1645–1654. [Online]. Available: https://proceedings.mlr.press/v164/church22a.html

[113] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Optimizing walking controllers for uncertain inputs and environments ," *ACM Trans. Graph.*, vol. 29, no. 4, Jul. 2010, doi: 10.1145/1778765.1778810.

[114] J. P. Hanna, S. Desai, H. Karnan, G. Warnell, and P. Stone, "Grounded action transformation for sim-to-real reinforcement learning," *Machine Learning*, vol. 110, no. 9, pp. 2469–2499, Sep. 2021, doi: 10.1007/s10994-021-05982-z.

[115] R. Xiao, C. Yang, Y. Jiang, and H. Zhang, "One-shot sim-to-real transfer policy for robotic assembly via reinforcement learning with visual demonstration," *Robotica*, vol. 42, no. 4, pp. 1074–1093, 2024, doi: 10.1017/S0263574724000092.

[116] V. Kumar, D. Hoeller, B. Sundaralingam, J. Tremblay, and S. Birchfield, "Joint Space Control via Deep Reinforcement Learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3619–3626. doi: 10.1109/IROS51168.2021.9636477.

[117] S. Christen, W. Yang, C. Perez-D'Arpino, O. Hilliges, D. Fox, and Y.-W. Chao, "Learning Human-to-Robot Handovers from Point Clouds," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2023, pp. 9654–9664. doi: 10.1109/CVPR52729.2023.00931.

[118] S. Iqbal *et al.*, "Toward Sim-to-Real Directional Semantic Grasping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 7247–7253. doi: 10.1109/ICRA40945.2020.9197310.

[119] A. Kalapos, C. Gór, R. Moni, and I. Harmati, "Sim-to-real reinforcement learning applied to end-to-end vehicle control," in *2020 23rd International Symposium on Measurement and Control in Robotics (ISMCR)*, 2020, pp. 1–6. doi: 10.1109/ISMCR51255.2020.9263751.

[120] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., in Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 1861–1870. [Online]. Available: https://proceedings.mlr.press/v80/haarnoja18b.html

[121] K.-H. Lee, T. Xiao, A. Li, P. Wohlhart, I. Fischer, and Y. Lu, "PI-QT-Opt: Predictive Information Improves Multi-Task Robotic Reinforcement Learning at Scale," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., in Proceedings of Machine Learning Research, vol. 205. PMLR, 2023, pp. 1696–1707. [Online]. Available: https://proceedings.mlr.press/v205/lee23a.html

[122] Y. Ou and M. Tavakoli, "Sim-to-Real Surgical Robot Learning and Autonomous Planning for Internal Tissue Points Manipulation Using Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2502–2509, 2023, doi: 10.1109/LRA.2023.3254860.

[123] H. Chan and U. Ozguner, "Closed-loop control of systems over a communication network with queues," in *Proceedings of 1994 American Control Conference - ACC '94*, 1994, pp. 811–815. doi: 10.1109/ACC.1994.751855.

[124] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang, "Understanding Domain Randomization for Sim-to-real Transfer," in 10th International Conference on Learning Representations, ICLR 2022. doi: 10.48550/arXiv.2110.03239.

[125] B. Osiński *et al.*, "Simulation-Based Reinforcement Learning for Real-World Autonomous Driving," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6411–6418. doi: 10.1109/ICRA40945.2020.9196730.

[126] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 572–587, 2024, doi: 10.1177/02783649231224053.

[127] J. Josifovski, S. Auddy, M. Malmir, J. Piater, A. Knoll, and N. Navarro-Guerrero, "Continual Domain Randomization," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 4965–4972. doi: 10.1109/IROS58592.2024.10802060.

[128] S. James *et al.*, " Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks ," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2019, pp. 12619–12629. doi: 10.1109/CVPR.2019.01291.

[129] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San*

Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1509.02971

[130] V. Mnih *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., in Proceedings of Machine Learning Research, vol. 48. New York, New York, USA: PMLR, 2016, pp. 1928–1937. [Online]. Available: https://proceedings.mlr.press/v48/mniha16.html

[131] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019, doi: 10.1016/j.neunet.2019.01.012.

[132] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active Domain Randomization," in *Proceedings of the Conference on Robot Learning*, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., in Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 1162–1176. [Online]. Available: https://proceedings.mlr.press/v100/mehta20a.html

[133] F. Muratore, C. Eilers, M. Gienger, and J. Peters, "Data-Efficient Domain Randomization With Bayesian Optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 911–918, 2021, doi: 10.1109/LRA.2021.3052391.

[134] R. Possas, L. Barcelos, R. Oliveira, D. Fox, and F. Ramos, "Online BayesSim for Combined Simulator Parameter Inference and Policy Improvement," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5445–5452. doi: 10.1109/IROS45743.2020.9341401.

[135] G. Tiboni, K. Arndt, and V. Kyrki, "DROPO: Sim-to-real transfer with offline domain randomization," *Robotics and Autonomous Systems*, vol. 166, p. 104432, 2023, doi: 10.1016/j.robot.2023.104432.

[136] OpenAI *et al.*, "Solving Rubik's Cube with a Robot Hand." Accessed: Jun. 23, 2025. [Online]. Available: https://arxiv.org/abs/1910.07113

[137] G. Tiboni, P. Klink, J. Peters, T. Tommasi, C. D'Eramo, and G. Chalvatzaki, "Domain Randomization via Entropy Maximization," in *12th International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024. [Online]. Available: https://openreview.net/forum?id=GXtmuiVrOM

[138] L. Hermann, M. Argus, A. Eitel, A. Amiranashvili, W. Burgard, and T. Brox, "Adaptive Curriculum Generation from Demonstrations for Sim-to-Real Visuomotor Control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6498–6505. doi: 10.1109/ICRA40945.2020.9197108.

[139] Y.-Y. Tsai, H. Xu, Z. Ding, C. Zhang, E. Johns, and B. Huang, "DROID: Minimizing the Reality Gap Using Single-Shot Human Demonstration," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3168–3175, 2021, doi: 10.1109/LRA.2021.3062311.

[140] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning*, 2017, pp. 334–343.

[141] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric Actor Critic for Image-Based Robot Learning," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, Jun. 2018. doi: 10.15607/RSS.2018.XIV.008.

[142] S. Ross, G. Gordon, and D. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., in Proceedings of Machine Learning Research, vol. 15. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 627–635. [Online]. Available: https://proceedings.mlr.press/v15/ross11a.html

[143] X. Zhang *et al.*, "Dynamics as Prompts: In-Context Learning for Sim-to-Real System Identifications," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3190–3197, 2025, doi: 10.1109/LRA.2025.3540391.

[144] A. Vaswani *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, in NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[145] I. Arnekvist, D. Kragic, and J. A. Stork, "VPE: Variational Policy Embedding for Transfer Reinforcement Learning," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 36–42. doi: 10.1109/ICRA.2019.8793556.

[146] R. Julian *et al.*, "Scaling simulation-to-real transfer by learning a latent space of robot skills," *The International Journal of Robotics Research*, vol. 39, no. 10–11, pp. 1259–1278, 2020, doi: 10.1177/0278364920944474.

[147] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1, pp. 151–175, May 2010, doi: 10.1007/s10994-009-5152-4.

[148] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *2011 International Conference on Computer Vision*, 2011, pp. 999–1006. doi: 10.1109/ICCV.2011.6126344.

[149] H. Ju, R. Juan, R. Gomez, K. Nakamura, and G. Li, "Transferring policy of deep reinforcement learning from simulation to reality for robotics," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1077–1087, 2022, doi: 10.1038/s42256-022-00573-6.

[150] J. Zhang *et al.*, "VR-Goggles for Robots: Real-to-Sim Domain Adaptation for Visual Control," *IEEE*

Robotics and Automation Letters, vol. 4, no. 2, pp. 1148–1155, 2019, doi: 10.1109/LRA.2019.2894216.

[151] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, "RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 10920–10926. doi: 10.1109/ICRA48506.2021.9561157.

[152] B. Xu, T. Hassan, and I. Hussain, "Seg-CURL: Segmented Contrastive Unsupervised Reinforcement Learning for Sim-to-Real in Visual Robotic Manipulation," *IEEE Access*, vol. 11, no. , pp. 50195–50204, 2023, doi: 10.1109/ACCESS.2023.3278208.

[153] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "DayDreamer: World Models for Physical Robot Learning," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., in Proceedings of Machine Learning Research, vol. 205. PMLR, 2023, pp. 2226–2240. [Online]. Available: https://proceedings.mlr.press/v205/wu23c.html

[154] X. Zhang *et al.*, "Close the Optical Sensing Domain Gap by Physics-Grounded Active Stereo Sensor Simulation," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2429–2447, 2023, doi: 10.1109/TRO.2023.3235591.

[155] P. M. Scheikl *et al.*, "Sim-to-Real Transfer for Visual Reinforcement Learning of Deformable Object Manipulation for Robot-Assisted Surgery," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 560–567, 2023, doi: 10.1109/LRA.2022.3227873.

[156] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, " RL-CycleGAN: Reinforcement Learning Aware Simulation-to-Real ," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2020, pp. 11154–11163. doi: 10.1109/CVPR42600.2020.01117.

[157] O.-M. Pedersen, E. Misimi, and F. Chaumette, "Grasping Unknown Objects by Coupling Deep Reinforcement Learning, Generative Adversarial Networks, and Visual Servoing," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 5655–5662. doi: 10.1109/ICRA40945.2020.9197196.

[158] M. Sasaki, J. Muguro, F. Kitano, W. Njeri, and K. Matsushita, "Sim–Real Mapping of an Image-Based Robot Arm Controller Using Deep Reinforcement Learning," *Applied Sciences*, vol. 12, no. 20, 2022, doi: 10.3390/app122010277.

[159] Z. Lončarević, M. Simonič, A. Ude, and A. Gams, "Combining Reinforcement Learning and Lazy Learning for Faster Few-Shot Transfer Learning," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 285–290. doi: 10.1109/Humanoids53995.2022.10000095.

[160] P. Reichenberg and T. Röfer, "Dynamic Joint Control for a Humanoid Walk," in *RoboCup 2023: Robot World Cup XXVI*, C. Buche, A. Rossi, M. Simões, and U. Visser, Eds., in Lecture Notes in Artificial Intelligence, vol. 14140. Springer, 2024, pp. 215–227.

[161] Y. Chen, C. Zeng, Z. Wang, P. Lu, and C. Yang, "Zero-shot sim-to-real transfer of reinforcement learning framework for robotics manipulation with demonstration and force feedback," *Robotica*, vol. 41, no. 3, pp. 1015–1024, 2023, doi: 10.1017/S0263574722001230.

[162] D. Kim and S. Oh, "TRC: Trust Region Conditional Value at Risk for Safe Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2621–2628, 2022, doi: 10.1109/LRA.2022.3141829.

[163] C. Neary, C. Ellis, A. S. Samyal, C. Lennon, and U. Topcu, "A Multifidelity Sim-to-Real Pipeline for Verifiable and Compositional Reinforcement Learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 4349–4355. doi: 10.1109/ICRA57147.2024.10610735.

[164] X. Huang, X. Wang, Y. Zhao, J. Hu, H. Li, and Z. Jiang, "Guided Model-Based Policy Search Method for Fast Motor Learning of Robots With Learned Dynamics," *IEEE Transactions on Automation Science and Engineering*, vol. 22, no. , pp. 453–465, 2025, doi: 10.1109/TASE.2024.3352580.

[165] J. Fiedler, "Gelenkwinkelvorhersage für Laufbewegungen humanoider Roboter mittels neuronaler Netze," 2023. [Online]. Available: https://b-human.de/downloads/theses/master-thesis-jfiedler.pdf

[166] B-Human Team, "B-Human Code Release." [Online]. Available: https://docs.b-human.de/coderelease2024/

[167] A. Schperberg, Y. Tanaka, F. Xu, M. Menner, and D. Hong, "Real-to-Sim: Predicting Residual Errors of Robotic Systems with Sparse Data using a Learning-Based Unscented Kalman Filter," in *2023 20th International Conference on Ubiquitous Robots (UR)*, 2023, pp. 27–34. doi: 10.1109/UR57808.2023.10202521.

[168] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, Aug. 2023, doi: 10.1038/s41586-023-06419-4.

[169] J. Gao, M. Y. Michelis, A. Spielberg, and R. K. Katzschmann, "Sim-to-Real of Soft Robots With Learned Residual Physics," *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8523–8530, 2024, doi: 10.1109/LRA.2024.3446287.

[170] R. Ferede, C. De Wagter, D. Izzo, and G. C. de Croon, "End-to-end Reinforcement Learning for Time-Optimal Quadcopter Flight ," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6172–6177. doi: 10.1109/ICRA57147.2024.10611665.

[171] A. Rusu *et al.*, "Progressive Neural Networks." Google DeepMind, 2016. doi: 10.48550/arXiv.1606.04671.

[172] W. Meng, H. Ju, T. Ai, R. Gomez, E. Nichols, and G. Li, "Transferring Meta-Policy From Simulation to Re-

ality via Progressive Neural Network," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3696–3703, 2024, doi: 10.1109/LRA.2024.3370034.

[173] L. Güitta-López, J. Boal, and Á. J. López-López, "Learning more with the same effort: how randomization improves the robustness of a robotic deep reinforcement learning agent," *Applied Intelligence*, vol. 53, no. 12, pp. 14903–14917, Jun. 2023, doi: 10.1007/s10489-022-04227-3.

[174] M. Akl, Y. Sandamirskaya, D. Ergene, F. Walter, and A. Knoll, "Fine-tuning Deep Reinforcement Learning Policies with r-STDP for Domain Adaptation," in *Proceedings of the International Conference on Neuromorphic Systems 2022*, in ICONS '22. Knoxville, TN, USA: Association for Computing Machinery, 2022. doi: 10.1145/3546790.3546804.

[175] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997, doi: 10.1016/S0893-6080(97)00011-7.

[176] M. Davies *et al.*, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018, doi: 10.1109/MM.2018.112130359.

[177] C. Rizzardo, F. Chen, and D. Caldwell, "Sim-to-real via latent prediction: Transferring visual non-prehensile manipulation policies," *Frontiers in Robotics and AI*, 2023, doi: 10.3389/frobt.2022.1067502.

[178] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. doi: 10.48550/arXiv.1312.6114.

[179] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., in Proceedings of Machine Learning Research, vol. 32. Bejing, China: PMLR, 2014, pp. 1278–1286. [Online]. Available: http://proceedings.mlr.press/v32/rezende14.pdf

[180] L. Viano, Y.-T. Huang, P. Kamalaruban, C. Innes, S. Ramamoorthy, and A. Weller, "Robust Learning from Observation with Model Misspecification," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, in AAMAS '22. Virtual Event, New Zealand: International Foundation for Autonomous Agents, Multiagent Systems, 2022, pp. 1337–1345.

[181] V. Kumar, S. Ha, and C. K. Liu, "Error-Aware Policy Learning: Zero-Shot Generalization in Partially Observable Dynamic Environments," in *Proceedings of Robotics: Science and Systems*, Virtual, Jul. 2021. doi: 10.15607/RSS.2021.XVII.065.

[182] M. Kegeleirs, D. G. Ramos, K. Hasselmann, L. Garattoni, G. Francesca, and M. Birattari, "Transferability in the Automatic Off-Line Design of Robot Swarms: From Sim-to-Real to Embodiment and Design-Method Transfer Across Different Platforms," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2758–2765, 2024, doi: 10.1109/LRA.2024.3360013.

[183] J. Qu, S. Otsubo, T. Yamanokuchi, T. Matsubara, and S. Miwa, "Domain Randomization-free Sim-to-Real : An Attention-Augmented Memory Approach for Robotic Tasks," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 5056–5063. doi: 10.1109/IROS58592.2024.10801944.

[184] RoboCup Federation, "RoboCup." Accessed: Jan. 09, 2025. [Online]. Available: https://www.robocup.org/

[185] RoboCup Federation, "RoboCup Standard Platform League." Accessed: Jan. 09, 2025. [Online]. Available: https://spl.robocup.org/

[186] R. Mecklenburg, "Evaluating Model Predictive Control for Step Planning on Humanoid Robots in the RoboCup Standard Platform League," in *RoboCup 2025: Robot World Cup XXVIII*, in Lecture Notes in Artificial Intelligence. Springer, 2025, p. to appear. [Online]. Available: https://hulks.de/_files/PA_mecklenburg_2025_Mecklenburg.pdf

[187] RoboCup Federation, "RoboCup Standard Platform League Rules." Accessed: Jun. 17, 2024. [Online]. Available: https://spl.robocup.org/downloads/

[188] Aldebaran Robotics, "NAO the humanoid and programmable robot." Accessed: Jan. 10, 2025. [Online]. Available: https://www.aldebaran.com/en/nao

[189] Aldebaran Robotics, "NAO - Technical Specifications." Accessed: Jan. 09, 2025. [Online]. Available: https://support.aldebaran.com/support/solutions/articles/80000959718-nao-technical-specifications

[190] R. Gelin, "NAO," in *Humanoid Robotics: A Reference*, A. Goswami and P. Vadakkepat, Eds., Dordrecht: Springer Netherlands, 2018, pp. 1–22. doi: 10.1007/978-94-007-7194-9_14-1.

[191] Aldebaran Robotics, "full-nao.png." Accessed: Oct. 26, 2024. [Online]. Available: https://www.aldebaran.com/themes/custom/softbank/images/full-nao.png

[192] X. Huang *et al.*, "DiffuseLoco: Real-Time Legged Locomotion Control with Diffusion from Offline Datasets," in *Proceedings of The 8th Conference on Robot Learning*, P. Agrawal, O. Kroemer, and W. Burgard, Eds., in Proceedings of Machine Learning Research, vol. 270. PMLR, 2025, pp. 1567–1589. [Online]. Available: https://proceedings.mlr.press/v270/huang25a.html

[193] D. Valevski, Y. Leviathan, M. Arar, and S. Fruchter, "Diffusion Models Are Real-Time Game Engines," in *International Conference on Learning Representations (ICLR)*, 2025. [Online]. Available: https://arxiv.org/abs/2408.14837