Guided Reinforcement Learning for Dynamic Control: A Case Study on a Two-Wheeled Robot

To obtain the academic degree of

Dr.-Ing.

from the Faculty of Mechanical Engineering
of TU Dortmund University
approved dissertation

Julian Eßer, M.Sc.
born in
Kamp-Lintfort

Day of Oral Examination: 23.09.2025

1. Examiner: Prof. Dr. Dr. h. c. Michael ten Hompel
2. Examiner: Prof. Dr. Sebastian Peitz
3. Examiner: Prof. Dr. Maren Bennewitz

Dortmund, 2025

Abstract

Autonomous mobile robots have the potential to transform industries by performing various tasks, such as material handling and transportation. As technology advances towards dynamic robots capable of interacting with their environments, controlling these systems becomes increasingly challenging. Learning-based control offers a promising approach, enabling robots to adapt to new tasks based on their experiences in dynamic environments. Such adaptability is crucial for success in real-world applications.
This dissertation explores Guided Reinforcement Learning (RL) methodologies aimed at addressing common challenges such as training speed, policy performance, and the transfer from simulation to reality, all of which are essential for deploying robots in real-world scenarios. The first part of this thesis introduces a novel taxonomy for Guided RL that systematically incorporates external knowledge sources, including expert demonstrations and prior experiences. This approach enhances the efficiency and effectiveness of the training process. The dissertation also includes a comprehensive survey of Guided RL methods and practical guidelines for their application in various robotics contexts, allowing researchers and practitioners to choose the most suitable strategies for their specific needs.
The second part includes an extensive case study on learning-based control for evoBOT, a dynamic two-wheeled robot developed by the Fraunhofer Institute for Material Flow and Logistics IML. This study illustrates how Guided RL techniques can significantly improve learning-based control, resulting in rapid training, robustness against disturbances, and successful transfer of trained policies to the physical robot. Lastly, the thesis introduces a generalized action space for robot control, providing a formulation that operates directly at the torque level. Benchmarking evaluates the impact of different action spaces on training efficiency and policy performance across multiple robotic platforms, offering valuable insights into optimizing control strategies.
Overall, this work contributes to the advancement of robot learning, offering valuable insights and methodologies for effectively training and implementing learning-based control strategies in complex, dynamic robotic systems. Therefore, this work lays the groundwork for improving the capabilities of autonomous robots in real-world applications, promoting greater integration of robotics into everyday life. Key results of this work, including publications, the robot simulation model, and the training code, are available for open access to encourage further research in this field.

Keywords: Embodied AI, Robotics, Reinforcement Learning, Dynamic Locomotion, Two-Wheeled Robot, Physics Simulation, Sim-to-Real Transfer, Action Space Design.

Kurzfassung

Autonome mobile Roboter haben das Potenzial die Industrie zu transformieren, indem sie verschiedene Aufgaben wie Materialhandhabung und Transport übernehmen. Mit dem technologischen Fortschritt hin zu dynamischen Robotern, die mit ihrer Umgebung interagieren können, wird die Steuerung dieser Systeme immer komplexer. Lernbasierte Regelung bietet einen vielversprechenden Ansatz, der es den Robotern ermöglicht, sich auf der Grundlage ihrer Erfahrungen in dynamischen Umgebungen an neue Aufgaben anzupassen. Eine solche Anpassungsfähigkeit ist entscheidend für den Erfolg in realen Anwendungen.

Diese Dissertation bietet eine umfassende Untersuchung von Methoden des geführten Verstärkungslernens (Guided Reinforcement Learning, RL), welche darauf abzielen die Trainingsgeschwindigkeit und Performance zu verbessern, sowie die Übertragbarkeit der lernbasierten Regler von der Simulation auf die realen Roboter zu ermöglichen. Der erste Teil dieser Arbeit stellt eine neuartige Taxonomie für Guided RL vor, die systematisch externe Wissensquellen einbezieht. Dieser Ansatz steigert die Effizienz und Effektivität des Trainingsprozesses. Die Dissertation enthält auch eine umfassende Studie zu Guided RL Methoden sowie praktische Richtlinien für deren Anwendung in verschiedenen Kontexten der Robotik, die es Forschern und Anwendern ermöglichen, die am besten geeigneten Strategien für ihre individuellen Anforderungen auszuwählen. Der zweite Teil der Arbeit enthält eine umfassende Fallstudie zur lernbasierten Regelung von evoBOT, einem dynamischen zweirädrigen Roboter, der vom Fraunhofer-Institut für Materialfluss und Logistik IML entwickelt wurde. Diese Studie veranschaulicht, wie Guided-RL-Techniken die gelernten Regler erheblich verbessern können, was zu schnellerem Training, Robustheit gegenüber Störungen und der erfolgreichen Übertragung der trainierten Regler auf den physischen Roboter führt. Schließlich wird im dritten Teil dieser Arbeit ein verallgemeinerter Aktionsraum für die Robotersteuerung eingeführt, der eine Formulierung bietet, die direkt auf Drehmomentebene arbeitet. Dieser Aspekt der Arbeit bewertet die Auswirkungen verschiedener Aktionsräume auf die Performance gelernter Regler für verschiedene Roboterplattformen und bietet wertvolle Einblicke in die Optimierung von Regelungsstrategien.

Insgesamt leistet diese Arbeit einen Beitrag zur Weiterentwicklung des Roboterlernens und bietet wertvolle Einblicke und Methoden für ein effektives Training und die Implementierung lernbasierter Regelstrategien in komplexen, dynamischen Robotern. Somit legt diese Arbeit den Grundstein für die Verbesserung der Fähigkeiten autonomer Roboter in realen Anwendungen und das tägliche Leben. Zentrale Ergebnisse dieser Arbeit, einschließlich der Veröffentlichungen, des Simulationsmodells und des Trainingscodes, sind frei zugänglich, um die weitere Forschung auf diesem Gebiet zu fördern.

Schlagworte: Embodied AI, Robotik, Reinforcement Learning, Dynamische Fortbewegung, Zwei-Rädriger Roboter, Physiksimulation, Sim-to-Real-Transfer, Aktionsraum.

Acknowledgments

This thesis is the result of my work as a research assistant at the Fraunhofer Institute for Material Flow and Logistics IML in Dortmund, in collaboration with the Lamarr Institute for Machine Learning and Artificial Intelligence and TU Dortmund University.
First and foremost, I would like to express my gratitude to my supervisor, Prof. Dr. Dr. h.c. Michael ten Hompel, the former managing director of Fraunhofer IML, professor at the Material Handling and Warehousing Chair at TU Dortmund University, and director of the Lamarr Institute. I am grateful for the opportunity to benefit from such a unique ecosystem and infrastructure, as well as for the continuous support and scientific freedom.
I also would like to thank the members of my thesis committee, Prof. Dr. Sebastian Peitz, Prof. Dr. Maren Bennewitz, and Prof. Dr.-Ing. Alice Kirchheim, for their time, effort, and invaluable insights into this work.
On the Fraunhofer side, I would like to extend special thanks to my previous manager, Dr.-Ing. Sören Kerner, for consistently allowing me the freedom to pursue my ideas and for his invaluable feedback on this thesis. This work would not have been possible without my exceptional colleagues. I particularly want to acknowledge my close collaborators, Dr.-Ing. Oliver Urbann, Nicolas Bach, and Christian Jestel, for the numerous theoretical and practical discussions we had in the field of robot learning. I also appreciate the contributions of my outstanding students, especially Shubham Bajpai, Frido Feldmeier, Praveen Rajendran, and Ashin Anandakrishnan. Additionally, I would like to thank the evoBOT team, especially Patrick Klokowski, Benedikt Pschera, and Nils Gramse, for their continuous support in conducting the real robot experiments.
I was fortunate to collaborate with some of the brightest minds in the emerging field of robot learning worldwide. I want to thank MIT CSAIL for providing me the opportunity for a research visit in Boston, especially Prof. Pulkit Agrawal and Gabriel Margolis, for our fantastic joint work on advancing the possibilities in reinforcement learning and sim-to-real transfer. Additionally, I am grateful to Liila Lorabi, Gavriel State, and Spencer Huang from NVIDIA for our close collaboration and for inviting me to Silicon Valley. Furthermore, I thank Renato Gasoto and Kelly Guo for the excellent joint work on integrating the robot simulation model and training code into state-of-the-art frameworks, making these results accessible to the broader robot learning community.
Lastly, none of this would have been possible without the support of my friends and family. I want to thank my parents for raising me with love and curiosity, and my little brother for reminding me of the fun side of life. Finally, I express my deepest appreciation to my wonderful wife, Monique, for her endless love and our little son Linus, whose smile makes me forget everything else in a second. I am forever grateful to have you in my life.

Dortmund, Spring 2025 *Julian Eßer*

Financial Support

The research underlying this work received financial support from the Federal Ministry of Transport and Digital Infrastructure of Germany, through the Silicon Economy Logistics Ecosystem; the Federal Ministry of Education and Research of Germany and the state of North Rhine-Westphalia, as part of the Lamarr Institute for Machine Learning and Artificial Intelligence and the AI Junior Research Group DynaFoRo.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
AMR	Autonomous Mobile Robot
CAD	Computer Aided Design
CoM	Center of Mass
DRL	Deep Reinforcement Learning
ELU	Exponential Linear Unit
GAE	Generalized Advantage Estimation
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit
MDP	Markov Decision Process
MLP	Multi-Layer Perception
ML	Machine Learning
ONNX	Open Neural Network Exchange
PD	Proportional-Derivative
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
ROS	Robot Operating System
Sim-to-Real	Simulation-to-Reality
TWIP	Two-Wheeled Inverted Pendulum
URDF	Unified Robot Description Format
USD	Universal Scene Description

1 Introduction

This introduction outlines the scope of the dissertation, which aims to advance the field of robot learning. It begins with a motivation for advanced robotic systems, highlighting learning-based control as a promising avenue for automation. Following this, the chapter outlines the specific objectives, research questions this thesis aims to address, and the scientific contributions made in this area. Finally, a brief overview of the structure of the thesis is provided.

1.1. Motivation

Humanity has always been fascinated by the potential of intelligent robots to assist with everyday tasks. These machines are designed to perform jobs that are too dangerous, heavy, or repetitive for humans to handle regularly. To operate autonomously, they require cognitive reasoning similar to that of humans. Recent advancements in artificial intelligence (AI) have created new opportunities for developing these cognitive abilities in autonomous systems. This progress could transform not only industries but also everyday life by enhancing human capabilities and improving the overall quality of life.

Embodied AI, also known as physical AI, explores the integration of AI into physical systems, such as robots, enabling them to interact meaningfully with their surroundings [80, 46]. This field encompasses all aspects of interaction and learning, from perception and understanding to reasoning, planning, and execution. Unlike traditional AI models that often operate in abstract, virtual environments, Embodied AI emphasizes the importance of physical presence and interaction. An AI system learns and understands its environment through direct engagement, similar to how humans and animals learn through exploration and interaction.

In line with the Triangular AI paradigm, one approach to advancing Embodied AI involves structuring machine learning (ML) methods to enhance the perception and planning of complex interactions between agents and their environments [193]. This methodology enables agents to learn from limited experiences and adapt their behavior accordingly. Simulation plays a critical role in creating a digital reality where agents can learn, relearn, and validate their actions [95]. The practical application of Embodied AI demonstrates its relevance beyond theoretical interests, impacting various domains such as logistics, manufacturing, and production.

The growing significance of automation in various industries is fueled by labor shortages, increasing efficiency demands, and the potential for enhanced productivity. The logistics sector, in particular, faces significant challenges, such as an aging workforce and rising consumer expectations for faster delivery times. As a result, companies are looking for solutions that improve operational efficiency while intuitively supporting their human workers. This has led to advancements in the use of Autonomous Mobile Robots (AMRs) [179]. Unlike traditional automation systems that depend on installed infrastructure, these robotic systems are equipped with advanced sensors and AI algorithms, enabling them to perceive and interact with their environments. This technology allows robots to make real-time decisions and carry out complex tasks independently, ultimately improving throughput and operational effectiveness in logistics applications [197].

The ability of robots to interact with and explore their environments has become a significant focus of research and development in the robotics community. As a result, the next generation of robotic systems will integrate both locomotion and manipulation skills to interact effectively with their surroundings. This emphasis on the synergy between these two skills, known as loco-manipulation [57], has led to various innovative robotic designs. For example, humanoid robots are versatile machines that replicate human functions, utilizing two legs for movement and two arms for handling objects [191]. Additionally, quadrupedal robots, which move using four legs, have been enhanced with robotic manipulators to interact with different objects [117]. Hybrid legged-wheeled robots have also been developed, combining the agility of legs with the speed of wheels, thus broadening the capabilities of mobile manipulation systems [198].

As robotic technology continues to advance, the complexity of control mechanisms is increasing significantly. While four-wheeled robots benefit from inherent stability, the next generation of robotic systems requires more advanced control strategies. For instance, humanoid and two-wheeled robots need active control to maintain balance. Furthermore, integrating handling tasks, such as lifting or transporting objects, adds another layer of complexity. The recently introduced evoBOT (see Fig. 1.1) is a dynamic and flexible mobile robot that operates on the principles of an inverted pendulum. This robot exemplifies the challenges of modern robotics by incorporating arms that allow for simultaneous manipulation skills [99]. Designing control systems for these advanced robots becomes increasingly intricate, especially when dealing with tasks that involve high dynamics and uncertainties in real-world environments.

Learning-based control has emerged as a promising approach to address challenges in robotics. In particular, reinforcement learning (RL) has shown significant success in robot control applications, enabling robots to learn from experience in a manner similar to humans [186]. This paradigm shift allows robots to progressively improve their performance by interacting with their environments and adjusting their behaviors based on feedback [100, 82, 182]. Unlike model-based controllers that depend on predefined rules, learning-based methods empower robots to adapt autonomously to new tasks and environments. This adaptability is crucial for applications that involve uncertainties and unmodeled effects of the real world, especially in dynamic settings. Robots trained using RL have demonstrated resilience across various challenges, including dexterous manipulation [146], robust navigation [77], and dynamic locomotion [104] tasks.

1.2. Objective and Contributions

This thesis aims to advance the field of robot learning by accelerating the training process and enhancing performance for real-world robotic applications. This section will present the three primary research questions associated with this objective and the scientific contributions of this thesis.

Guided Reinforcement Learning

RL offers a promising approach to learning-based control in robotics, enabling agents to acquire skills like human learning through trial-and-error interactions with their environments. However, challenges such as training speed (efficiency), policy performance (effectiveness), and the transfer of learned behaviors from simulations to reality (sim-to-real) continue to pose obstacles to broader implementation.
To address these issues, guiding the data-driven training process with external knowledge, such as expert demonstrations or previous experiences, has emerged as a viable solution. This approach facilitates more efficient and effective training for real-world scenarios. Research contributions to hybrid learning for Embodied AI often come from various disciplines, highlighting the need for a structured overview. Therefore, the first research question relevant to this thesis can be formulated as follows:

> **Research Question I:** How can external knowledge sources be systematically incorporated into RL training to enhance efficiency, effectiveness, and sim-to-real transfer?

This thesis explores the field of robot learning through the lens of the research question regarding Guided RL. Key contributions associated with this research question include:

- **Contribution I.A:** A novel taxonomy for Guided RL serves as a modular toolbox that integrates various knowledge sources into the RL training pipeline.
- **Contribution I.B:** A comprehensive survey of Guided RL methods, focusing on problem representations, learning strategies, task structures, and sim-to-real.
- **Contribution I.C:** Practical guidelines to select Guided RL methods for typical robotics applications, including locomotion, manipulation, and navigation tasks.

For more detailed information about the concept of Guided RL and the associated contributions, please refer to Chapter 3.

Case Study: Learning-based Control for evoBOT

Building upon the foundations of Guided RL, this thesis aims to develop learning-based control methods for a dynamic and challenging real-world robotic system. The two-wheeled robot evoBOT, developed by the Fraunhofer Institute for Material Flow and Logistics IML, is selected as a case study platform for this research (see Fig. 1.1). With a weight of 50 kg and a maximum arm reach of 1.5 m, evoBOT can handle many objects, making it suitable for various applications.

Figure 1.1.: **evoBOT** developed by Fraunhofer IML. A highly dynamic and flexible two-wheeled robot used as case study for learning-based control of challenging systems within this thesis [62].

evoBOT is a highly flexible mobile robot that can reach up to 10 m/s, representing a new class of robotic systems operating on the principle of an inverted pendulum. Therefore, the findings from this research contribute to the broader field of robotics, enhancing the understanding of how learning-based approaches can be effectively applied to dynamic and versatile robotic systems.
The learning-based controller is supposed to fulfill the following requirements:

- Training within a few minutes from scratch for fast iteration speed (Efficiency).
- Save training in simulation to prevent damage to the real robot (Efficiency).
- Fast control reaction times utilizing the dynamic limits of the robot (Effectiveness).
- Robustness against both internal and external disturbances (Effectiveness).
- Policies should be transferable from simulation to the real robot (Sim-to-Real).
- Seamless integration of trained policies into the robot's firmware (Sim-to-Real).

Consequently, the most promising methods from the Guided RL taxonomy are selected to enhance the efficiency and effectiveness of the real-world robotic application. In particular, the second research question explored in this thesis is as follows:

Research Question II: How can Guided RL methods improve the efficiency and effectiveness of learning-based control for the two-wheeled mobile robot evoBOT?

This thesis thus also contributes to the technical advancements and insights necessary for training two-wheeled mobile robots, such as evoBOT, in complex and dynamic tasks. The key contributions related to this research question are as follows:

- **Contribution II.A:** A robust RL framework designed for highly dynamic motions, including robust balancing and whole-body locomotion with object handling.
- **Contribution II.B:** A reference study aimed at minimizing the sim-to-real gap for highly dynamic two-wheeled robots, using real-world measurements.
- **Contribution II.C:** An initial pipeline to deploy policies trained in simulation onto the physical robot in real-time critical control applications.

For detailed technical information regarding implementing the learning-based controller in the case study, please refer to Chapter 4.

Generalized Action Space for Robot Control

The case study demonstrates that highly dynamic tasks in training are significantly influenced by the control commands that a neural network can direct to a real robot, commonly referred to as the action space in RL. Typically, robots operate within an action space defined by physics. For instance, electric actuators utilize current to track torque through a high-frequency control loop. While RL policies can directly output torque values, many research papers use alternative action spaces such as joint position, joint velocity, or task-space setpoints. These values are then converted into torques using feedback laws. However, practitioners often rely on intuition when selecting action spaces in RL. Therefore, it would be beneficial for the community to have a method for identifying the most efficient and effective action space for their robotic tasks. Finally, the third research question addressed in this thesis is as follows:

Research Question III: How does action space design affect the performance of learning-based control strategies across various robot morphologies and learning tasks?

This thesis also contributes to developing and evaluating a novel action space for robot control. Specifically, the contributions include:

- **Contribution III.A:** A formulation of a generalized action space for configuration-based robot learning that operates directly at the torque level.
- **Contribution III.B:** Benchmarking the impact of action spaces on training speed (efficiency) and policy performance (effectiveness) across multiple robot platforms.
- **Contribution III.C:** Practical guidelines for action space selection, considering robot dynamics, parameter selection, control frequencies, and sim-to-real transfer.

For technical details about the generalized action space and its application to various robotic platforms, please refer to Chapter 5.

1.3. Outline of the Thesis

Based on these contributions, this thesis consists of seven chapters (see Fig. 1.2). Below is a summary of each chapter to facilitate easy navigation through the document.

Chapter 1 (Introduction) motivates the problem and outlines the overall objective and specific contributions of this thesis. It also provides an overview of the thesis structure.

Chapter 2 (Related Work) summarizes the current state of learning-based control, highlighting the research gap and unique aspects of the approach.

Chapter 3 (Guided RL) investigates the integration of additional knowledge into the RL pipeline to improve training efficiency, performance, and sim-to-real transfer.

Chapter 4 (Case Study) demonstrates the application of Guided RL to improve the efficiency and effectiveness of learning-based control for the two-wheeled robot evoBOT.

Chapter 5 (Action Space) proposes a generalized parametrization of action spaces and benchmarks its impact on a diverse set of robot platforms and tasks.

Chapter 6 (Conclusion) summarizes the contributions and findings of the thesis and provides recommendations for future research.

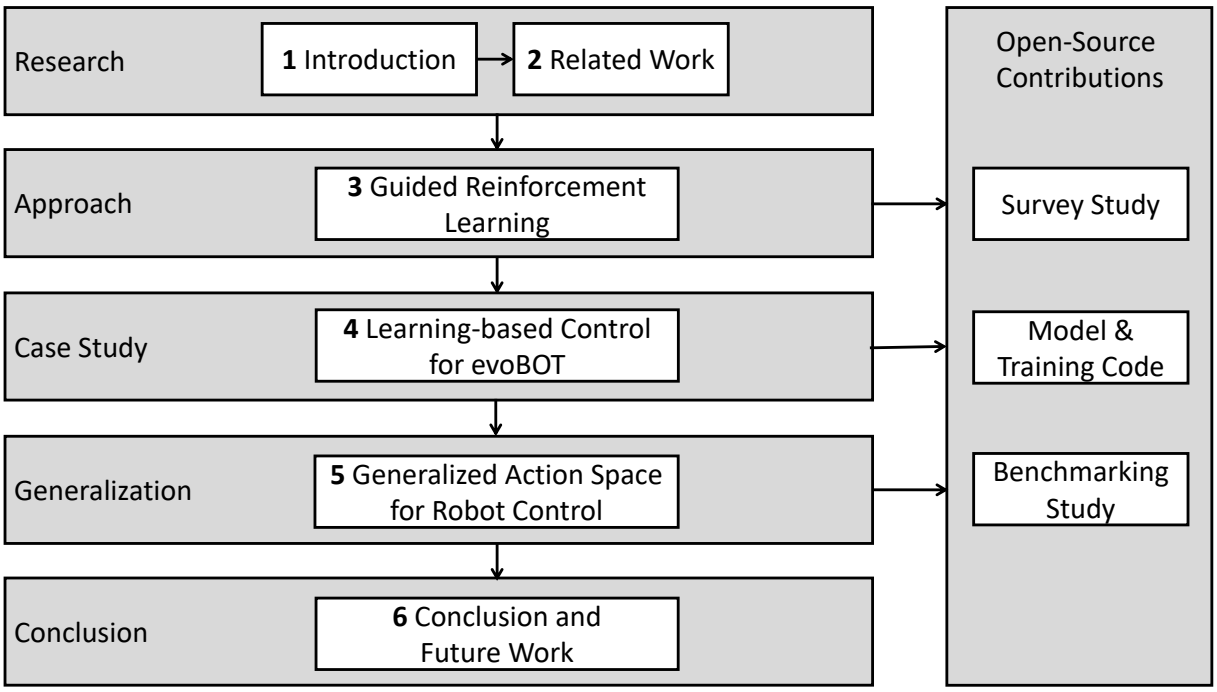


Figure 1.2.: **Overview of the thesis structure**. The interconnection of chapters is highlighted along with the related open-source contributions.

2

Related Work on Robot Learning

This chapter provides a comprehensive overview of the latest advancements in RL for robot control. It begins with a mathematical formalization of the RL problem, followed by an explanation of neural networks and algorithms. The discussion then explores the concept of inductive bias in RL, emphasizing its importance in tackling challenges related to robot control. The section on RL for dynamic motions examines the application of RL to various complex tasks and platforms, underscoring the necessity of robust training under sim-to-real conditions. Additionally, the discussion on action space design in RL highlights how the choice of action spaces can affect learning efficiency and overall performance. Finally, the current state of the art in the field of robot learning is summarized.

2.1. Reinforcement Learning (RL)

Introduction to RL in Robotics

RL is a subfield of ML that focuses on how agents can learn to make decisions by interacting with an environment. It represents a promising methodology for addressing decision-making challenges by emulating human behavior through trial-and-error interactions. In recent years, RL has exhibited remarkable advancements across a diverse array of complex tasks, including traditional strategy games, real-time computer games [147], and applications within the field of robotics [7]. It has effectively been employed to tackle continuous control problems [115], encompassing dynamic locomotion [104, 168, 180], robot navigation [77, 32, 88], and dexterous manipulation [146, 24, 86]. These accomplishments are predicated upon the data-driven nature of the approach, facilitating extensive exploration of the solution space.

Markov Decision Process

Mathematically, at the core of RL is the Markov Decision Process (MDP), which provides a formal framework for modeling decision-making scenarios [186]. In RL, an agent observes the current state of the environment, selects actions based on a policy, and receives rewards as feedback for its actions. The primary goal of the agent is to maximize the cumulative reward over time (see Fig. 2.1).

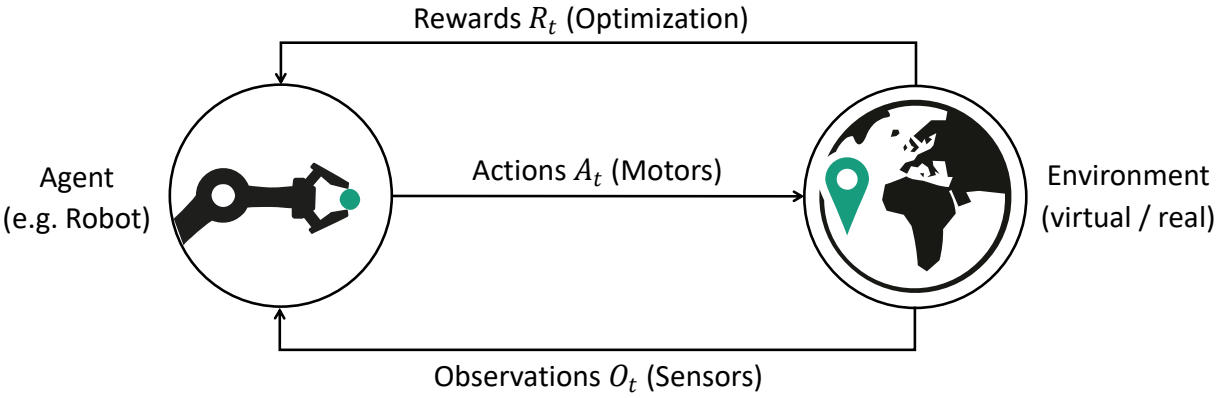


Figure 2.1.: **Overview of RL** based on Markov Decision Processes in the context of robotics. Sensory observations and related rewards are processed by the agent to generate a set of actions.

An MDP is characterized by a set of states S, commonly referred to as observations O in robot learning, a set of actions A, a transition function $P(s'|s,a)$, a reward function $R(s,a)$, and a discount factor $\gamma \in [0,1)$.
A policy $\pi(a|s)$ determines the probability of taking action a in state s. The agent aims to learn an optimal policy that maximizes the cumulative expected reward over time.
To evaluate the effectiveness of a policy, we define the State Value Function $V(s)$ as:

$$V(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s\right] \tag{2.1.1}$$

This function represents the expected return starting from state s and following the policy π. The Action Value Function $Q(s,a)$ is defined as:

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a\right] \tag{2.1.2}$$

This function estimates the expected return for taking action a in state s and then following the policy π. To relate these value functions, we use the Bellman Equations. For the value function, we have:

$$V(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a) \left(R(s,a) + \gamma V(s')\right) \tag{2.1.3}$$

This equation decomposes the value of a state into the expected rewards from taking actions and transitioning to subsequent states. The action value function is expressed as:

$$Q(s,a) = \sum_{s'} P(s'|s,a) \left(R(s,a) + \gamma \sum_{a'} \pi(a'|s') Q(s',a')\right) \tag{2.1.4}$$

Neural Networks

In the field of robotics, deep learning techniques are frequently used to approximate value functions in RL [7, 113]. These neural networks are capable of processing high-dimensional inputs, which makes them well-suited for environments represented by complex data.

These approaches, also known as Deep Reinforcement Learning (DRL), will be referred to as RL in the remainder of this thesis. The neural network approximating the action value function $Q(s, a)$ is trained using a loss function defined as:

$$L(\theta) = \mathbb{E}\left[(y - Q(s, a; \theta))^2\right] \tag{2.1.5}$$

where y is the target value defined as:

$$y = R(s, a) + \gamma \max_{a'} Q(s', a'; \theta^-) \tag{2.1.6}$$

Here, θ are the parameters of the neural network, and θ^- represents the parameters of a target network, which stabilizes learning by providing consistent targets.
A practical example of RL can be seen in the game of Go [181], where the agent learns to play against itself or human players. The state s can be represented by the board configuration, while the actions a correspond to placing a stone in one of the empty intersections. The reward function $R(s, a)$ is typically defined as +1 for winning, -1 for losing, and 0 for a draw or non-terminal state.
The agent uses a neural network to approximate the action value function $Q(s, a)$, learning from numerous games played. Over time, through trial and error, the agent refines its policy π, becoming increasingly adept at selecting moves that lead to victory. This combination of deep learning and RL enables the agent to develop complex strategies, surpassing human-level performance in the game.

Proximal Policy Optimization

One popular algorithm in the RL landscape is Proximal Policy Optimization (PPO) [175]. PPO is designed to stabilize training and improve performance through a clipped objective function. The objective function for PPO is given by:

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t, \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{2.1.7}$$

Here, $\hat{A}_t$ is the estimated advantage at time t, and ϵ is a hyperparameter that controls the clipping range. The objective encourages the new policy π_θ to stay close to the old policy $\pi_{\theta_{\text{old}}}$, thus preventing extensive updates that could destabilize training.
PPO operates on a trust region approach, where the clipped objective effectively constrains the policy's updates, ensuring that changes are not too aggressive. This mechanism allows for a good balance between exploration and exploitation, making it robust to hyperparameter choices.
The algorithm has been shown to perform well across various tasks [81, 93]. Consequently, it is widely used in the robot learning community due to its balance between ease of implementation and effective performance. Its versatility and stability have made it a preferred choice for many applications, including games, robotics, and continuous control tasks. The ability of the algorithm to leverage both on-policy and off-policy data further enhances its efficiency in learning optimal policies.

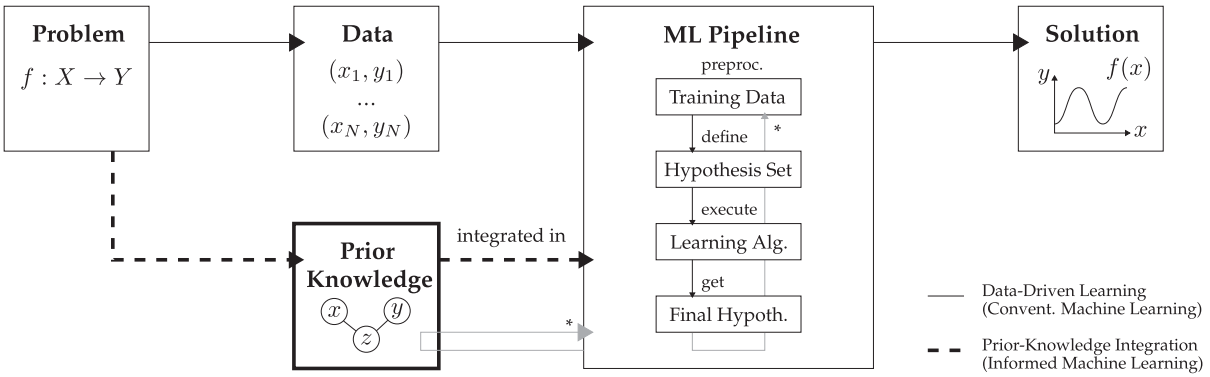


Figure 2.2.: **Informed machine learning** combines data-driven and knowledge-driven strategies by integrating additional knowledge into the training process [169].

2.2. Inductive Bias in RL

Challenges in Reinforcement Learning for Robotic Control

Recent accomplishments in the field of RL are predicated upon the data-driven nature of the approach, facilitating extensive exploration of the solution space. Nevertheless, acquiring control policies through such methods necessitates numerous interactions with the environment, thereby underscoring the significance of collecting high-quality samples and efficiently exploring the search space. While the prospect of direct learning on real robots is enticing, it is accompanied by substantial challenges, such as elevated sample costs, partial observability, and safety constraints [41]. Consequently, simulators are frequently employed as scalable training environments, alleviating safety concerns associated with real-world scenarios. Although training within the simulation is expedited, more economical, and inherently safer, the deployment of these policies onto physical robots may fail due to discrepancies between the simulated and actual environments, a phenomenon referred to as the sim-to-real gap [220].

Hybrid Approaches as a Promising Solution

Combining data-driven and knowledge-driven strategies may effectively resolve these challenges. In this direction, the concept of Triangular AI integrates three key dimensions: data, knowledge, and context [193]. Current AI systems often overlook these critical elements essential for reliable and trustworthy decision-making in dynamic environments. By incorporating knowledge from various contexts, informed simulations, and explicit human input, Triangular AI can offer a promising framework.
In this context, hybrid strategies, such as those proposed by Von Rueden et al. [169], exemplify the integration of knowledge into learning systems. Their abstract framework for informed machine learning considers the sources of knowledge, their representations, and their integration into the ML pipeline (see Fig. 2.2). This approach highlights the potential for combining physical laws, human expertise, and domain-specific knowledge to improve sample efficiency and policy robustness. Building upon this foundation, hybrid approaches may offer valuable avenues for advancing RL in real-world robotics applications.

Efficient and Effective Reinforcement Learning

In the context of robotics, numerous research avenues have emerged to enhance the efficiency of exploration and effectively deploy policies for real-world systems. For instance, specialized algorithms have been developed to augment sample efficiency [6, 14, 173]. Demonstration data has been utilized to expedite RL methodologies [24, 203, 108]. Thoughtfully selecting task-specific state representations, reward functions, and action spaces can significantly improve both convergence time and overall performance [131, 180, 128]. Additionally, RL approaches can be synergistically combined with classical control to facilitate learning within lower-complexity state spaces [40, 208]. Finally, the integration of knowledge regarding the structure of the learning task has been shown to enhance performance and accelerate convergence [143, 214]. The rich variety of approaches from diverse disciplines complicates a comprehensive understanding of the current state-of-the-art learning control policies for real-world robotics, thereby underscoring the need for a structured overview.

Related Survey Studies

Recent surveys provide partial insights into this field. For example, [217] emphasizes strategies to improve sample efficiency in RL in a general context, while [100] focuses specifically on its applications in the robotics domain. Von Rueden et al. [170] analyze how ML and simulation can be harmoniously combined into a hybrid modeling approach. Another survey [220] places special emphasis on sim-to-real transfer methods for robotics. Dulac-Arnold et al. [41] delineate unique challenges associated with real-world RL. Lastly, a recent case study [82] offers valuable practical insights for deploying policies in real-world scenarios. This research complements the above-mentioned work by providing a systematic overview of integrating knowledge into the RL pipeline to enhance efficiency and effectiveness in real-world robotic applications.

2.3. RL for Dynamic Motions

Learning Dynamic Locomotion

RL has been employed to achieve dynamic locomotion in various robotic platforms. In the field of humanoid robotics, RL is used for tasks such as learning omnidirectional movement [166], which demonstrates the capability of humanoid robots to navigate in multiple directions fluidly. Additionally, RL has been applied to determining foot placements [40], enabling robots to optimize their gait for stability and efficiency. Ensuring robust locomotion on diverse indoor and outdoor terrains is another critical application of RL [161], where the focus is on adapting locomotion strategies based on varying environmental conditions. For quadruped robots, RL has been applied to navigate increasingly complex terrains [168], showcasing their ability to traverse challenging environments while maintaining stability. Furthermore, RL has been utilized to mimic the agile movement skills of animals [155], allowing quadrupeds to perform complex maneuvers that enhance their mobility. Developing safe recovery strategies [213] is also a key focus, ensuring that robots can regain balance and stability after disturbances.

(a) Dynamic Locomotion [127] (b) Robust Balancing [198] (c) Loco-Manipulation [67]

Figure 2.3.: **Dynamic robot learning tasks** recently been trained with RL.

RL also facilitates rapid motor adaptations [104], enabling quadrupeds to adjust their locomotion dynamics in real-time based on sensory feedback. Additionally, research focused on RL-based locomotion has aimed to push the physical limits of robots, including achieving high velocities [127], demonstrating the potential for faster and more efficient movement. Traversing extreme parkour environments [31] further illustrates the versatility of RL in training robots to perform complex and dynamic tasks.

Learning Robust Balancing

As two-wheeled robots become increasingly important in research and industrial applications (see Fig. 2.4), developing robust balancing strategies is crucial for their effective operation. Two-wheeled robots, also known as two-wheeled inverted pendulums (TWIP), present more significant challenges in control than wheeled robots, quadrupeds, and humanoids, which have inherent stability due to their multiple points of contact with the ground. RL has been employed to improve control strategies for these platforms to address this issue. Research in this area has primarily focused on the learning control of inverted pendulums using RL, but most studies have dealt with simplified models [149, 116, 38]. Initial investigations have also looked into the control of hybrid legged-wheeled robots. For example, Vollenweider et al. demonstrate how to learn multiple advanced skills simultaneously for a quadruped-humanoid transformer, including balancing, standing up, and navigating obstacles [198]. Baltes et al. propose an RL algorithm for the dynamic control of a two-wheeled scooter integrated with a humanoid robot, showing robustness against external disturbances. Additionally, Baek et al. explore a hybrid approach that combines RL with model-based techniques to control a wheeled humanoid robot [9].

Learning Loco-Manipulation

Recent research and development in robotics have increasingly focused on how robots can both explore and interact with their environments simultaneously. Consequently, new tasks for robot learning have been designed to combine locomotion and manipulation skills to enhance engagement with surroundings. This emphasis on integrating these capabilities, called loco-manipulation [57], has initiated research into learning-based control methods.

(a) Handle [23] (b) Ascento [97] (c) Wheeled-Humanoid [9]

Figure 2.4.: **Two-wheeled robots** have recently been introduced in research and industry.

RL has been successfully applied to general-purpose humanoid robots, which utilize two legs for movement and two arms for handling objects [37]. Additionally, quadrupedal robots operate using four legs and are equipped with robotic manipulators that enable them to interact with a variety of objects [67, 183]. Early research is also investigating the use of RL for loco-manipulation in hybrid legged-wheeled robots. These robots combine the agility of legs with the speed of wheels, thereby enhancing the capabilities of mobile manipulation systems [176].

RL with High Control Frequencies

Many approaches discussed concerning RL in robotic locomotion tasks focus on quasi-static systems. While the low-level control of these systems typically operates at frequencies between 500 Hz and 2 kHz, the control policies developed are often trained and executed at rates of 50 Hz or lower [40, 166, 168, 155, 105]. Even for the high-speed running capabilities of these robots, inference rates up to 100 Hz prove effective for maintaining their dynamic stability [126, 18]. However, learning dynamic movements in highly unstable systems necessitates higher control frequencies, which have already been explored, such as decentralized control for aggressive drone maneuvers [17].

Physics Simulators for Robot Learning

Simulators are often used as scalable training environments in robot learning, helping to mitigate safety concerns associated with real-world scenarios. As interest in robotics simulation continues to grow, the variety of available physics simulators has significantly expanded [34]. However, selecting the appropriate simulator can be challenging due to the rapidly evolving technologies in this field. Standard physics simulators used in robotics include Gazebo [101], developed by the Open Robotics Foundation, Bullet [35], and MuJoCo [190], which is known for its precise contact models. For the evoBOT simulation model, the Graphics Processing Unit (GPU)-based NVIDIA Isaac SIM framework is chosen as the simulation platform [192]. This framework integrates advanced RL tools, such as Isaac Gym [123], Omniverse Isaac Gym [74], and Isaac Lab [133], which enable GPU-accelerated training.

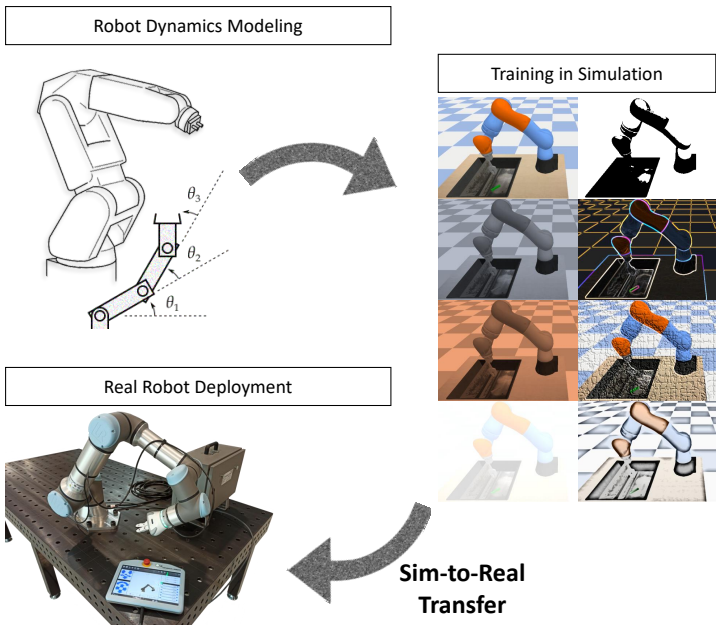


Figure 2.5.: **Workflow of sim-to-real transfer** in RL for robotics. It involves robot dynamics modeling, training RL policies in simulation, and deploying them to the real robot [220].

Sim-to-Real Transfer

For transferring trained policies from simulation to real robots, domain randomization introduces significant variability in the simulation environment [220], initially focusing on randomizing visual elements [189]. Instead of altering visual aspects, Peng et al. [153] propose dynamics randomization, which emphasizes variations in parameters related to the robot and its environment, such as link weights, joint damping, and friction coefficients. Similar visual and dynamic randomization concepts have been utilized in other studies, where disturbances are introduced to create more resilient agents. For instance, Rodriguez et al. [166] incorporate noisy sensor inputs, while Rudin et al. [168] apply random external forces to enhance policy deployment in real-world conditions. Additionally, methodologies like multi-modal delay randomization [83] have been suggested to ensure that deployed policies are robust against time delays.

2.4. Action Space Design in RL

Action Spaces for Robot Control

Choosing the appropriate action space for robot learning often relies on intuition. For instance, a wheeled robot may use a wheel velocity action space, while a legged robot typically employs joint positions [60]. Robots operate within an action space defined by physical principles. For example, electric actuators utilize current input, which is regulated to achieve a target torque through high-frequency control [21]. While it is feasible with RL to generate torques as actions [115, 174] directly, many studies opt for alternative action spaces such as joint position [109, 188, 127], joint velocity [209, 99], or task space [128] setpoints that converted into torques using a state-dependent feedback mechanism.

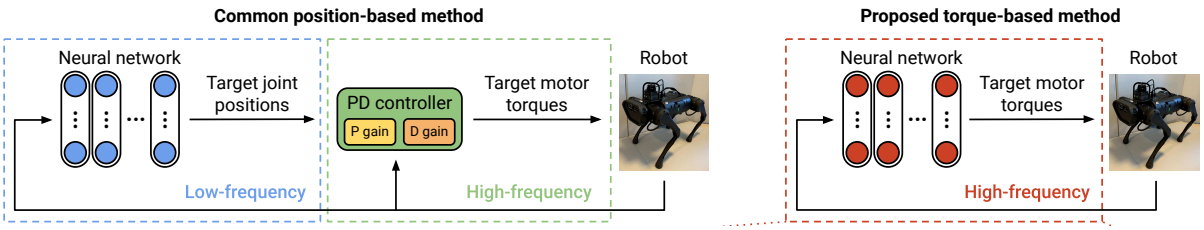


Figure 2.6.: **Action space design** and its implications for robot learning tasks [28].

Studies on Action Space Design

Numerous studies have explored how the selection of action space influences robot learning for character animation [157, 165], manipulation [196, 3], and aerial robots [94]. Research focusing on simulated characters [157, 165] and manipulation tasks [196] often highlights the advantages of position control in these areas.

Conversely, Aljalbout et al. found that joint velocity is preferable for sim-to-real transfer in manipulation tasks. Moreover, Kaufmann, Bauersfeld, and Scaramuzza determined that a Proportional-Derivative (PD) control feedback scheme is optimal for quadrotor flight. Despite these findings favoring position control, the reality of action space implementation in contemporary RL frameworks is complex. For instance, the OmniIsaacGymEnvs task suite [123] features environments balanced between position, delta position (velocity), and torque action spaces, illustrating their varied dynamics.

Additionally, various studies have examined alternative design considerations for action spaces in robotics, such as configuration-space versus end-effector-space control [128, 36, 144, 70], or whether to maintain a continuous action space or discretize it [177]. These avenues show promise for enhancing learning when additional insights into the robot's dynamics are utilized. In our research, we limit our focus to configuration-space action spaces, as joint space control is prevalent in sim-to-real RL and involves design decisions that can significantly influence learning.

Temporal Aspects in Action Space Design

Choosing an action space introduces an inductive bias in RL algorithms, as noted by Reda, Tao, and Van De Panne. This design choice can limit or modify the policy's hypothesis space before learning begins, which can also be viewed as a form of environment shaping or policy initialization [55]. Each action space choice comes with parameters that can affect learning.

Control frequency is one such parameter; lower frequencies may restrict controller bandwidth but can simplify learning by shortening trajectory lengths, resulting in smoother performance [71]. For action spaces that track setpoints, the selection of gains can influence effectiveness and sim-to-real transfer [211, 184, 180]. Other considerations include action filtering [140] for smoother movements and action chunking [219] to facilitate learning in long-horizon tasks.

2.5. Concluding Remarks

This chapter provided a comprehensive overview of recent advancements in RL for robot control. Most research in this field currently focuses on static or quasi-static robotic systems, such as manipulators and quadrupeds. At the same time, fewer studies address dynamically unstable systems like humanoids and two-wheeled robots. In contrast, this thesis develops methods for the dynamically unstable two-wheeled robot, evoBOT, where precise real-time control is crucial for maintaining balance and preventing falls.
Many existing methods are tested through experiments involving tasks with limited dynamics, such as slow grasps or low-velocity locomotion. These tasks result in simpler MDP formulations due to minimal state transitions. This work, however, emphasizes highly dynamic tasks, including high-velocity locomotion and rapid arm movements, where robots often operate at their actuator limits. To explore actual dynamic limits, the study evaluates the impact of an accurate simulation model with optimized dynamic parameters, such as compliant contacts for wheels and actuator models based on real-world data.
Additionally, numerous publications primarily present simulation-only experiments, which overlook the real-world challenges of policy transfer, including limited data, sensor noise, latency, and actuator dynamics. This work directly deploys policies trained in simulation onto real-time hardware, facilitating a seamless connection between neural networks, sensors, and actuators. These tasks notably benefit from high control frequencies and torque action spaces, allowing complete control over the robot's dynamics.

3

Guided Reinforcement Learning

RL is a promising approach for learning-based control in robotics. Agents develop skills similar to human learning by interacting with their environment through trial and error. However, as highlighted in the current literature (see Chapter 2), several challenges still hinder wider adoption. These include issues related to training speed (efficiency), policy performance (effectiveness), and the transfer of learned behaviors from simulations to real-world situations (sim-to-real). Utilizing external knowledge, such as expert demonstrations or prior experiences, has become a practical solution to address these challenges. This approach can enhance the efficiency and effectiveness of training for real-world applications. Research contributions in hybrid learning for Embodied AI often come from various fields and would benefit from a structured overview on integrating knowledge into the RL pipeline for robotics applications. Therefore, the underlying research question to be covered in this chapter is:

Research Question I: How can external knowledge sources be systematically incorporated into RL training to enhance efficiency, effectiveness, and sim-to-real transfer?

To overcome this challenge, this chapter introduces Guided RL, a modular framework to accelerate training and improve performance in real-world robotics applications. It begins with a conceptual overview, outlining the overall pipeline for incorporating knowledge into RL and providing essential definitions (Section 3.1). Next, a novel taxonomy is introduced to categorize various Guided RL approaches, illustrating how different sources of knowledge can be integrated into the learning process (Section 3.2). The chapter then surveys existing methods in the field of Guided RL and provides comprehensive descriptions of different strategies to integrate knowledge into the RL pipeline (Section 3.3). Following up, an evaluation study quantitatively assesses the effects of Guided RL methods on efficiency, effectiveness, and sim-to-real transfer (Section 3.4). Finally, the chapter provides practical guidelines for selecting Guided RL methods for typical robotics applications (Section 3.5).

Disclaimer: Parts of this chapter are based on a prior publication of this thesis [55].

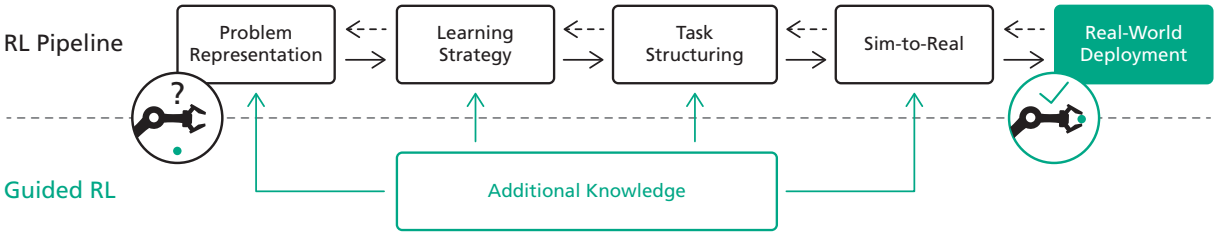


Figure 3.1.: **Pipeline of Guided Reinforcement Learning** (based on prior publication [55]). Additional knowledge can be integrated at all levels of the RL pipeline to accelerate the training process and improve the performance for real-world robotic settings.

3.1. Conceptual Overview

This section introduces the concept of Guided RL, outlining its definition, objectives for effective and efficient deployment in real-world robotics, and its connections to related research fields.

3.1.1. Idea & Pipeline

Guided RL refers to incorporating external knowledge into the learning process to enhance the speed and success of robotics applications in real-world scenarios. As illustrated in Fig. 3.1, this knowledge can be integrated at various stages of the RL pipeline: problem representation, learning strategy, task structuring, or sim-to-real transfer methods. Refer to Section 3.2.3 for an in-depth pipeline discussion.

3.1.2. Efficient & Effective Learning

The central goal of Guided RL is to foster success in real-world robotics deployment by ensuring learning occurs in both an *efficient* and *effective* manner, as shown in Fig. 3.2. Based on commonly used metrics in the literature [128, 45, 168, 77, 158, 166, 86], the following definitions are adopted:

Definition 3.1.1 (Efficiency)**:** *A training process is deemed more efficient if it requires fewer interactions with the environment or less time to reach convergence compared to the baseline.*

Definition 3.1.2 (Effectiveness)**:** *A training process is considered more effective if the policy's performance, in terms of total return or success rate, exceeds the baseline.*

Definition 3.1.3 (Sim-to-Real)**:** *A training process is classified as sim-to-real if it utilizes simulation for training policies or evaluating methods before real-world deployment.*

While efficient and effective policy training and real-world robotics deployment are natural dimensions of Guided RL, integrating these three aspects serves as the primary motivation. For a comprehensive evaluation of existing approaches in this area (see Section 3.4), the following term is introduced:

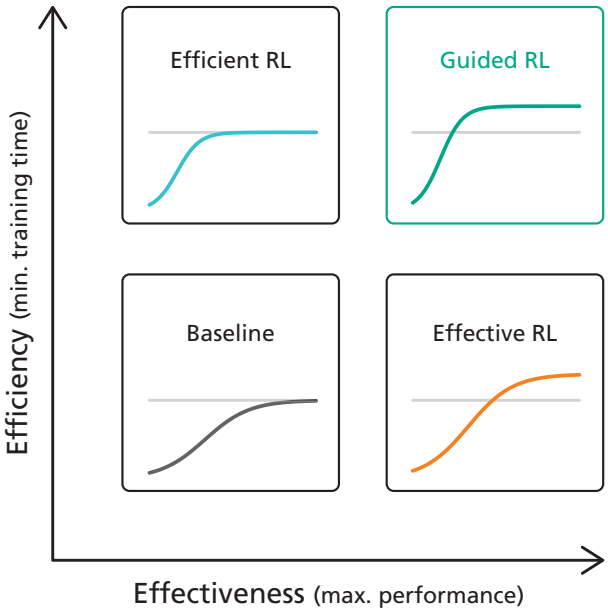


Figure 3.2.: **Objective of Guided Reinforcement Learning** (based on prior publication [55]). Guided RL aims to accelerate training and optimize performance for real world robotic systems.

Definition 3.1.4 (Guided RL Compliance)**:** *A training process is regarded as fully Guided RL compliant when improvements are observed across all three dimensions: efficiency, effectiveness, and sim-to-real.*

3.1.3. Related Research Areas

This study centers on Guided RL, which directly incorporates prior knowledge into the learning pipeline to enhance success in real-world robotics applications. Thus, this review occupies the intersections of deep RL, robotics, and simulation.

Several related research areas are noted but are *not* explicitly addressed in this study. For instance, using model-based or off-policy algorithms has been shown to improve sample efficiency relative to on-policy algorithms [82]. Additionally, fine-tuned hyperparameters in learning algorithms can enhance overall policy performance [79]. Moreover, multi-task learning, which involves training multiple tasks simultaneously, can lead to more efficient training [90]. Similarly, meta-learning research aims to solve previously unseen tasks quickly and efficiently [64].

However, the methods identified in the realm of Guided RL remain agnostic regarding the choice of algorithm (e.g., [175, 75, 115, 135, 45]) and learning tasks such as locomotion, navigation, or manipulation.

This study does not aim to establish a strict distinction between efficient, effective, and Guided RL. Instead, its primary motivation is to review existing approaches and provide a structured overview of recent research directions that may strengthen the relationship between the RL and robotics communities.

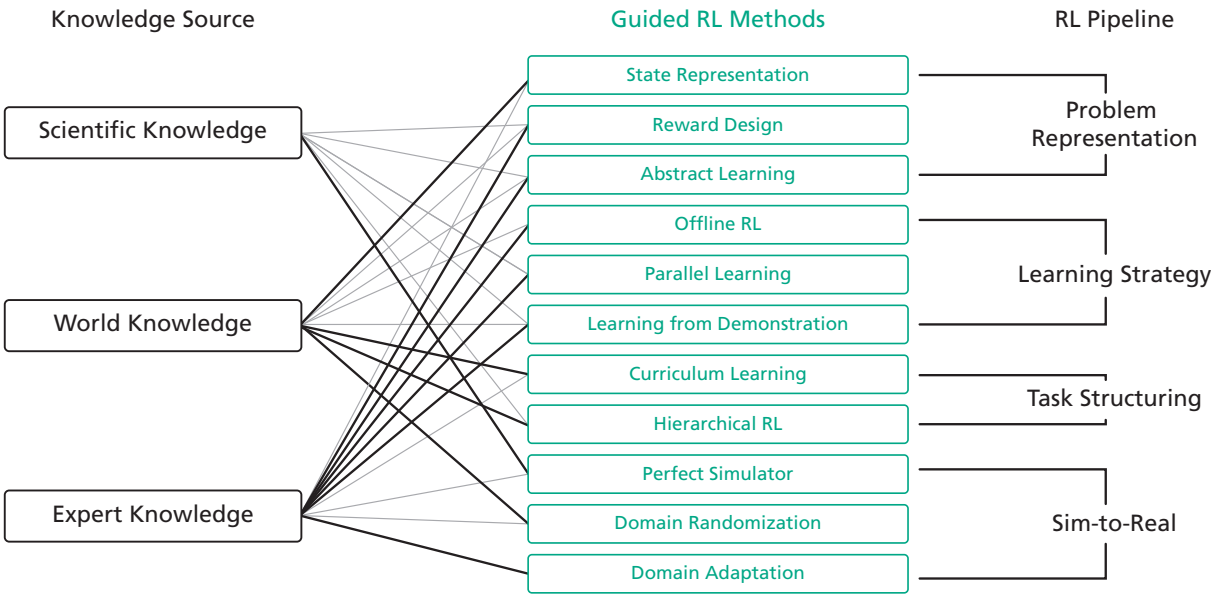


Figure 3.3.: **Taxonomy of Guided Reinforcement Learning** (based on prior publication [55]). The taxonomy organizes Guided RL research based on the underlying knowledge source, specific methodological approaches, and their integration into the learning pipeline. The thickness of the connecting lines indicates the relevance of knowledge sources. This taxonomy aims to provide a structured overview for categorizing recent research activities (see Table 3.1).

3.2. Guided RL Taxonomy

This section presents a taxonomy for Guided RL (Fig. 3.3). Based on the concept of informed machine learning [169], the taxonomy is organized according to the *knowledge source*, *methodical representation*, and *integration into the pipeline*. In the following, central components of the taxonomy are introduced on a conceptual level, while a comprehensive categorization of approaches will be detailed in Section 3.3.

3.2.1. Knowledge Source

There are three main types of prior knowledge that support most Guided RL methods. These can be categorized into scientific knowledge, world knowledge, and expert knowledge. According to [169], these sources of knowledge range from formal and structured to intuitive and experiential. A brief description of each category is provided below.

Scientific Knowledge

Scientific knowledge is formalized and derived from the domains of physics, biology, or engineering. It can be validated through experiments or empirical analysis. This knowledge aids in developing realistic simulators and enhances the integration of biological insights into the learning process.

World Knowledge

World knowledge can be either formalized or intuitive and includes facts from everyday life, making it accessible to a wide range of people. In the context of Guided RL, world knowledge can help design intuitive observation and action spaces and guide the natural structuring of the learning tasks.

Expert Knowledge

Expert knowledge is typically held by a specific group of experienced professionals with strong ties to the fields of robotics and RL. This knowledge is often informal and can significantly influence engineering design decisions. For example, it may be incorporated when defining an RL problem or developing an overall learning strategy.

3.2.2. Guided RL Methods

This category is the core of the taxonomy (see Fig. 3.1), connecting to the RL pipeline and related robotic applications. An overview of these methods is given here, along with detailed descriptions of the most common approaches found in Section 3.3.

- **State Representation** involves transforming or extending the observable state into more informative representations for the model (see Section 3.3.1).
- **Reward Design** encompasses techniques for inducing knowledge by creating suitable dense reward functions or automatic learning methods (see Section 3.3.2).
- **Abstract Learning** involves selecting a task-specific action space for a robotics problem, which may integrate model-based approaches (see Section 3.3.3).
- **Offline RL** focuses on utilizing offline data to learn policies from recorded training sets efficiently (see Section 3.3.4).
- **Parallel Learning** concerns the parallelization of algorithmic components while maintaining scalability and robustness in the learning process (see Section 3.3.5).
- **Learning from Demonstration**, also known as imitation learning, leverages example trajectories to inform the trained policy (see Section 3.3.6).
- **Curriculum Learning** is based on gradually solving more straightforward tasks with increasing difficulty in structuring complex tasks (see Section 3.3.7).
- **Hierarchical RL** utilizes the hierarchical structure of the learning task to address different subtasks or deploy high- and low-level policies (see Section 3.3.8).
- **Perfect Simulator** seeks to build more realistic simulations with accurate robot models, physics computation, and environment representation (see Section 3.3.9).
- **Domain Randomization** aims to enhance policy robustness by randomizing simulation parameters, such as visual or dynamic properties (see Section 3.3.10).

- **Domain Adaptation** involves conditioning an adaptation module to transition observations between the simulated and real worlds, or vice versa (see Section 3.3.11).

3.2.3. Knowledge Integration

An extensive literature review indicates that a practical RL pipeline for real-world robotics can be structured into four components: problem representation, learning strategy, task structuring, and sim-to-real methods (see Fig. 3.1). Each iterative step in this pipeline can incorporate additional knowledge through Guided RL methods.

Problem Representation

Representing a real-world robotics problem in a formal RL framework requires substantial knowledge. Key challenges include selecting appropriate observations, defining a reward function, and specifying the agent's action space for the desired learning task. Additionally, choosing suitable training data necessitates careful consideration of real-world versus synthetic data.

Learning Strategy

Incorporating expert knowledge into the learning strategy can involve employing parallel learning architectures for specific problems, framing the issue online or offline, or utilizing real or synthetic demonstration samples.

Task Structuring

Depending on the complexity of the robotics problem, further knowledge can be integrated to structure the learning task meaningfully. For example, complex tasks may be learned sequentially with increasing difficulty or decomposed into several subtasks.

Sim-to-Real Methods

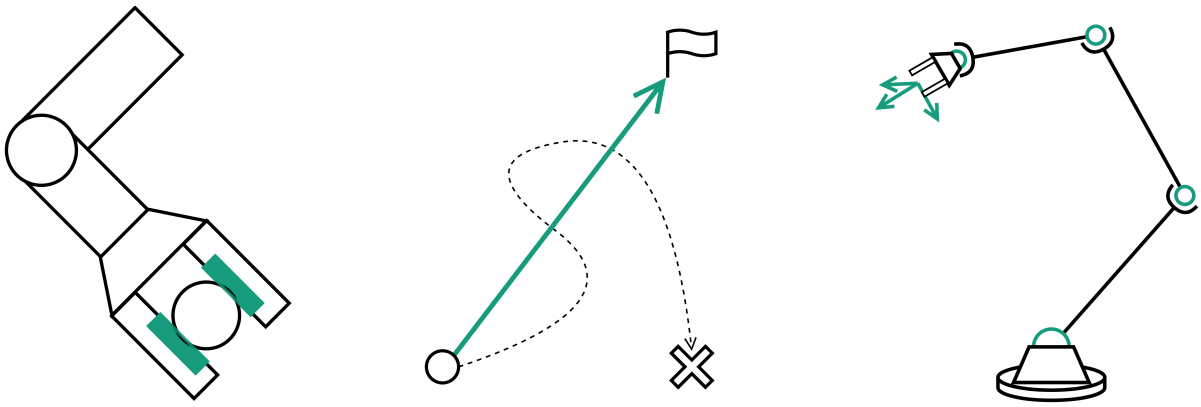
Finally, additional knowledge can be leveraged to enhance success in real-world robotics deployment by minimizing discrepancies between simulated and real environments. For instance, scientific knowledge might be used to fine-tune the simulation environment, while world knowledge through domain randomization could enhance the robustness of the policy training process.

3.3. Description of Methods

This section offers a comprehensive overview of the Guided RL strategies identified during the literature review. These descriptions align with the taxonomy introduced in Section 3.2, establishing a clear link between knowledge sources and practical applications.

Table 3.1.: **Recent research activities** in the field of Guided RL. Various approaches and their methods are connected to the corresponding knowledge sources within the Guided RL taxonomy.

Guided RL Methods	**Scientific Knowledge**	**World Knowledge**	**Expert Knowledge**
State Representation		[131, 33, 132, 145, 87]	[30, 212, 89]
Reward Design	[180, 68]	[139, 207, 25]	[88, 32, 61, 44]
Abstract Learning	[40]	[158, 159]	[128, 208, 206, 4, 22]
Offline RL		[39]	[69, 205, 1, 173, 215, 27, 103]
Parallel Learning	[84, 171]		[134, 45, 78, 15, 123, 168, 92, 124]
Learning from Demonstration	[10, 65]	[26]	[29, 203, 108, 2, 85, 106]
Curriculum Learning		[166, 129, 66, 178, 121, 42]	[98]
Hierarchical RL	[214, 126]	[154, 142, 107, 141, 112, 201]	
Perfect Simulator	[72, 163, 136, 120]		[216, 81, 76, 210]
Domain Randomization		[130, 160, 146, 194]	[189, 154, 137, 200]
Domain Adaptation			[24, 86, 218, 118, 77, 155, 164, 104]



(a) State Representation (b) Reward Design (c) Abstract Learning

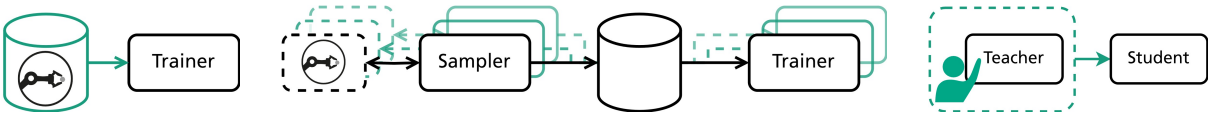
Figure 3.4.: **Guided RL methods for problem representation** (based on prior publication [55]). (a) Example of an enhanced state representation utilizing tactile sensor data (see Section 3.3.1). (b) Employing a dense reward function to steer the policy towards convergence (see Section 3.3.2). (c) Abstract learning across various action spaces, such as joint or end-effector space (see Section 3.3.3).

3.3.1. State Representations

Selecting an appropriate state representation is crucial for addressing learning tasks, as it defines the agent's observable environment. Crafting the observation space in a task-specific manner, supported by measurable sensor data, can significantly boost training efficiency (see Fig. 3.4a).
Melnik et al. [131] enhanced the shadow-dexterous hand with tactile sensor data, ultimately improving the performance of RL agents. Other studies, like Church et al. [33], have explored tactile-based RL agents that learn from tactile inputs depicted as depth images. They achieved zero-shot policy transfer using a GAN that translates real tactile images to simulated depth images. Ning et al. [145] introduced a robotic ultrasound imaging system where observations comprised a concatenated latent vector from two standard autoencoders. Chen et al. [30] assessed the feasibility of conveying ambient sounds to inform 3D scene structures. Xu et al. [212] constructed a dataset of 15,000 transparent objects and developed TransparentNet to estimate depth images, accounting for light refraction and absorption. Ji et al. [89] proposed a state estimator for quadrupedal locomotion to enhance state representation.
Some research combines existing sensor modalities. For example, Miki et al. [132] created a state representation merging proprioception and exteroception for the quadrupedal ANYmal, facilitating locomotion across diverse terrains with occasional sensor failures. Jangir et al. [87] demonstrated a manipulator setup with two cameras, incorporating a transformer-based cross-view attention mechanism to extract related features.

3.3.2. Reward Design

Reward design focuses on refining reward functions, both in terms and parameters, and automatically learning them from data.
This approach is an effective means of incorporating expert knowledge into RL. For intricate tasks where conventional RL algorithms struggle to converge, crafting suitable dense reward functions can enhance sample efficiency and overall performance (see Fig. 3.4b). Research incorporating reward shaping includes Jestel et al. [88], who developed a robust multi-robot navigation policy that learned emergent behaviors in scenarios like swapping and intersections while recovering from dead ends. Siekmann et al. [180] designed a parametric reward function for common bipedal gaits, demonstrating successful policy transfer to the real robot Cassie. Fu et al. [68] proposed a bio-inspired reward function for locomotion to minimize energy consumption while walking and generating natural gaits based on command velocity. Eteke et al. [61] presented a skill learning framework that derives rewards from minimal demonstrations using a hidden Markov model, yielding better performance than sparse reward signals. Chiang et al. [32] employed AutoRL [151] to automatically apply reward-shaping techniques for navigating a mobile robot in extensive indoor environments.
Reward learning often emerges in human-robot interaction contexts, where users rate agent trajectories to derive a reward function. Myers et al. [139] introduced a multi-modal reward learning method that requires users to rank given trajectories. Wilde et al. [207] proposed a new feedback mechanism using a slider bar for scaled user ratings. Cabi et



(a) Offline RL (b) Parallel Learning (c) Learning from Demonstration

Figure 3.5.: **Guided RL methods for learning strategy** (based on prior publication [55]). (a) Offline RL from a recorded dataset (see Section 3.3.4). (b) Parallel learning utilizing multiple samplers and trainers (see Section 3.3.5). (c) Learning from Demonstration with distilled student policies (see Section 3.3.6).

al. [25] introduced reward sketching for efficiently gathering dense human feedback to train reward models. Escontrela et al. [44] implemented an adversarial RL approach [156] to adjust rewards, effectively translating the walking style of a real dog to a robotic counterpart.

3.3.3. Abstract Learning

Beyond carefully selecting the observation space, the choice of the action space significantly influences the representation of the learning problem (see Fig. 3.4c). Approaches that employ an abstract action space typically exhibit enhanced sample efficiency, with some methods integrating hybrid learning and model-based strategies. Varin et al. [196] provide an introductory comparison of commonly utilized action spaces in various manipulation tasks.

In manipulation, some studies advance beyond the traditional end-effector space. For example, Martin-Martin et al. [128] introduced variable impedance control in the end-effector space, enhancing exploration ease and robustness to disturbances. Wong et al. [208] presented OSCAR, a data-driven version of Operational Space Control [96] that adapts to changes in manipulation dynamics. Bogdanovic et al. [22] proposed a policy learning approach that manages impedance and desired position in joint space, comparing it with torque control and fixed gain PD controllers. Duan et al. [40] suggested a task space for bipedal locomotion, where the policy learns to select foot setpoints, which are then translated to joint-level control by an inverse dynamics controller.

Other methods leverage learned action spaces to simplify the learning problem. Pertsch et al. [158, 159] utilized offline datasets to learn latent space representations of action sequences along with prior distributions. They demonstrated that these priors can guide policy learning in new downstream tasks, allowing agents to efficiently tackle long-horizon challenges like robotic manipulation. Whitney et al. [206] and Allshire et al. [4] proposed latent action representations for manipulation that effectively handle dynamic manipulation settings, showing improvements in sample efficiency and performance within pixel-based continuous control environments.

3.3.4. Offline RL

Offline RL, or batch RL, can significantly enhance the sample efficiency of other RL methods by training policies on offline data. Given the novelty of this field, researchers are

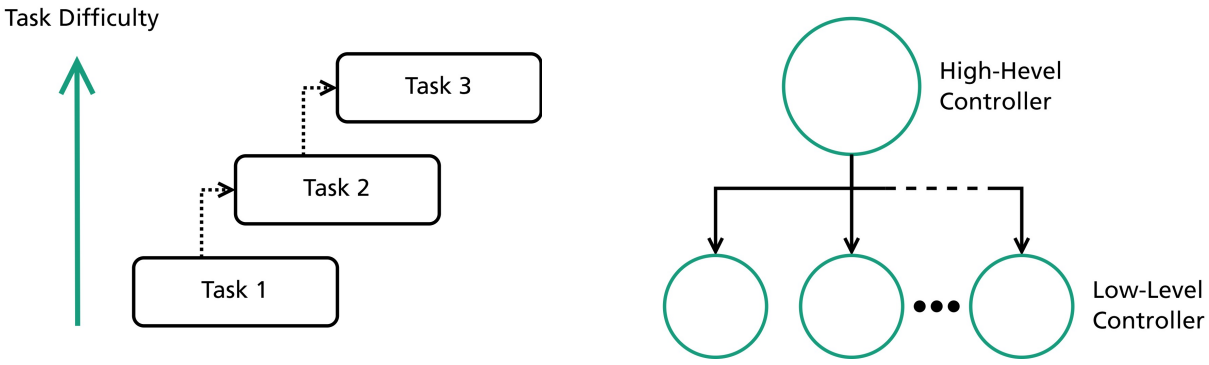
primarily focused on developing algorithms that yield high-performance policies. Offline datasets are typically gathered from past online RL training runs or preprocessed real-world sensor recordings (see Fig. 3.5a). More details on offline RL can be found in [110].
Offline RL faces the extrapolation problem, where the policy may generate out-of-distribution actions that are inaccurately overestimated by the value function [69]. Various algorithms have shown robustness against this issue, including Batch-Constrained Deep Q-learning [69], Random Ensemble Mixture [1], Critic Regularized Regression [205], Implicit Q-learning [103], and MuZero Unplugged [173].
Another approach within offline RL involves employing machine learning techniques. For instance, Chen et al. [27] utilized a transformer architecture conditioned on the desired reward, past states, and actions to produce actions that achieve a specified return. Other studies focus on the offline data itself. Yarats et al. [215] proposed using reward-free unsupervised data first, then annotating the reward to train an RL policy.
Efforts to create extensive offline datasets have also emerged. For example, Dasari et al. [39] introduced an open database for learning vision-based robotic manipulation models, comprising 15 million video frames across seven robot manipulators.

3.3.5. Parallel Learning

Parallel Learning optimizes one or more heterogeneous hardware resources through parallelization (see Fig. 3.5b). It also addresses scalability and robustness to accommodate varying learning process sizes.
Numerous parallel learning architectures have been developed in recent years, including A3C [134], IMPALA [45], Ape-X [78], D4PG [15], and R2D2 [92]. Each of these architectures demonstrates robustness in parallel deployment, leading to significant improvements in sample efficiency and final policy performance compared to baseline methods.
Adjacent optimization paradigms, like evolutionary strategies [171], can also scale to large extents and produce competitive policies relative to RL-based methods. Mania et al. [124] proposed a scalable variant of random search called augmented random search, which yields near-optimal policies.
Some studies utilize hardware accelerators to enable extensive parallelization, facilitating the training process [114]. For instance, Makoviychuk et al. [123] introduced Isaac Gym, a fully GPU-based RL simulator capable of simulating numerous environments simultaneously. Building on this, Rudin et al. [168] employed Isaac Gym to train a quadruped to walk across progressively complex terrains in mere minutes.
The combination of other optimization techniques with RL appears promising. Jaderberg et al. [84] presented a two-level optimization evolutionary process targeting a population of RL agents, enabling agents to play a complex 3D game with performance comparable to humans.

3.3.6. Learning from Demonstration

Although learning from demonstration [172] is a distinct research area, many RL strategies incorporate such techniques. For foundational insights into this field, refer to Osa et al. [148] and Billard et al. [20].



(a) Curriculum Learning (b) Hierarchical RL

Figure 3.6.: **Guided RL methods for task structuring** (based on prior publication [55]). (a) Curriculum learning for progressively complex tasks (see Section 3.3.7). (b) Hierarchical RL utilizing dedicated low-level policies for various subtasks (see Section 3.3.8).

In addition to standard methods like behavior cloning [12] (offline supervised learning) and DAgger (online expert imitation) [167], novel approaches have emerged. Florence et al. [65] proposed implicit behavior cloning using an energy-based model to represent the policy. Laskey et al. [106] introduced the DART algorithm, which collects demonstrations with injected noise and adjusts the noise level according to the trained policy.
Some methodologies train a teacher policy on fixed task setups through RL and subsequently distill a policy capable of interpolating between different setups. For instance, [10] utilized neural dynamical policies [11] for this purpose. Others derive a student policy from a teacher policy based on accurate state information, such as Chen et al. [29], whose vision-based policies successfully reorient objects in the shadow hand domain, or Lee et al. [108], who also distilled vision-based policies for their RGB-stacking benchmark. Some studies leverage classical optimization methods to represent the expert and convert their demonstrations into trainable policies, like Wang et al. [203], who combined RL and imitation learning with the OMG planner [202] as an expert.
In addition to teacher models, some research benefits from human demonstrations. Akbulut et al. [2] introduced the ACNMP framework, merging supervised and RL to retain old skills from robot demonstrations while adapting to new environments. James and Davison [85] presented a coarse-to-fine discrete RL method to tackle sparse reward manipulation tasks with minimal Demonstration and exploration data. Celemin et al. [26] incorporated human corrective feedback into the action domain through an LfD strategy guided by an RL algorithm that filters out non-reward-maximizing human input.

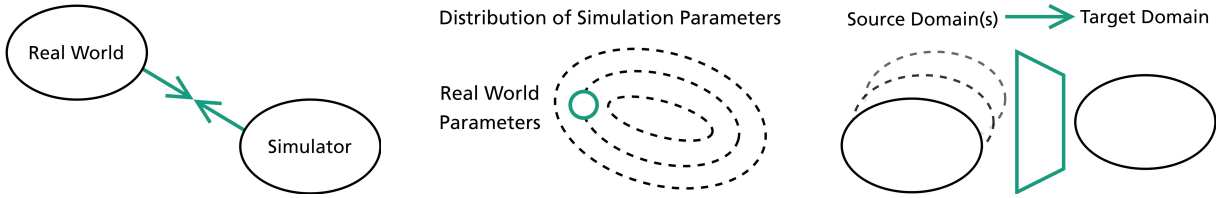
3.3.7. Curriculum Learning

In RL, curriculum learning [19] provides a framework to improve sample efficiency by structuring tasks. It involves developing policies for complex tasks in a progressive manner, progressively increasing the difficulty (see Fig. 3.6a). This approach can reduce convergence time and assist in solving challenges that may be too complex to tackle from the outset [143, 185]. Most methodologies rely on either expert insights for gradually increasing task difficulty or data-driven methods for automatic curriculum generation.

Matiisen et al. [129] introduced a framework for automatic curriculum learning called teacher-student curriculum learning, where a teacher selects appropriate subtasks based on the student's learning progress. Klink et al. [98] proposed self-paced contextual reinforcement, granting agents control over the intermediate task distribution. Florensa et al. [66] presented a method for reverse curriculum generation, enabling robots to gradually learn to reach more distant goals from nearer ones. Similarly, Sharma et al. [178] devised a curriculum of initial states, where agents learn to reset to generated subgoals based on their performance. Rodriguez and Behnke [166] developed an approach for learning omnidirectional locomotion in humanoid robots, progressively increasing task difficulty with scheduled target velocities.
Lastly, some studies utilize curriculum learning for complex tasks involving multiple goals or robots. For instance, Luo et al. [121] established a curriculum to gradually adjust precision requirements for multi-goal reach experiments, demonstrating performance improvements. Eoh et al. [42] applied curriculum learning to challenging multi-robot object transportation tasks, incrementally increasing the transportation distance and the number of robots involved. Leyendecker et al. [111] combined reward curriculum with domain randomization to develop a robust sim-to-real transferable policy for executing manipulation tasks in industrial environments.

3.3.8. Hierarchical RL

Hierarchical RL [16, 152] enhances training efficiency and effectiveness by breaking down complex tasks into a hierarchy of subtasks (see Fig. 3.6b). These subtasks are typically managed by specialized low-level policies, coordinated by a more general high-level policy, allowing for potential reuse in a sample-efficient manner. A foundational model for hierarchies is the options framework introduced by [187], where high-level policies select options instead of individual actions, executed by low-level policies that generate outputs for specified durations, facilitating temporal abstraction. Bacon et al. [8] expanded on this concept with an option formulation for the critic.
Various hierarchical algorithms have been proposed, such as Yang et al. [214], who introduced hierarchical-deep deterministic policy gradient for continuous robotic control tasks, enabling simultaneous learning of compound and essential skills across two hierarchical levels. Nachum et al. [142] developed a general, data-efficient Hierarchical RL algorithm named HIRO, designed to learn complex robotic behaviors through low-level controllers supervised by automatically generated high-level goals.
Others have implemented hierarchies in their policies, like Peng et al. [154], who proposed a two-level hierarchical control framework for diverse locomotion skills in a simulated bipedal robot. Le et al. [107] introduced a hierarchical guidance framework that effectively utilizes expert feedback, directing the low-level learner toward pertinent areas of the state space. Margolis et al. [126] tackled dynamic locomotion over uneven terrain using a high-level controller to generate a trajectory based on visual inputs, which a low-level controller then tracks. Nachum et al. [141] employed a hierarchy to learn low-level goal-reaching skills coordinated by a high-level controller for multi-agent object manipulation. Wang et al. [201] applied a hierarchical policy in a 6D, cluttered scene for grasping, learning an embedding space based on expert plans to select sampled plans via a critic and appropriate



(a) Perfect Simulator (b) Domain Randomization (c) Domain Adaptation

Figure 3.7.: **Guided RL methods for sim-to-real transfer** (based on prior publication [55]). (a) Perfect simulator to minimize the reality gap (see Section 3.3.9). (b) Domain randomization to align the real-world parameter distribution (see Section 3.3.10). (c) Domain adaptation for matching source and target domains (see Section 3.3.11).

options [187] through an option classifier. Lastly, Li et al. [112] utilized a hierarchical structure for interactive navigation tasks, where a high-level policy generates subgoals and selects low-level policies to produce task-phase-specific robot actions.

3.3.9. Perfect Simulator

A promising approach for effective real-world deployment is the development of a realistic simulator that minimizes the reality gap (see Fig. 3.7a). Simulators that accurately reflect real-world physics are valuable, as they may enable the direct zero-shot transfer of trained models into real-world applications [220]. Enhancing the realism of a simulated environment involves improving robot models, physics computation, and environment representations.

System identification [119] involves creating an accurate mathematical model of a physical system. In robotics simulation, precisely tuning physical parameters like friction, weight, or elasticity can greatly enhance simulator realism. Additionally, machine learning techniques can be applied either offline [91] or online, as demonstrated by Yu et al., who predict dynamics model parameters in real-time [216].

Accurate simulation of complex robot dynamics demands careful selection of a physics engine. Erez et al. [43] evaluated simulation performance and speed concerning the numerical challenges of multi-body dynamics in robotics. The chosen physics simulator must also cater to the specific needs of the robotics application; as noted by [34], different simulators are preferred across various robotics sub-domains based on factors like sensor relevance, dynamic contacts, or friction modeling. Muratore et al. [136] applied dynamics randomization and employed a novel algorithm to adjust parameters, preventing overfitting to simulator dynamics. Lowrey et al. [120] utilized real-world robot data to identify robot parameters accurately [102], facilitating direct policy transfer from simulation to reality. Heiden et al. [76] proposed a hybrid simulator incorporating learned neural networks to alternate between analytical and learned computations of physical effects. Xia et al. [210] introduced the Gibson environment, designed for realistic visual perception of active agents based on real-world data.

Lastly, a precise representation of the environment can substantially minimize the reality gap. Ramos et al. [163] introduced BayesSim, a framework providing adaptive Bayesian estimates for simulation parameters through simulation-based inference, while Golemo et

al. [72] developed neural-augmented simulation to enhance robotic simulators with real robot trajectories. Hwangbo et al. [81] presented a neural network trained on real data for the robot ANYmal, converting policy actions into torque values for the simulation model.

3.3.10. Domain Randomization

Domain randomization involves extensively randomizing simulation parameters across various distributions (see Fig. 3.7b). Rather than meticulously modeling real-world parameters in simulation, the real world is treated as another variation of these distributions [220]. Depending on the parameters to be randomized, standard methods include randomizing either visual or dynamic components. A comprehensive review of randomization simulations can be found in [138].
Tobin et al. [189] pioneered randomizing rendering within simulators to transfer neural networks to reality for robotic control. Mehta et al. [130] proposed active domain randomization, which learns a parameter sampling strategy to optimize the informative ranges of randomization. OpenAI et al. [146] introduced Automatic Domain Randomization that adjusts environmental randomization parameters based on policy success when solving Rubik's Cube with a real robot hand. Prakash et al. [160] presented structured domain randomization, which generates context-aware synthetic data by considering the scene's structure. Instead of focusing solely on visual components, Peng et al. [154] introduced dynamics randomization, encompassing parameters such as link masses, joint damping, and PD gains. Muratore et al. [137] proposed neural posterior domain randomization, which fine-tunes simulator parameters using a limited number of real-world rollouts to align with observed dynamics. Tsai et al. [194] utilized a single human demonstration to identify the simulator's distribution over dynamics parameters, adapting domain randomization to minimize the sim-to-real gap.
Similar concepts in visual and dynamics randomization have been adopted in other studies, introducing perturbations to create more robust agents. For instance, Wang et al. [200] considered noisy rewards. Other works have implemented noisy sensor signals [166] or random external forces [168] for effective policy deployment in the real world.

3.3.11. Domain Adaptation

Domain adaptation methods aim to bridge the reality gap by training adaptation modules, often represented as autoencoders, capable of transforming one domain into another, such as real-world camera images into simulation-like images (see Fig. 3.7c). The target domain may be simulated environments, the real world, or abstract latent spaces. Wang et al. [204] provide a thorough survey in this area of research.
Several researchers have explored vision-based domain adaptation. Bousmalis et al. [24] implemented this concept through GraspGAN, training an adaptation module to convert synthetic images from simulation to more photorealistic representations. James et al. [86] introduced randomized-to-canonical adaptation networks that learn to project synthetic images from randomized simulations into the style of canonical simulations. Rao et al. [164] developed RL-CycleGAN to convert synthetic images into more realistic ones. Liu et al. [118] presented Real-Sim-Real, which adapts real-world states into simplified ones

through a segmentation model. Zhang et al. [218] proposed adaptation modules that are trained independently of the DRL agent and can be applied across various scenarios, such as indoor and outdoor navigation. Hoeller et al. [77] introduced a navigation policy for the quadrupedal robot ANYmal, enabling navigation in cluttered environments with static and dynamic obstacles.
Other studies examined the use of adaptation modules to address environmental factors. Peng et al. [155] presented a framework for training quadrupedal robots to imitate agile locomotion skills from animals, facilitating sim-to-real transfer of learned policies through efficient domain adaptation. Kumar et al. [104] introduced the rapid motor adaptation algorithm, which adapts to previously unseen real-world scenarios in real time.

3.4. Evaluation Study

This section provides a systematic assessment of Guided RL methods. Combining various techniques and approaches will be shown to improve all three aspects of Guided RL: efficiency, effectiveness, and sim-to-real transfer. First, the methodological approach is outlined, followed by key findings related to individual methods and their combinations.

3.4.1. Methodical Approach

Table 3.2 provides an overview of the Guided RL methods discussed in Section 3.3. Each method is evaluated against the three dimensions of Guided RL (Section 3.1), determining if (i) training time has decreased (efficiency), (ii) policy performance has improved (effectiveness), or (iii) the policy has been successfully applied in real-world scenarios (sim-to-real). These evaluations are based on the claims made by the authors, supported by figures, tables, or textual evidence. The table's last column also details the Guided RL methods employed in each approach, offering a comprehensive view of their achievements across the three dimensions.
Based on these categorized references, Fig. 3.8 illustrates the normalized contributions of the respective methods in terms of efficiency, effectiveness, and sim-to-real transfer. For example, several papers that utilize hierarchical RL have reported enhancements in policy performance, indicating this method's significant role in boosting learning effectiveness. To strengthen the statistical validity of this evaluation, the papers corresponding to each method are considered, and all papers adopting that method (see Table 3.2).

3.4.2. Key Insights on Individual Methods

The quantitative evaluation of references (Fig. 3.8) reveals that specific methods are linked with efficiency, effectiveness, or sim-to-real transfer enhancements. The following insights can guide the selection of methods to boost the likelihood of achieving greater efficiency, effectiveness, or successful real-world deployment.

Table 3.2.: **References for the evaluation study** on efficiency, effectiveness, and sim-to-real. Approaches with an asterisk achieved simultaneous improvements along all three dimensions.

	Ref	Efficiency	Effectiveness	Sim-to-Real	Guided RL Methods
State Representation (SR)	[131]	✓	✓		SR (PL,DR)
	[132]			✓	SR (RD,LfD,CL,DR)
	[33]	✓		✓	SR (PL,DR,DA)
	[30]				SR
	[212]				SR
	[145]			✓	SR
	[87]		✓	✓	SR (DR)
	[89]		✓	✓	SR (CL,DR)
Reward Design (RD)	[180]			✓	RD (DR)
	[88]			✓	RD
	[139]	✓		✓	RD
	[32]		✓	✓	RD (CL)
	[61]*	✓	✓	✓	RD (LfD)
	[68]			✓	RD (LfD,DR,DA)
	[207]		✓	✓	RD
	[25]		✓		RD (OL)
	[44]		✓	✓	RD (PL,LfD,DR)
Abstract Learning (AL)	[128]*	✓	✓	✓	AL (RD,CL,DR,DA)
	[40]*	✓	✓	✓	AL (RD,PS)
	[158]	✓	✓		AL (LfD)
	[208]		✓		AL (PS,DR)
	[159]	✓	✓		AL (HL,LfD)
	[206]	✓	✓		AL (SR,LfD)
	[4]	✓			AL (LfD)
	[22]		✓	✓	AL (RD)
Offline RL (OL)	[69]		✓		OL
	[1]		✓		OL
	[103]	✓	✓		OL
	[205]		✓		OL
	[39]				OL (PS)
	[173]	✓	✓		OL (LfD)
	[215]		✓		OL (LfD)
	[27]		✓		OL
Parallel Learning (PL)	[134]	✓	✓		PL
	[45]	✓	✓		PL
	[78]	✓	✓		PL
	[15]		✓		PL
	[92]	✓	✓		PL
	[84]		✓		PL
	[123]	✓			PL (PS,DR)
	[168]	✓		✓	PL (SR,RD,CL,PS,DR)
	[124]	✓	✓		PL
	[171]	✓			PL
Learning from Demonstration (LfD)	[29]				LfD (SR, PL, CL,DR)
	[10]*	✓	✓	✓	LfD (AL,PL)
	[203]		✓	✓	LfD (SR,AL,PL,DR)
	[65]		✓	✓	LfD

	[108]		✓	✓	LfD (SR,RD,AL,OL,CL,DR)
	[2]*	✓	✓	✓	LfD (SR,AL)
	[85]*	✓	✓	✓	LfD (SR,AL)
	[26]*	✓	✓	✓	LfD (AL)
	[106]*	✓	✓	✓	LfD (AL,DR)
Curriculum Learning (CL)	[166]			✓	CL (RD,PS,DR)
	[129]		✓		CL
	[66]		✓		CL
	[121]*	✓	✓	✓	CL
	[178]	✓	✓		CL
	[42]	✓	✓		CL
	[98]*	✓	✓	✓	CL (DR)
	[111]*	✓	✓	✓	CL (RD,AL,PL,PS,DR)
Hierarchical RL (HL)	[154]		✓		HL (RD,LfD)
	[214]		✓		HL
	[142]	✓	✓		HL
	[107]	✓	✓		HL (LfD)
	[201]		✓	✓	HL (SR,LfD)
	[126]		✓	✓	HL (RD,AL,PL,LfD,DR)
	[141]		✓	✓	HL (DR)
	[112]		✓		HL (RD)
Perfect Simulator (PS)	[216]		✓		PS
	[76]	✓		✓	PS (LfD)
	[120]			✓	PS (PL)
	[72]		✓	✓	PS (DA)
	[163]				PS (DR)
	[136]			✓	PS (DR)
	[81]			✓	PS (DR)
	[210]				PS (DA)
Domain Randomization (DR)	[189]			✓	DR
	[130]*	✓	✓	✓	DR
	[160]				DR
	[154]		✓	✓	DR
	[137]			✓	DR
	[146]		✓	✓	DR (PL,LfD,CL)
	[194]		✓	✓	DR (RD)
	[200]		✓		DR
Domain Adaptation (DA)	[24]*	✓	✓	✓	DA (DR)
	[86]*	✓	✓	✓	DA (DR)
	[118]*	✓	✓	✓	DA
	[218]		✓	✓	DA
	[77]*	✓	✓	✓	DA (SR,PL,CL,DR)
	[155]			✓	DA (DR)
	[164]*	✓	✓	✓	DA (DR)
	[104]		✓	✓	DA (SR,RD,CL,DR)

Enhancing Efficiency

The findings indicate that parallel learning architectures, abstract learning, and learning from demonstration data particularly expedite the RL training process. Firstly, efficient parallelization of algorithmic components allows for scaling the learning problem [134, 45, 78]. Secondly, simplifying the learning task through task-specific action spaces or hybrid model-based and model-free approaches can enhance overall efficiency.
Lastly, training with expert demonstrations often provides rich information, thus accelerating policy training [10, 203, 194]. Moreover, more informative state representations may improve efficiency by implementing a curriculum for progressively tackling challenging tasks or utilizing precise simulation environments.

Enhancing Effectiveness

Regarding effectiveness, offline RL, hierarchical RL, and curriculum learning influence overall policy performance. Training policies on recorded datasets can be a productive route to effectiveness, as the full spectrum of potential information can be extracted from the samples [69, 205, 25]. Additionally, employing curricula or hierarchical learning schemes enhances policy performance and addresses more intricate robotic tasks [166, 168, 129].
Furthermore, other methods can also improve overall task performance, such as clearly defining the learning problem, incorporating demonstration data, or leveraging parallel learning structures.

Enhancing Sim-to-Real Transfer

As the evaluation results indicate, domain randomization, domain adaptation, and perfect simulators are frequently used for transitioning policies trained in simulation to the real world. Firstly, domain randomization is a widely adopted method for effective sim-to-real transfer [130, 154, 168], likely due to its straightforward implementation applicable to most robotics challenges. Secondly, domain adaptation through adaptation modules is often employed for successful transfers between simulated and real environments [24, 86, 77].
Lastly, many methods aim to minimize the reality gap by enhancing simulator realism, focusing on improved robot models, better physics calculations, or enhanced environment representation to close the sim-to-real gap [216, 43, 104].

3.4.3. Key Insights on Guided RL Compliance

Efficient and effective policy training and real-world robotics deployment are crucial components of Guided RL. The ultimate goal is to integrate these three dimensions effectively (see Fig. 3.2). To achieve this, correlations among the papers that have successfully improved all three dimensions are analyzed, referred to as *Guided RL compliant* (Table 3.2, marked with an asterisk). These papers identify three recurring patterns that enhance training efficiency, boost policy effectiveness, and facilitate the transfer from simulation to real-world applications.

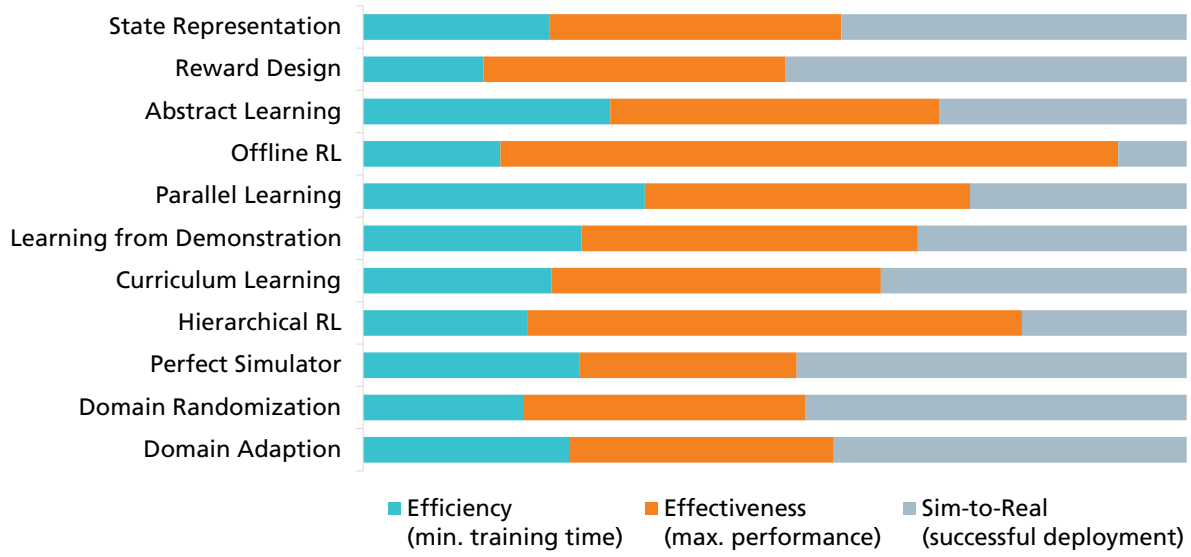


Figure 3.8.: **Evaluation study on Guided RL** methods (based on prior publication [55]). Based on the literature review (see Table 3.2), quantitative results depict the relative contributions of various methods toward accelerating the training process (efficiency), enhancing overall policy performance (effectiveness), and successful real-world deployment (sim-to-real).

Utilizing Multiple Guided RL Methods

Firstly, Guided RL-compliant papers frequently employ a variety of Guided RL methods. For example, [40, 10, 85] utilize at least three different Guided RL approaches, while [128, 111, 77] implement five or more methods to achieve enhancements across all three Guided RL dimensions.

Combining Specific Guided RL Methods

Secondly, not only does the quantity of methods matter, but combining specific Guided RL techniques can also increase the likelihood of improving efficiency, effectiveness, and sim-to-real transfer simultaneously. The analysis of Pearson correlation coefficients reveals significant positive associations between DA-DR for sim-to-real transfer and PL-DR for data-driven scalability. Furthermore, the findings indicate that AL-LfD is often integrated through reduced action spaces, while RD-PS incorporates additional knowledge for both simulation and reward design.

Leveraging Multiple Levels of the Guided RL Pipeline

Lastly, Guided RL-compliant papers tend to incorporate multiple levels of the Guided RL pipeline (see Fig. 3.1). For instance, [61, 106, 98] utilize Guided RL methods across two or three pipeline stages. Additionally, several of the compliant papers [77, 111] integrate knowledge across all four levels of the Guided RL pipeline, facilitating improvements in training efficiency, policy performance, and sim-to-real transfer simultaneously.

3.5. Practical Guidelines for Methods Selection

This section offers a set of practical guidelines on when to select which of the Guided RL methods (see Section 3.3). The survey identified three typical robot learning tasks: locomotion, manipulation, and navigation. Beginning with general recommendations for all robotic tasks, this section will then provide specific guidelines for these three main areas of robot learning.

General Recommendations

The evaluation study (see Section 3.4) has already shown that using multiple Guided RL methods and leveraging various levels of the Guided RL pipeline can enhance efficiency, effectiveness, and the transfer from simulation to real-world applications simultaneously. Therefore, it is generally essential to carefully consider each Guided RL method when establishing the RL training pipeline for a new robotic task. However, some methods can and should be applied at least to a certain extent across most robotic tasks. The following two are generally recommended Guided RL methods.

1. **Parallel Learning:** Parallel learning schemes significantly improve the speed of training iterations and are now widely adopted for various robotic tasks. With the advent of GPU-accelerated frameworks for robot learning, experiences can now be collected from thousands of robots operating in parallel on local machines. This advancement has considerably increased speed, making parallel learning one of the most popular Guided RL methods that benefit each robotic learning task.

2. **Perfect Simulator:** Accurate simulation models are crucial for minimizing the reality gap, facilitating more effortless transfer of trained models to real robots. A realistic simulator improves training effectiveness by authentically representing real-world physics and enabling better tuning of robot models and environmental representations. Techniques such as system identification and the incorporation of real-world data help to find accurate estimates for simulation parameters, enabling randomization approaches that enhance effective sim-to-real transfer.

Case I: Locomotion

Locomotion is the area of robot learning focused on generating control policies that enable robots to navigate their environments robustly and quickly. These environments range from flat terrain to more complex and challenging surfaces, such as stairs, stones, or inclined pathways. Therefore, locomotion highly benefits from structured rewards and tasks and randomization techniques.

1. **Reward Design:** Learning to move can be difficult, especially for articulated robots like quadrupeds or those needing balance, such as humanoid or two-wheeled robots. Complex reward functions are often used to address this challenge, incorporating various regularization terms to promote efficient exploration.

2. **Domain Randomization:** Agile robots frequently operate near their physical limits, such as contact stability constraints or the torque capabilities of their actuators. Domain randomization techniques are employed in simulations to enhance the generalization of control policies across extreme scenarios. This involves varying robot dynamic parameters, including mass distribution, stiffness, damping, and actuator limits.

3. **Curriculum Learning:** To enhance both the efficiency and effectiveness of locomotion tasks, curriculum learning techniques are commonly adopted. These methods typically involve a gradual increase in the target velocity for locomotion tasks or progressively challenging the robots with more difficult terrains as their performance improves over time.

Case II: Manipulation

Robotic manipulation is a key area where learning-based control is applied, as these platforms are often more affordable, and experimental setups tend to be more reproducible than in other tasks. Many approaches focus on developing control policies that generalize to unseen objects, grasp challenging items, or perform manipulation tasks requiring high precision. Consequently, manipulation often benefits from randomization and reducing the task complexity.

1. **Domain Randomization:** A major challenge in manipulation involves grasping various, potentially unknown, and complex objects. To address this, domain randomization techniques are commonly used in simulations to help generalize the training of policies across different objects and conditions. Typical randomization parameters often include mass, friction, and the material properties of the objects to be grasped.

2. **Learning from Demonstration:** The reproducibility of experiments in manipulation, combined with effective interfaces for human-machine interaction, allows for demonstrations to guide the training process. These demonstrations may include real-world data from actual manipulation tasks, such as those performed through teleoperation or direct human control. Additionally, teacher-student approaches distill policies based on real-world (visual) sensor inputs.

3. **Abstract Learning:** Manipulation and object grasping often require high-level reasoning and complex scene perception using vision or depth information. As a result, simpler action spaces are frequently chosen over joint-space control to enhance exploration in these challenging and often sparse task settings. Typical architectures involve an RL controller that operates on task space setpoints (like end-effector commands), with a low-level controller translating these into desired joint configurations.

Case III: Navigation

Learning-based navigation aims to develop control policies that guide robots on where to go. This task often encounters challenges due to environments that are partially unknown,

difficult in terms of traversability, or subject to dynamic obstacles, such as those found in multi-robot scenarios or situations involving human interactions. As a result, navigation greatly profits from integrating knowledge into the task structure and high-level action spaces.

1. **Hierarchical RL:** In multi-robot learning tasks, incorporating a form of hierarchy can enhance initial exploration and the final performance of the policy. For instance, a high-level policy can orchestrate different local low-level policies across various robots. Alternatively, a high-level policy can make real-time decisions between different locomotion skills to reach a target destination.

2. **Curriculum Learning:** Gradually increasing the difficulty of tasks can make navigation more manageable and improve sample efficiency and effectiveness. One method involves progressively increasing the density of obstacles in the environment based on the policy's performance, thereby raising the challenge. Another approach could be incrementally increasing the distance between the starting and target positions, which fosters more effective exploration of the environment.

3. **Abstract Learning:** Navigation tasks often require high-level reasoning and may involve multi-agent decision-making. Abstracting the action space can be beneficial in these instances. For example, low-level controllers can manage the robot's dynamics or kinematics, allowing the reinforcement learning controller to operate in task space by outputting velocity vectors instead.

3.6. Concluding Remarks

This chapter presented the concept of Guided RL and its potential to enhance the efficiency and effectiveness of training in real-world robotics. The contributions discussed here provide a solid foundation for understanding and applying Guided RL in various robotic contexts. The introduction of a novel taxonomy for Guided RL serves as a modular toolbox, enabling the integration of diverse knowledge sources into the RL training pipeline and enriching the learning process. Additionally, the comprehensive survey of existing Guided RL methods highlights critical aspects of problem representation, learning strategies, task structures, and knowledge transfer from simulation to real-world environments. The quantitative evaluation of these methods demonstrates their potential to improve sample efficiency, task performance, and successful transfer from simulation to real-world scenarios. Furthermore, the practical guidelines for selecting appropriate Guided RL methods cater to the specific needs of typical robotics applications, such as locomotion, navigation, and manipulation tasks. Together, these contributions advance the understanding of Guided RL and provide valuable insights and tools for the robotics community, paving the way for more efficient and effective learning-based control strategies in dynamic robotic systems. All findings from this chapter are available for open access, alongside a comprehensive survey to encourage further research and collaboration. These insights will serve as a foundation for the subsequent case study, which focuses on applying Guided RL methods to the two-wheeled robot evoBOT to train dynamic motions.

4

Case Study: Learning-based Control for evoBOT

Building upon the foundations of Guided RL (see Chapter 3), this chapter presents a case study on learning-based control methods for a dynamic and challenging real-world robotic system. The two-wheeled robot evoBOT, developed by Fraunhofer IML, is selected as a case study platform for this research. evoBOT is a highly flexible mobile robot representing a new class of robotic systems operating on the principle of an inverted pendulum. The robot can handle many objects, making it suitable for various applications. Therefore, the findings from this research contribute to the broader field of robotics, enhancing the understanding of how learning-based approaches can be effectively applied to dynamic and versatile robotic systems. Hence, the key research question to be addressed in this chapter is:

Research Question II: How can Guided RL methods improve the efficiency and effectiveness of learning-based control for the two-wheeled mobile robot evoBOT?

This chapter begins by selecting the most promising methods from the Guided RL taxonomy for the case study on learning-based control (Section 4.1). It then provides an overview of the physics simulation model developed for evoBOT, outlines strategies for optimization using real-world data, and evaluates the resulting sim-to-real gap using a large-scale motion capture system (Section 4.2). Next, the learning-based control framework is introduced, detailing the GPU-accelerated training setup, definitions of the RL tasks, and mathematical formalization of the learning problem to enhance the robot's performance (Section 4.3). The chapter then evaluates the trained policies through simulation experiments, assessing their velocity tracking performance, benchmarking efficiency and effectiveness, and evaluating their adaptability to new tasks. (Section 4.4). Finally, an initial pipeline for transferring the trained policies to the real robot is presented, including an ablation study on the underlying sim-to-real methodologies (Section 4.5).

Disclaimer: Parts of this chapter are based on a prior publication of this thesis [99].

4.1. Selection of Guided RL Methods

This section provides an overview of the Guided RL methods selected to train control policies for the two-wheeled robot evoBOT case study (see Fig. 4.1). The evaluation study (see Section 3.4) demonstrates that employing multiple Guided RL methods and utilizing various levels of the Guided RL pipeline can simultaneously enhance efficiency, effectiveness, and the transfer from simulation to real-world applications. Consequently, it is generally recommendable to consider each Guided RL method when establishing the RL training pipeline for a new robotic task. In line with the practical guidelines for Guided RL (see Section 3.5), the methods selected for this case study include:

1. **Reward Design:** Learning dynamic tasks can be challenging, especially for articulated robots like quadrupeds or those requiring balance, such as humanoid or two-wheeled robots. Dense reward structures are employed for task completion and regularization to address this, facilitating early exploration and later exploitation during training (see Section 4.3).

2. **Parallel Learning:** Collecting training experiences in a parallelized fashion significantly accelerates training iterations and is widely adopted for various robotic tasks. This case study utilizes a GPU-accelerated framework for robot learning, enabling the collection of experiences from thousands of robots operating in parallel on local machines (see Section 4.3).

3. **Curriculum Learning:** To improve both efficiency and effectiveness, curriculum learning techniques are commonly implemented. An adaptive training schedule is incorporated, gradually increasing task difficulty based on the robot's performance (see Section 4.3).

4. **Perfect Simulator:** Accurate simulation models facilitate the sim-to-real transfer of trained models to real robots. A realistic simulator improves training effectiveness by authentically representing real-world physics and enabling better tuning of robot models and environmental representations. System identification methods are combined with real-world data to minimize the sim-to-real gap for the two-wheeled robot (see Section 4.2).

5. **Domain Randomization:** Agile robots often operate near their physical limits, such as contact stability constraints or actuator torque capabilities. Domain randomization techniques are employed in simulation to enhance the generalization of control policies across extreme scenarios by varying robot dynamics, including mass distribution, stiffness, damping, and actuator limits (see Section 4.5).

The specific implementations of the selected Guided RL methods for the case study are detailed in the corresponding sections. In particular, the mathematical foundations for the training setup (rewards, parallel learning, and curriculum learning) are presented in Section 4.3, domain randomization techniques for sim-to-real transfer are addressed in Section 4.5, and methods related to perfect simulator are discussed in the next section.

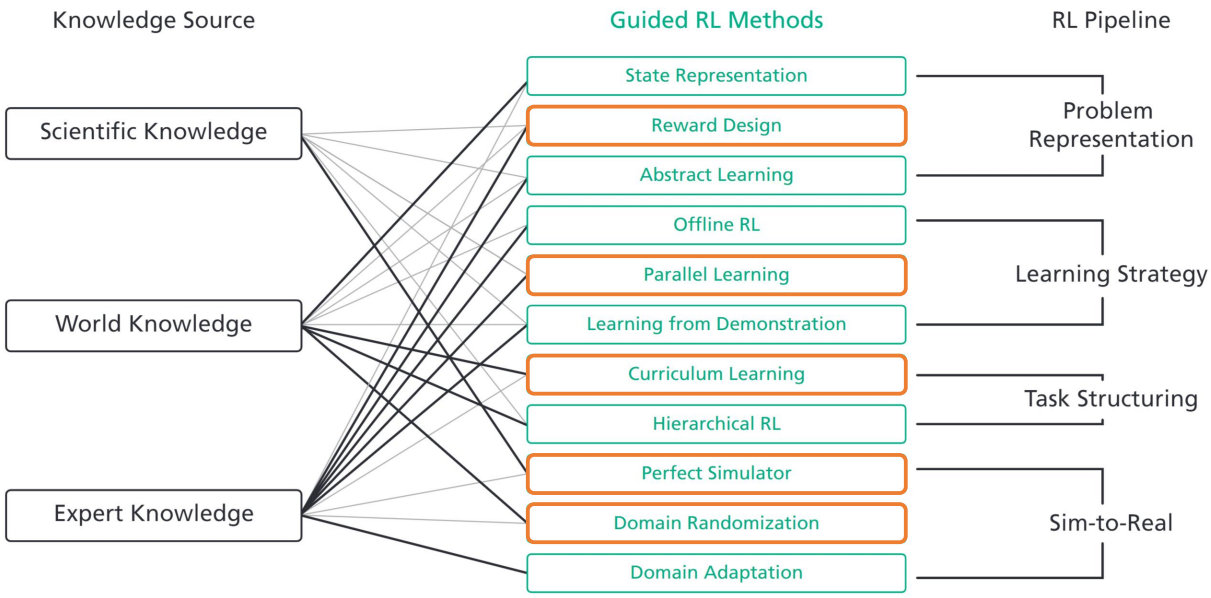


Figure 4.1.: **Selection of Guided RL methods** for learning-based control of evoBOT. The chosen methods are highlighted in orange in the Guided RL taxonomy (see Section 3.2).

4.2. Physics Simulation Model

This section describes the physics simulation model created for evoBOT (see Fig. 4.2), which will build the foundation for training control policies in the case study. It starts with an overview of the simulation model, covering the kinematic model, dynamic parameters, and collision shapes. Then, it outlines system identification strategies to reduce the sim-to-real gap using real-world data from the robot. Finally, it presents results from experimental runs of the robot, using a motion capture system, to assess the sim-to-real gap of the simulation model.

4.2.1. Simulation Modeling in Isaac SIM

The simulation model is designed to comprehensively capture the kinematic and dynamic aspects of the evoBOT while minimizing complexity to improve simulation performance. It uses detailed Computer-Aided Design (CAD) data of the robot, which includes more than 300 individual parts, as the foundation for determining the overall dimensions, weights, and materials of the components (see Fig. 4.3). Additionally, the simulation model offers interfaces to the popular Robot Operating System (ROS).

Kinematic Structure

The overall kinematic structure of the robot consists of links connected by corresponding joints for translational or rotational movements. In particular, the simulation model contains eight links and seven joints, with wheels attached to the main body via revolute

[1] `https://git.openlogisticsfoundation.org/silicon-economy/simulation-model/evobotsimmodel`

Figure 4.2.: **evoBOT simulation model** in NVIDIA Isaac SIM (based on prior publication [99]). Available in the state-of-the-art framework and via the Open Logistics Foundation repository [1].

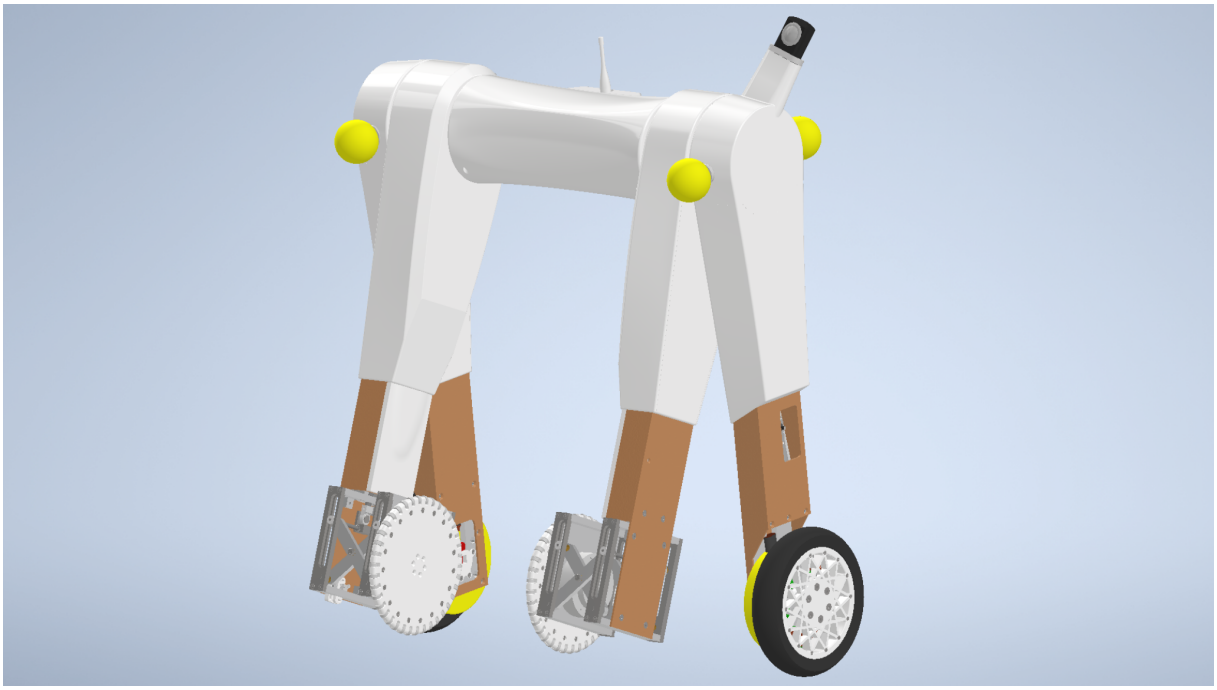
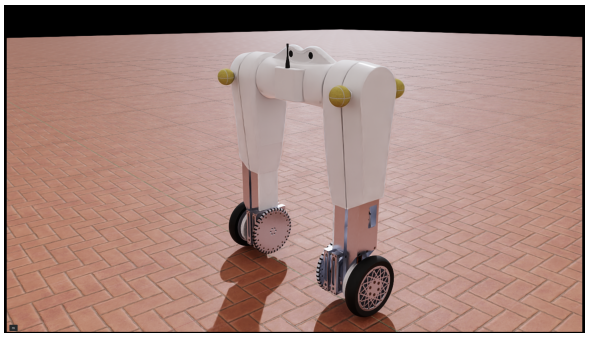
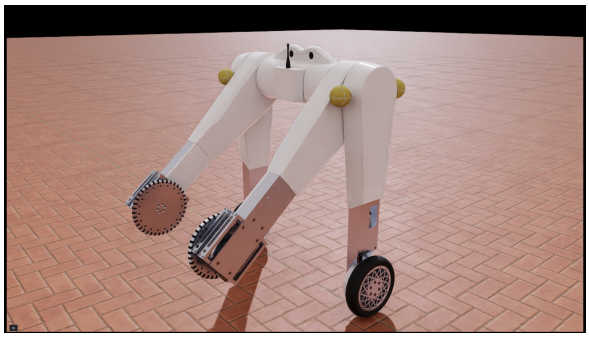


Figure 4.3.: **CAD data** serves as foundation for the evoBOT simulation model [63]. The dimensions, weights, and materials of the individual components are transferred into the simulation.



(a) Arms fixed.



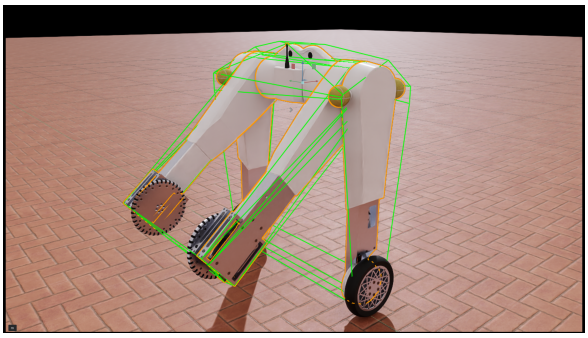
(b) Arms stretched.

Figure 4.4.: **Arm mechanism** in different joint configurations.

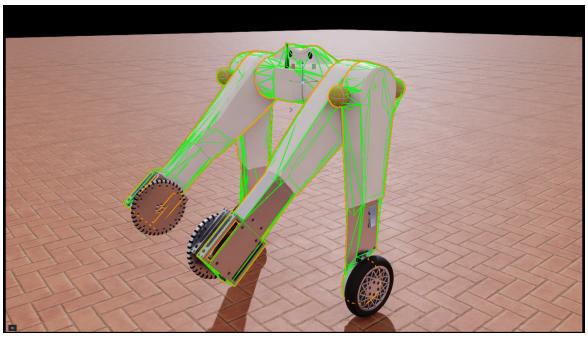
joints. This setup allows for separate control of the right and left wheels. The arms are connected with one revolute joint, permitting simultaneous control (see Fig. 4.4).
Grippers on the evoBOT's arms facilitate object handling, with prismatic joints enabling the grippers to open and close, assisted by rotational joints for precise object placement. Joint drives and an articulation root are integrated into the model to control the joints. The articulation root is defined at the head_link, which serves as the robot's origin, located at the head area of the main body. The model includes several invisible links that provide access to position and orientation data during physics simulations. Due to pitch angle changes from acceleration, the origin may not align with the wheel axis, complicating navigation tasks. To address this, an invisible link, base_footprint, is added to accurately project the wheel axis center onto the ground, enhancing positioning and enabling the derivation of linear velocity from this reference. Additionally, two more invisible links, base_link and grab_center, are included to ensure proper frame alignment during robot learning tasks.
The simulation model is represented in Universal Scene Description (USD), the standard data format for Isaac SIM that enables flexible 3D scene representation and seamless integration of various environments for robot interaction testing. Based on the available CAD data, a Unified Robot Description Format (URDF) file is first generated to retain material properties and import them into Isaac SIM. Moreover, the model is optimized for performance by removing invisible internal parts, and adjustments are made in Blender before defining dynamic parameters and collision shapes within URDF. Feldmeier [63] provides further details on transferring CAD to USD format, reducing the file sizes, and implementing the gripper mechanism.

Dynamic Parameters

Accurate dynamics modeling is essential when creating simulation models for highly dynamic robots like the evoBOT. To achieve this, the masses, centers of mass (CoM), and moments of inertia of all links are calculated and incorporated into the simulation model. Rigid bodies must be assigned to each link to integrate these dynamic parameters, enabling the physics engine to compute gravitational and external forces acting on the components. Masses of the links are determined by segmenting existing CAD files into their respective components while retaining internal parts to preserve dynamic integrity. Detailed CAD



(a) Convex hull.



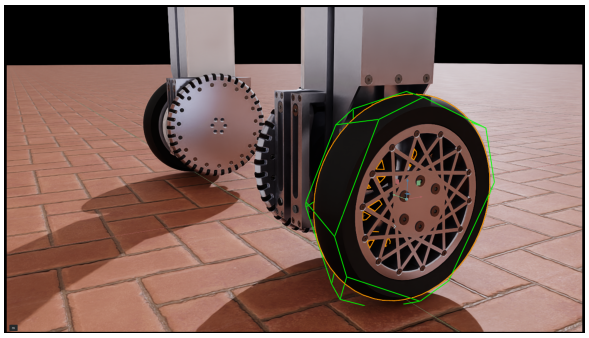
(b) Convex decomposition.

Figure 4.5.: **Main body** with different approximations of collision shapes.

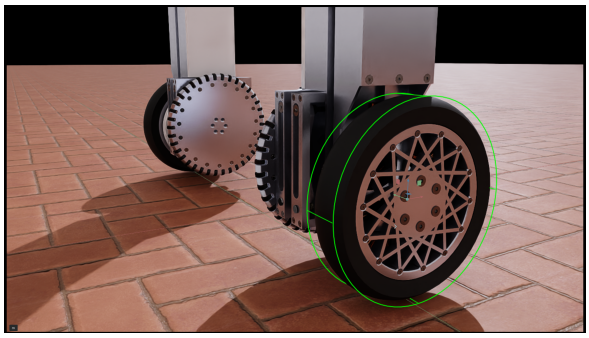
files for electronic components from the manufacturer ensure accurate representation, while other files specify dimensions, materials, and densities. Wiring is omitted for simplicity in mass distribution.
Positioning the CoM is critical for simulating the inverted pendulum. The evoBOT is assumed to be symmetric along the Z-axis, placing the CoM along this axis. The CoM for all links can be estimated using CAD data, applying a translational transformation to account for differences in coordinate origins between the CAD files and the simulation model. Moments of inertia for complex shapes can also be recalculated based on CAD data. These moments of inertia are represented as an inertia tensor, which must be transformed into principal moments of inertia and principal axes for use in Isaac SIM.

Collision Shapes

Collision shapes are essential for facilitating physical interactions between the links of the robot simulation model and the environment. These colliders can be defined for all robot links using representations such as boxes, cylinders, spheres, or custom mesh files. In Isaac SIM, colliders can be approximated with the visual meshes of the links, taking the form of bounding boxes or spheres for a straightforward and computationally efficient definition. For greater precision, collision shapes can also be modeled as a convex hull that adapts to the mesh's shape, though this approach may not be ideal for complex geometries like the arms or main body of the evoBOT (see Fig. 4.5a). An alternative method, convex decomposition, approximates the mesh with multiple convex shapes, providing a more accurate representation of intricate geometries but at a higher computational cost. Figure 4.5b illustrates this decomposition, with separate geometric shapes shown in distinct colors and the complete collider represented by green boundary lines.
For cylindrical or spherical shapes, precise representation is achievable in the physics engine, eliminating the need for approximation. Accurate collider representation is crucial for realistic behavior, especially in simulations involving the evoBOT, where the wheels maintain constant ground contact and turntables frequently engage with objects. Therefore, these components' colliders are defined as cylindrical geometries to enhance simulation fidelity. Figure 4.6 compares the two approaches, showing that the wheel approximated with convex decomposition leads to uneven representation, negatively impacting driving behavior and balance control. At the same time, the collider created with a precise



(a) Convex decomposition.



(b) Cylindrical mesh.

Figure 4.6.: **Wheel model** with different approximations of collision shapes.

cylindrical mesh definition optimizes contact with the ground.

4.2.2. Model Optimization using Real-World Data

The following outlines methods to reduce the sim-to-real gap of the evoBOT simulation model. A realistic simulation model aims to improve accuracy and facilitate the sim-to-real transfer of trained models to the real robot. To achieve this, real-world data regarding actuator dynamics, friction, contacts, sensors, and communication delays is collected on the robot and incorporated to optimize the physics simulation model (see Table 4.1).

Actuator Dynamics

Physics simulators like Isaac SIM typically use idealized motor models that assume instantaneous responses, infinite torques, and perfect tracking capabilities. However, to accurately represent a physical robot, it is essential to realistically model the actual motor behavior, which is crucial for reducing the sim-to-real gap. Actuator dynamics, in particular, play a vital role in executing dynamic motions that involve fast accelerations. While technical data sheets provide initial information about motor characteristics, actual performance can vary based on usage time, control specifications, environmental conditions, and other factors. Therefore, collecting real-world data from the robot's physical actuators is essential for minimizing the sim-to-real gap.
Several experiments have been conducted on the real robot to analyze various aspects of actuator dynamics. Specifically, the following factors were investigated:

- Actuator limits (position, velocity, torque)
- Tracking accuracy of the low-level controller
- Verification of the torque-current constant
- Modeling of delays in actuator responses

In particular, actuator limits are assessed to ensure the model accurately reflects the physical capabilities of the actuators. For evoBOT, this includes evaluating the velocity and torque limits of the wheel motors responsible for balancing and locomotion. Next,

Table 4.1.: **System identification techniques** utilized to optimize the simulation model. Based on different real-world measurements collected on the actuators, materials, and sensors.

Category	Collected Real-World Data
Actuator Dynamics	Speed and torque limits, tracking accuracy, response time
Friction	Material properties for static/dynamic friction, restitution
Compliant Contacts	Material properties for compliant stiffness and damping
Sensor Noise	Gaussian approximations for static sensor noise profiles
Latency Model	Communication delay in the sensing and actuation pipeline

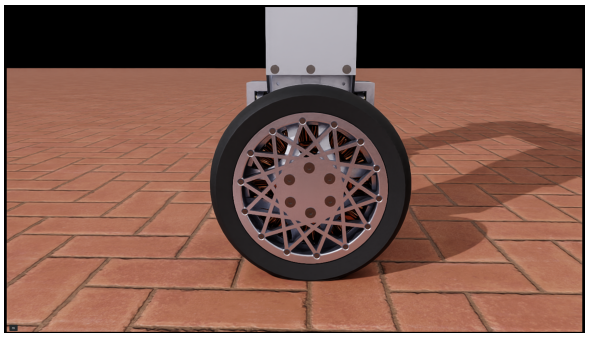
the tracking accuracy of the low-level controller is assessed to confirm that the commands sent to the actuators lead to precise movements. Additionally, the torque-current constant is verified to represent how electrical input translates to mechanical output accurately. Finally, delays in actuator responses are modeled to account for inherent latencies in actuator behavior. These factors ensure that the motor behaviors in the simulation closely mimic real-world performance during dynamic tasks.

Friction

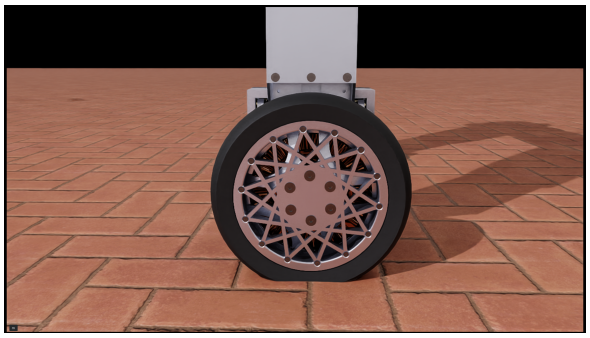
Material properties, particularly frictional forces, significantly impact the interactions when two objects come into contact. In Isaac SIM, these properties can be accurately represented by creating and assigning physics materials to the colliders associated with the robot components. This precise definition of material characteristics is essential for simulating realistic physical interactions in robot simulations. Frictional forces opposing relative motion between objects can be defined using static and dynamic friction parameters within the physics material properties. However, it is essential to note that frictional forces depend on the interaction between the contacting materials, meaning that the friction between rubber and rubber differs from that between rubber and wood.
To account for these interactions, Isaac SIM includes a friction combine mode feature that considers the material properties of both contacting components. This feature outlines the priority order for multiple components and is accessible within the physics material properties. As the evoBOT operates on a single surface type in this study, the combined mode for the tires is set to ensure that the simulated surface material parameters do not influence friction behavior. To realistically model the interaction between the rubber tires and the ground, corresponding parameters for static and dynamic friction and restitution are selected.

Compliant Contacts

Rigid body simulators are often preferred for training control policies in robot learning due to their performance advantages. However, the rigid-body assumption does not apply universally to all robot learning tasks and can vary depending on the type of robot. For instance, in real-world applications of the evoBOT, low tire pressures have been found



(a) Default Rigid-body simulation, resulting in a *contact line* between the wheel and the ground.



(b) Compliant contacts active, resulting in a *contact surface* between the wheel and the ground.

Figure 4.7.: **Compliant contacts** with stiffness and damping variations on the wheel model.

to enhance performance. Specifically, a larger contact surface area helps prevent slippage during highly dynamic tasks, allowing for more significant acceleration changes.

Consequently, modeling this compliance aspect of the wheels becomes essential to minimize the sim-to-real gap. Isaac SIM provides compliant stiffness and damping parameters for rigid body materials to account for compliance in the rigid body simulation, enabling the simulation to avoid complex and computationally intensive soft-body simulations. These parameters simulate material compliance and capture the deformable nature of bodies in contact. For the evoBOT, a spring-damping system is utilized with approximated values for compliant stiffness and damping to replicate the deformation observed in real-world experiments.

Fig. 4.7 shows the effects of compliant contacts emulating high tire pressure (left) and low tire pressure (right) for evoBOT within the simulation. This method offers a more accurate representation of deformable materials' physical interactions and behaviors within the simulation environment, thereby enhancing the realism and precision of the robot learning tasks.

Sensor Noise

Sensor noise is another critical factor in ensuring the accuracy of robotic perception and control. To address this issue, real-world data from the evoBOT's sensors has been collected to evaluate the inherent noise characteristics of the sensor readings. Fig. 4.8 presents the experimental results of these measurements on the robot. By analyzing this data, the statistical properties of the noise are identified and typically modeled using Gaussian approximations (see Table 4.2).

Based on these measurements, realistic sensor profiles can be incorporated into the ground truth data generated from the simulation. Adding these noise characteristics to the simulated sensor data creates a more realistic environment for training and testing control policies. This approach helps bridge the gap between simulation and real-world performance, enabling the robot to better manage uncertainties and variability in its sensory inputs during operation.

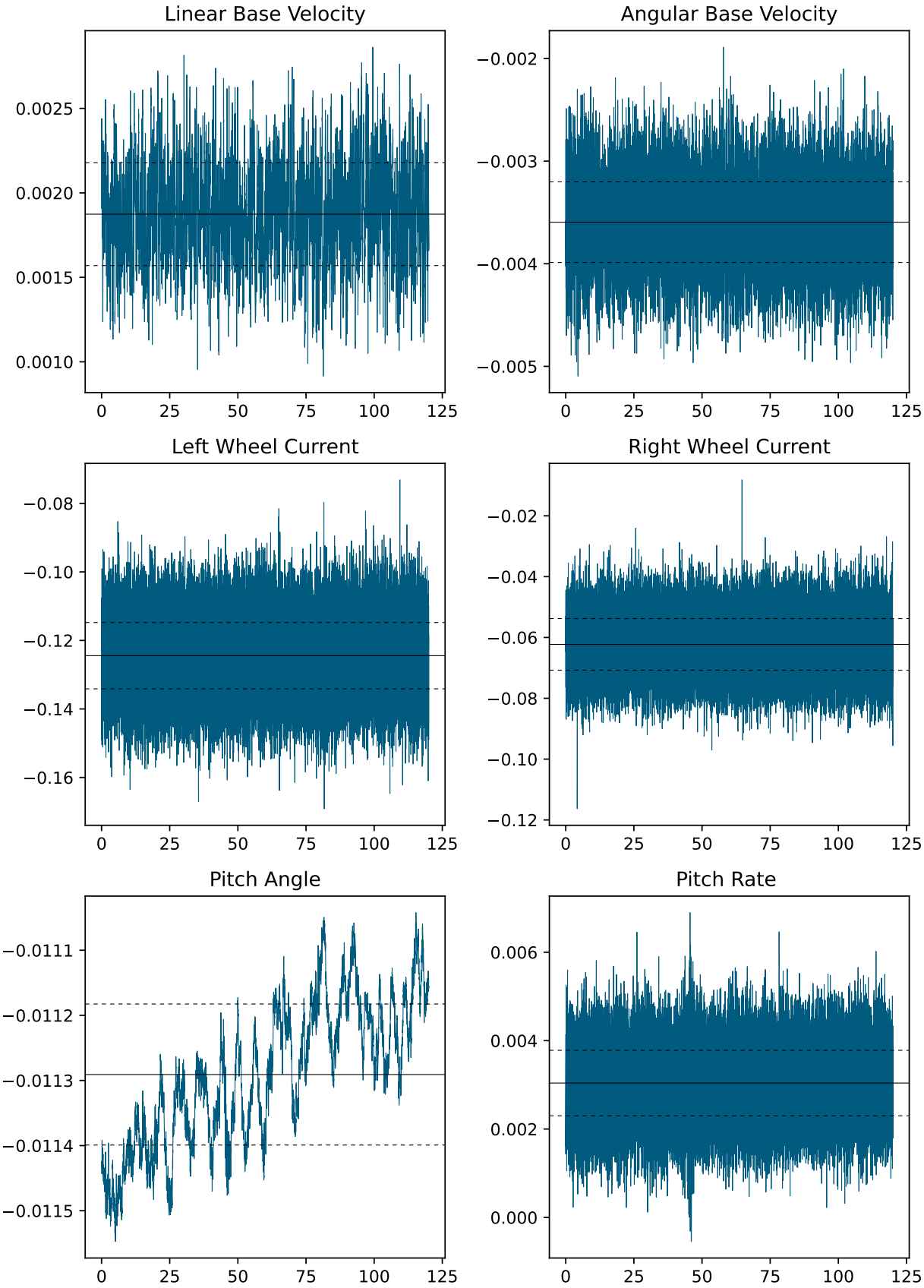


Figure 4.8.: **Noise measurements** from the sensors on the real robot. Gaussian approximations are incorporated into the simulation based on the identified statistical properties (see Table 4.2).

Table 4.2.: **Gaussian approximation** values of the measured sensor noise.

Sensor Data	Mean	Standard Deviation
Linear Base Velocity	1.87×10^{-3}	3.04×10^{-4}
Angular Base Velocity	-3.59×10^{-3}	3.93×10^{-4}
Left Motor Current	-0.12	9.68×10^{-3}
Right Motor Current	-0.062	8.48×10^{-3}
Pitch Angle	-0.011	1.08×10^{-4}
Pitch Rate	3.04×10^{-3}	7.41×10^{-4}

Latency Model

Accurate modeling of delays is crucial in robot control, as it accounts for the time lag between sensory inputs and actuator commands. Experimental analyses have been conducted on these delays in the evoBOT to enhance simulations and reduce the gap between simulated and real-world performance. The experiments focus on two main components: the communication cycle between the sensors and actuators and the slight delays that occur within the actuators as they reach the desired torque levels.
By measuring these delays and applying Gaussian approximations, their statistical properties can be identified and modeled accordingly. Integrating realistic latency effects into the simulation provides a more accurate representation of the timing dynamics involved in robot control. Ultimately, incorporating this latency model improves the robot's performance in real-world scenarios, ensuring effective and timely responses to sensory inputs.

4.2.3. Evaluation of the Sim-to-Real Gap

The following evaluates how closely the simulation model aligns with the behavior of the real robot. An experimental setup utilizing a motion capture system facilitates the collection of ground truth data from real-world robot runs. By benchmarking these real-world runs against identical simulations, the resulting sim-to-real gap can be quantified and analyzed.

Experimental Setup

The PACE Lab, based at Fraunhofer-IML, is a unique research facility specializing in precise positioning within an industrial context. The word PACE stands for Positioning Accuracy Communication Evaluation. The lab includes a high-precision VICON motion capture system that can detect objects in the sub-millimeter range, covering an area of about 1000 square meters (see Fig. 4.9).
The VICON motion capture system generates ground truth data to compare with noisy sensor data. This system employs infrared cameras to identify and track reflective markers placed on the robot. By positioning multiple markers on the robot, VICON achieves tracking with six degrees of freedom (position and orientation) at 250 frames per second. Ten markers are strategically arranged to detect the evoBOT and differentiate it from other

Figure 4.9.: **Experimental setup** at Fraunhofer IML [150]. The PACE Lab is equipped with a large-scale motion capture system to track robots and accurately evaluate their sim-to-real gap.

objects accurately; markers are placed vertically on each wheel axle and along the robot's legs to capture the pitch angle, while the remaining markers are arranged asymmetrically for clear orientation. The collected position data can also be integrated to calculate the robot's velocity.
To evaluate the sim-to-real gap, the experimental setup proceeds as follows: First, the real robot is driven along a specific trajectory using a predetermined set of commands for linear and angular velocity. The recorded data from this run includes internal sensory measurements and external measurements from the VICON system, serving as ground truth data. Next, the same driving commands are applied in the simulation using the same control approach as the real robot. This approach enables observation and quantitative analysis of differences in robot responses to identical trajectories and can be used to assess the accuracy of the physics simulation model.

Benchmarking Real and Simulated Trajectories

To evaluate the accuracy of the simulation model, trajectory profiles of commanded velocities are compared between real and simulated robots. The real robot's response is analyzed against two models: the baseline simulation model (refer to Section 4.2) and the simulation model optimized using real-world data (see Section 4.2.2). This study aims to assess the effects of system identification on the two-wheeled robot. The experiment involves simultaneous commands for linear and angular velocities, achieving a maximum linear velocity of 1.7 m/s and a maximum angular velocity of 1.7 rad/s, alongside rapid acceleration changes.
Figure 4.10 compares simulation results with the real robot's performance concerning commanded velocities, actuator responses, and robot orientation. Findings indicate that

the real robot and the simulation model effectively track commanded velocity profiles, exhibiting similar control characteristics during rapid acceleration changes. Additionally, the physics simulation accurately reflects actuator responses, such as motor current and velocity, underscoring the importance of system identification with real-world motor data. Finally, the experiment offers insights into the real robot's sensor characteristics. While the pitch rate is accurately represented in the simulation, discrepancies in pitch angle estimation on the real robot reveal drifts and offsets. A comparative analysis of pitch angles from the Inertial Measurement Unit (IMU) and VICON systems shows that changes in pitch angle are primarily captured by the IMU data, indicating sensor-related behavior.

Practical Guidelines for Model Optimization

The following provides practical guidelines for reducing the sim-to-real gap in two-wheeled robots. Section 4.2.2 has introduced various system identification methods to improve the simulation model, including the collection of real-world data to accurately model actuator dynamics, friction, contacts, sensors, and delays (refer to Table 4.1).
Based on benchmarking real and simulated trajectories for the evoBOT, we can evaluate how these system identification methods affect the sim-to-real gap. The following guidelines (in descending order of importance) are recommended for collecting real-world data:

1. **Actuator Dynamics:** Accurate modeling of motor characteristics is crucial, especially for dynamic tasks with high accelerations. In addition to assessing low-level tracking accuracy, verify the torque-current constant and realistic torque limits.

2. **Friction:** Precise modeling of static and dynamic friction is essential, particularly for the wheel model of two-wheeled robots. Optimizing friction parameters is also vital for object manipulation, as appropriate contact forces are necessary for effective handling.

3. **Compliant Contacts:** With an accurate friction model, incorporating compliant stiffness and damping parameters can help reduce the sim-to-real gap. Compliant contacts enhance traction during dynamic motions and prevent wheel slippage on the ground.

4. **Sensor Noise:** Experimental results show that Gaussian distributions of sensor noise have a limited impact on the sim-to-real gap for two-wheeled robots. However, accounting for sensor characteristics such as drift or offset can be beneficial.

5. **Delay Model:** Delays in the sensor-actuator communication cycle appear to have a low effect when comparing simulated and real trajectories. While these parameters may not directly influence the sim-to-real gap, they could still be relevant for robot learning and improving generalization capabilities.

4.3. Training Setup

This section outlines the training setup for the learning-based control of evoBOT. It starts with an overview of the simulation framework, parallel training configuration, and the

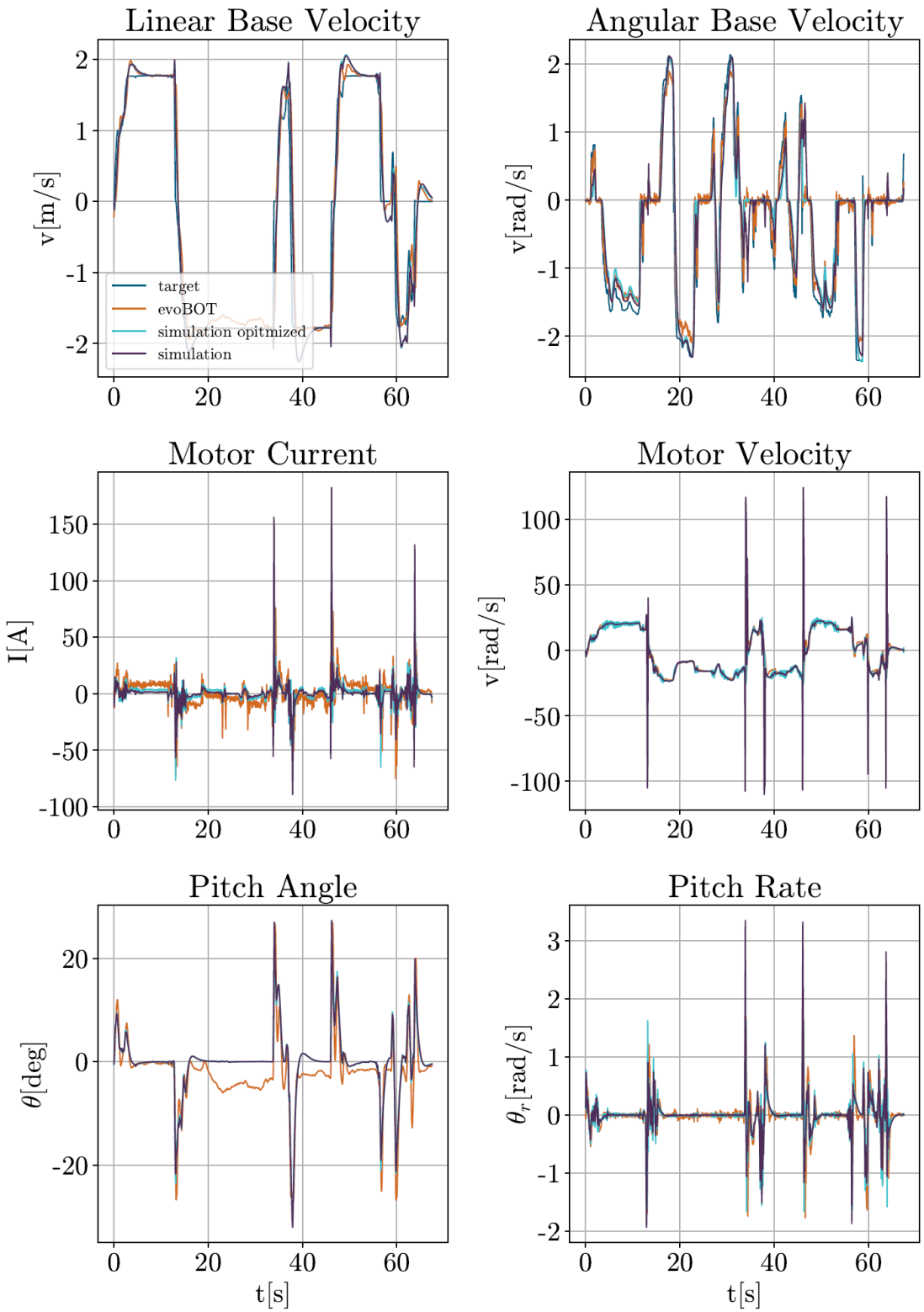


Figure 4.10.: **Sim-to-Real benchmarking** of the simulation model [63]. The resulting trajectories from the simulation and the real robot are shown with the same velocity commands.

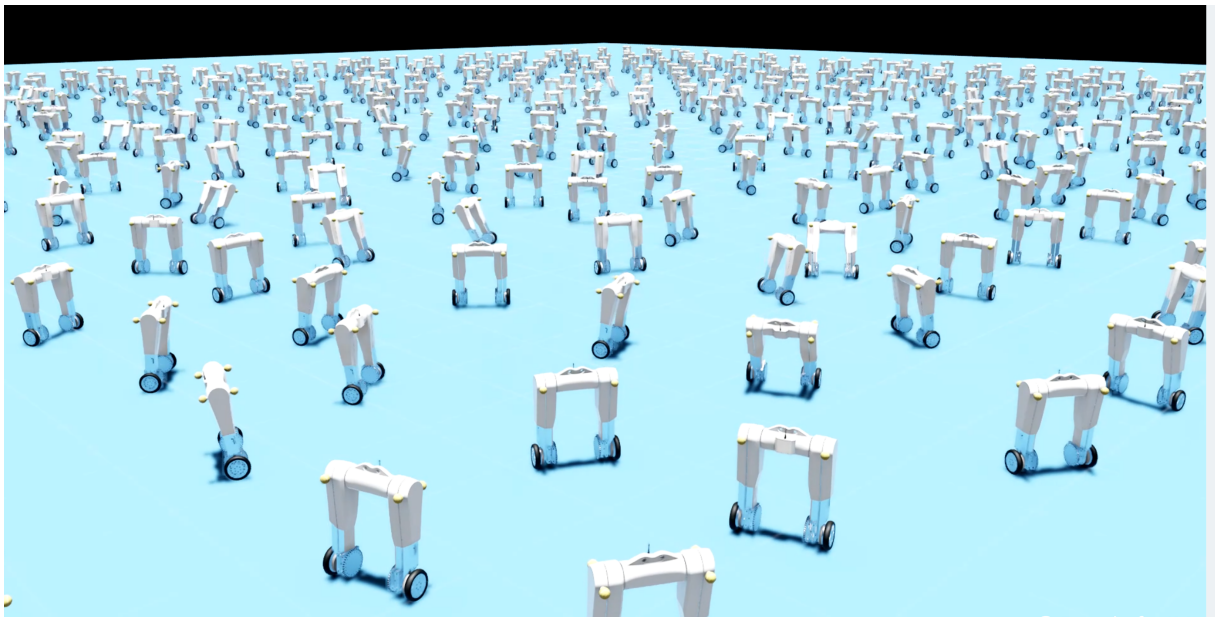


Figure 4.11.: **Parallel training** with 4096 evoBOT instances in Isaac Gym. Experiences from all robots are combined into one neural network for robust control within minutes of training.

learning algorithm. Next, the RL task includes the learning objectives, conditions, and curriculum approach. Finally, the MDP formulation provides a mathematical definition of the observation space, action space, and reward structure.

4.3.1. Parallel Training in Isaac Gym

To train control policies using simulation, Isaac Gym, a GPU-accelerated physics simulation framework designed for robot learning, is employed [123]. Its efficient parallel processing capabilities enable simultaneous experience gathering from thousands of robots (see Fig. 4.11). This parallel training scheme allows for the rapid development of control policies for evoBOT within a couple of minutes on a consumer notebook equipped with an NVIDIA RTX A5000 GPU.

Instanceable Assets

The physics simulation model of evoBOT (see Section 4.2) serves as the baseline for training control policies. To optimize performance in large simulation scenes with multiple clones of the same robot, the USD file is converted into an instanceable asset. USD's scenegraph instancing functionality allows shared mesh assets to be marked as instanceable, enabling each evoBOT instance to reference a single mesh copy. This reduces memory usage in the simulation environment. By correctly structuring the robot hierarchy, multiple evoBOT instances can efficiently share mesh data, facilitating faster and more memory-efficient training of control policies.

Table 4.3.: **Hyperparameters** for the learning algorithm, reward scaling, and task configuration.

Algorithm		Reward		Task	
Horizon	48	w_1	1.0	Linear velocity commands	$\pm$1m/s
Mini-batch size	32768	w_2	0.5	Angular velocity commands	$\pm$1rad/s
Clip range	0.2	w_3	-0.001	Total episode length	10 s
Discount factor γ	0.99	w_4	-0.001	Initial pitch randomization	$\pm$0.4 rad
GAE discount factor λ	0.95	w_5	-0.001	Command velocity interval	4 s
Learning rate α	0.0005			Alternating push interval	6 s

Learning Algorithm

For the learning algorithm, Proximal Policy Optimization (PPO) is utilized [175]. A horizon length of 48 timesteps is chosen to effectively capture short-term dynamics while maintaining computational efficiency, which is critical for balancing a dynamically unstable system. Using five minibatches per epoch with a mini-batch size of 4096 ensures that each policy update benefits from diverse samples, enhancing stability in updates. The discount factor is set to 0.99 to prioritize long-term reward optimization, which is crucial for tasks like velocity tracking.

The Generalized Advantage Estimation (GAE) parameter is set to 0.95 to balance bias and variance, promoting stable training for continuous control of evoBOT. An adaptive learning rate dynamically adjusts the step size for stability during updates, while a KL threshold of 0.008 prevents excessive divergence from the previous policy. These hyperparameters are tailored to the evoBOT task, ensuring the PPO algorithm achieves stable convergence amid dynamic disturbances and payload variability.

Neural Network Structure

Control policies are implemented using fully connected neural networks, specifically Multi-Layer Perceptrons (MLP), featuring two hidden layers with 128 and 64 neurons, respectively, and employing the Exponential Linear Unit (ELU) activation function. The experiences collected from all robot instances are aggregated into a single neural network, benefiting the learning-based controller from diverse training scenarios. This enables the neural network to learn from various states, enhancing the robustness and adaptability of the learning-based controller.

4.3.2. RL Task Definition

A single control policy is trained for robust balancing, dynamic locomotion, and packet handling with the evoBOT. The episode length is set to 10, which improves training convergence compared to longer episodes. Additionally, training with fixed velocity targets per environment results in higher rewards than when variations occur during the episode.

Command Ranges

Control commands for the evoBOT are defined within specific ranges to ensure stability and effectiveness during training. The ranges are as follows:

- Linear velocity commands: $\mathbf{v}_x^{\text{cmd}} \in [-1, 1]\,\text{m/s}$
- Angular velocity commands: $\boldsymbol{\omega}_z^{\text{cmd}} \in [-1, 1]\,\text{rad/s}$
- Arm angle commands: $\mathbf{q}_{\text{arm}}^{\text{cmd}} \in [-135°, 135°]$ (relative to the main body)

This bounded control space prevents overly aggressive movements that could lead to instability, making careful selection of command ranges crucial for achieving the desired balance and performance in various training scenarios.

Reset Conditions

Reset handling terminates episodes when the robot enters an unrecoverable state, providing a structured mechanism to enforce safety constraints and avoid accumulating undesirable behaviors that could destabilize the training process. It is essential to distinguish between resets triggered by failure and those based on episode time-out. Time-out resets are managed separately, where the total reward is replaced by a high negative cost, signaling the agent's failure to maintain task objectives. The following reset conditions are established:

- Pitch angle exceeding $> 60°$ (close to the ground)
- Contact between the gripper and the ground
- Episode time-out, resulting in a negative cost

Curriculum Approach

Incorporating unknown payloads and variable arm states presents significant challenges for achieving stable locomotion, especially during abrupt changes in command velocity or arm movement. A structured reward-based curriculum addresses these complexities, enabling the RL agent to tackle increasing difficulties progressively. This curriculum dynamically adjusts task difficulty based on the agent's performance, measured through cumulative rewards. Rajendran [162] shows that it can allow for faster progression when the agent demonstrates proficiency, aligning task complexity with the agent's learning pace.
Key parameters regulated by the curriculum include payload mass, arm motion range, and command velocity range. Training begins with minimal payloads to support initial learning, progressively increasing the mass to challenge the agent's balance control. The curriculum also starts with limited arm movement, facilitating gradual adaptation to disturbances caused by arm motion. The command velocity range expands after the agent has navigated the entire range of payload masses. Empirical results indicate that this reward-based curriculum leads to improved stability and velocity tracking, as it prevents the agent from encountering overly complex challenges prematurely, thereby enhancing overall learning outcomes.

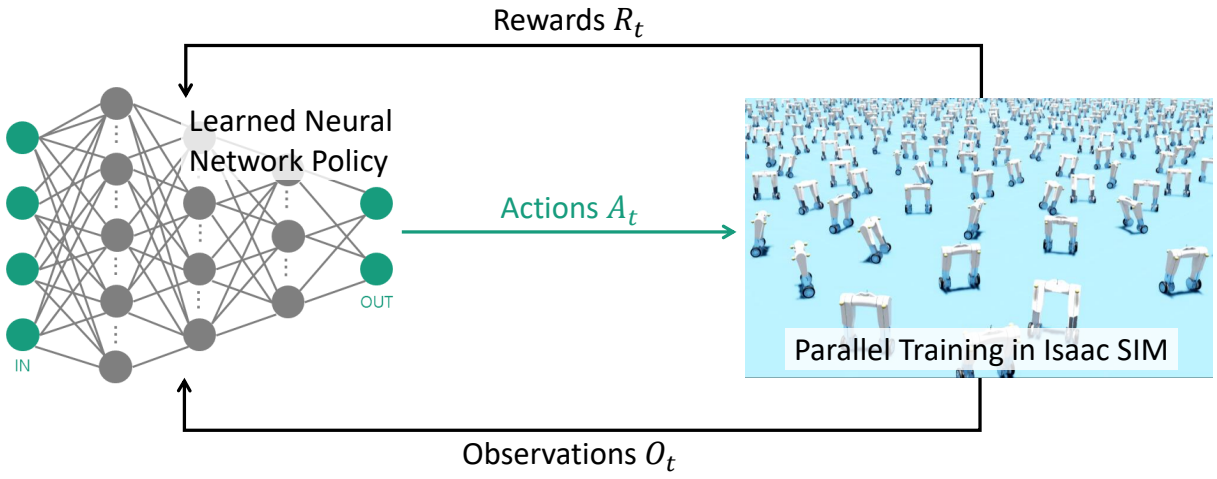


Figure 4.12.: **MDP framework** for learning-based control of evoBOT. The neural network is trained based on observed experiences, collected rewards, and actions taken in the environment.

4.3.3. Mathematical MDP Formulation

The RL task is modeled as a Markov Decision Process (MDP) by defining a tuple of state space $\mathbf{S}$ (referred to as observation space in this thesis), action space $\mathbf{A}$, rewards $r_t(\mathbf{s}_t, \mathbf{s}_{t+1})$, and transitions $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t)$. These components enable the RL agent to learn and make decisions based on maximizing cumulative rewards. The agent acts according to a stochastic policy $\pi(a_t \mid \mathbf{s}_t)$ parametrized by θ. The learning algorithm updates θ to maximize the cumulative discounted rewards $E\left[\sum_{t=k}^{\infty} \gamma^{t-k} r_t\right]$ through interactions with the environment, where k is the current timestep and γ is the discount factor.

Observation Space

The overall observation space $\mathbf{O}$ is defined as:

$$\mathbf{O} = [\mathbf{v}_x, \boldsymbol{\omega}_z, \boldsymbol{\theta}_{\text{base}}, \boldsymbol{\varphi}_{\text{base}}, \mathbf{h}_{\text{arm}}, \mathbf{q}_{\text{arm}}, \dot{\mathbf{q}}_{\text{wheels}}, \dot{\mathbf{q}}_{\text{arm}}, \mathbf{a}_{t-1}, \mathbf{v}_x^{\text{cmd}}, \boldsymbol{\omega}_z^{\text{cmd}}, \mathbf{q}_{\text{arm}}^{\text{cmd}}]. \tag{4.3.1}$$

This representation includes the linear base velocity $\mathbf{v}_x$, angular base velocity $\boldsymbol{\omega}_z$, pitch orientation $\boldsymbol{\theta}_{\text{base}}$, roll orientation $\boldsymbol{\varphi}_{\text{base}}$, gripper position $\mathbf{h}_{\text{arm}}$, arm joint position $\mathbf{q}_{\text{arm}}$, arm joint velocity $\dot{\mathbf{q}}_{\text{arm}}$, wheel joint velocities $\dot{\mathbf{q}}_{\text{wheels}}$, previous actions $\mathbf{a}_{t-1}$, and the command velocities $\mathbf{v}_x^{\text{cmd}}$, $\boldsymbol{\omega}_z^{\text{cmd}}$, and the arm command $\mathbf{q}_{\text{arm}}^{\text{cmd}}$.
This formulation provides the policy with sufficient task-relevant information, enabling it to anticipate the effects of arm motion and improve overall stability and velocity tracking. To enhance training efficiency, all observations are scaled to consistent dimensions, clipped within the range of ± 5, and the pitch orientation is normalized using the 2-argument arctangent function to ensure a continuous representation of state changes.

Action Space

The RL action space defines the range of control inputs available to the agent for influencing the environment. An appropriate action space is essential for effective whole-body locomotion and balance control, closely tailored to specific task requirements. The action

space of the policy is defined by the control inputs to the wheel joints $\mathbf{A} \in \mathbb{R}^2$, which can be configured as either wheel joint torque or wheel joint velocity.
In Isaac Gym [123], actions generated at each pre-physics step are normalized such that $a \in [-1, 1]$, which are subsequently scaled by the maximum torque or velocity limits before being applied as control targets for the wheel joints. A torque-based action space allows the RL agent to control wheel torques directly. The torque control can be formulated as:

$$\boldsymbol{\tau} = K_p(\mathbf{a}_{\text{scaled}} - \dot{\mathbf{q}}), \tag{4.3.2}$$

where $\mathbf{a}_{\text{scaled}} \in [-20, 20]\,\text{rad/s}$ represents target wheel velocities. Explicit velocity control can also be achieved using a Proportional Controller (P-controller) that computes wheel torques based on the scaled RL actions. Velocity-based control may offer smoother transitions between commands and reduce system sensitivity to rapid input changes. In contrast, torque control can be implemented more straightforwardly, requiring only scaled actions without additional damping parameters. Torque control provides the agent refined control over joint accelerations, allowing for improved responsiveness to dynamic disturbances such as arm movement and payload-induced shifts.

Reward Structure

The reward formulation includes terms that track the command velocities, while regularization terms promote smooth control.
The total reward r_{total} is calculated as:

$$\mathbf{r}_{\text{total}} = \mathbf{r}_{\text{lin,x}} w_1 + \mathbf{r}_{\text{ang,z}} w_2 - \mathbf{r}_{\text{acc}} w_3 - \mathbf{r}_{\text{tor}} w_4 - \mathbf{r}_{\text{pow}} w_5. \tag{4.3.3}$$

In this formulation, $\mathbf{r}_{\text{lin,x}}$ represents the linear velocity tracking reward, encouraging the agent to minimize the error between the commanded linear velocity $\mathbf{v}_x^{\text{cmd}}$ and the actual base linear velocity $\mathbf{v}_x$. Similarly, $\mathbf{r}_{\text{ang,z}}$ promotes accurate tracking of the commanded yaw rate $\boldsymbol{\omega}_z^{\text{cmd}}$ relative to the actual yaw rate $\boldsymbol{\omega}_z$. The joint acceleration penalty $\mathbf{r}_{\text{acc}} w_3$ discourages excessive acceleration and promotes smoother control by penalizing rapid changes in joint velocities. The joint torque penalty $\mathbf{r}_{\text{tor}}$ aims to minimize abrupt torque changes, ensuring gradual adjustments in wheel torques for stability. The energy penalty $\mathbf{r}_{\text{pow}}$ discourages high power consumption by penalizing excessive torque and velocity products $\boldsymbol{\tau} \cdot \dot{\mathbf{q}}$, thereby promoting energy-efficient operation.
The maximum reward values for $\mathbf{r}_{\text{lin,x}}$ and $\mathbf{r}_{\text{ang,z}}$ are set at 1 and 0.5, respectively, ensuring that the agent adjusts precisely near target velocities and discourages significant corrections. Adaptive factors scale penalties based on tracking errors, guiding the robot to focus on restoring balance while discouraging abrupt movements during large deviations. The weights for regularization penalties are tuned to maintain their impact relative to primary rewards (see Table 4.3). This ensures that acceleration and torque penalties are effective without overwhelming the agent's learning process. This structured approach enhances the agent's ability to track velocities while accurately maintaining stability in dynamic environments.

4.4. Simulation Experiments

This section examines the proposed learning-based control approach for evoBOT. It starts with a performance analysis regarding velocity tracking and actuator responses. Next, it reviews relevant benchmarking studies to evaluate the training framework's efficiency and effectiveness. Finally, the learning-based controller is tested in unseen scenarios to assess its generalization capabilities to new tasks.

4.4.1. Task Performance Analysis

Based on the described training setup (see Section 4.3), a learning-based controller is trained for evoBOT. This policy is trained for various task variants, including robust balancing, dynamic locomotion, whole-body locomotion, and packet handling (see Fig. 4.13).

Performance Metrics

To evaluate the overall performance of the trained control policy, the following metrics are utilized:

$$\begin{aligned} \text{Survival} &= \Sigma_{\text{finishes}}/\Sigma_{\text{resets}}, \\ \Delta\text{Lin Vel} &= |\mathbf{v}_x^{\text{cmd}} - \mathbf{v}_x|, \\ \Delta\text{Ang Vel} &= |\boldsymbol{\omega}_z^{\text{cmd}} - \boldsymbol{\omega}_z|, \\ \Delta\text{Actions} &= |\mathbf{a}_t - \mathbf{a}_{t-1}|, \end{aligned} \tag{4.4.1}$$

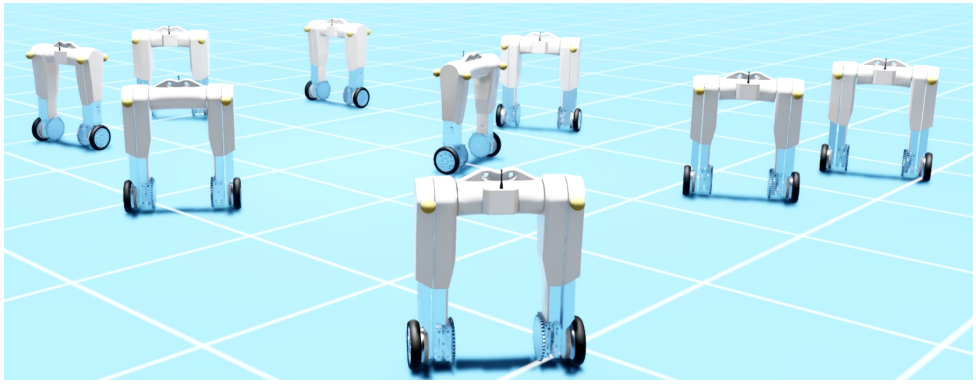
where Survival indicates the overall success rate of the policy in terms of finished episodes, ΔLin Vel and ΔAng Vel account for the relative linear and angular velocity tracking errors, and ΔActions represents instantaneous changes in commanded actions.

Velocity Tracking

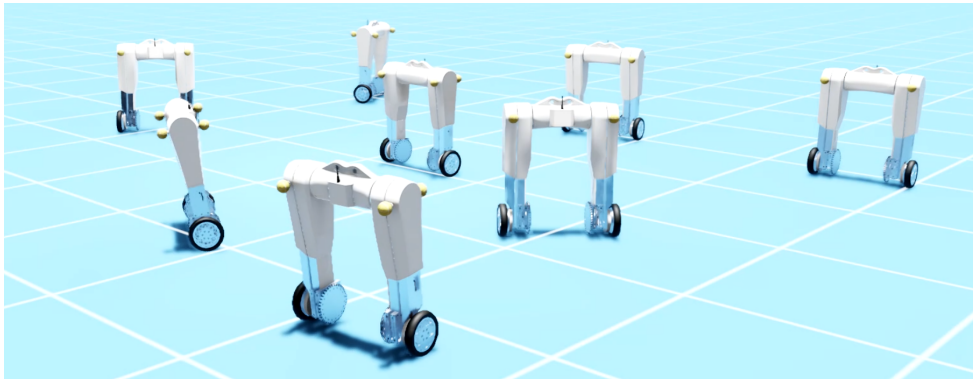
Fig. 4.14 shows the resulting tracking performance of a trained policy for evoBOT. The experiment contains several phases with increased levels of dynamics. In phase I (0 to 2 s), the policy is requested to balance itself from a difficult initial pitch position of up to 30 degrees. Phase II (2 to 6 s) consists of alternating linear velocity commands of $\pm$0.5m/s, while turn rates of up to $\pm$1rad/s are requested. In phase III (6 to 10 s), both linear and angular velocity commands change instantaneously, before returning to dynamic balancing again.

Actuator Responses

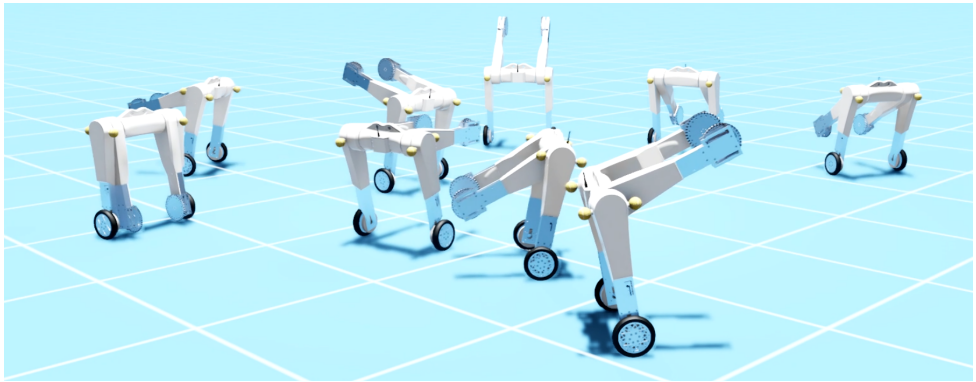
The results indicate that the policy learns to respond to the given velocity targets dynamically. With average tracking errors of 0.13m/s and 0.05rad/s for linear and angular velocity tracking, respectively, the system can operate at the physical boundaries of the system. On the one hand, to realize such rapid changes in linear velocity, the robot first needs to accelerate in the opposite direction to create the necessary tilting moment. On the other hand, the policy learns to exploit the limits of the physically constrained robot given by the motor dynamics to reach both maximum speed and acceleration limits.



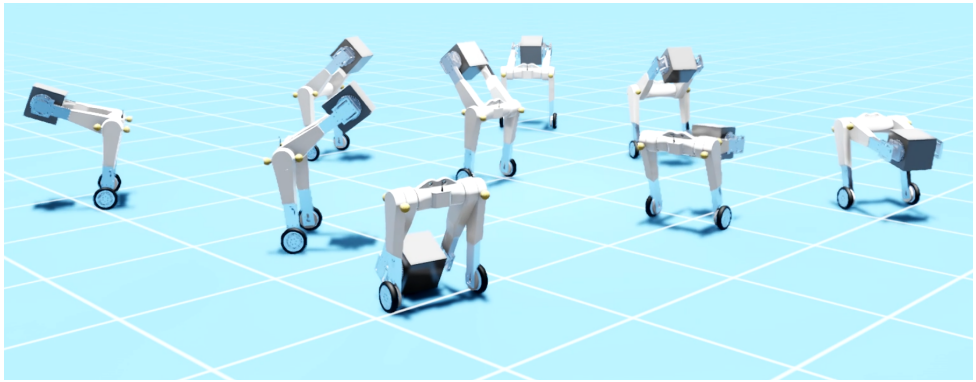
(a) Balancing Task.



(b) Locomotion Task.



(c) Whole-Body Locomotion Task.



(d) Packet Handling Task.

Figure 4.13.: **Task variants** for learning-based control with evoBOT. A single control policy is trained for all aspects simultaneously, leading to fast iteration cycles in robot development.

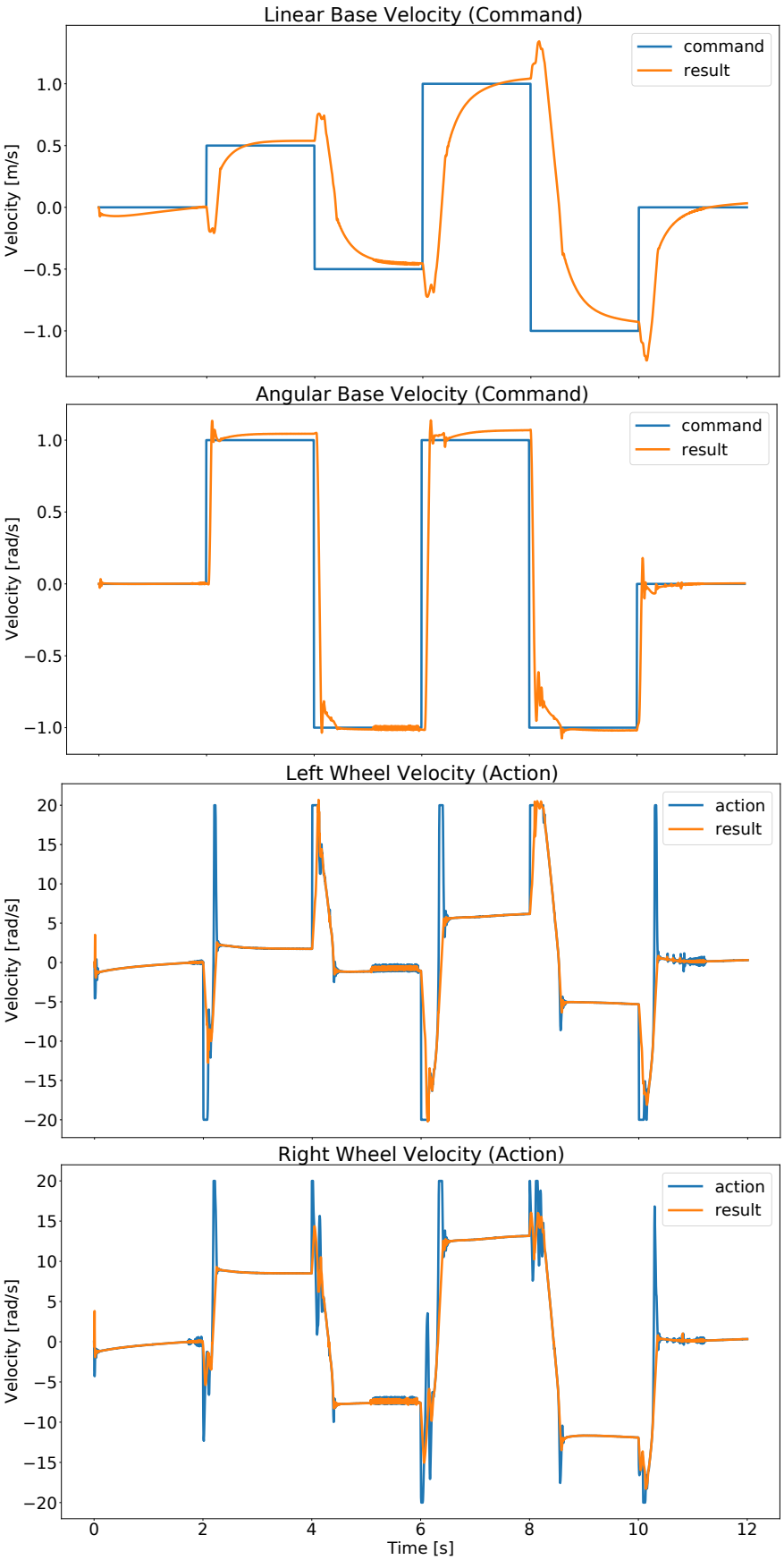


Figure 4.14.: **Performance analysis** of the learning-based controller. Dynamic responses to linear and angular velocity commands and actions sent to the actuators are shown (top down).

4.4.2. Benchmarking RL Design Choices

The following evaluates the RL framework developed for the evoBOT task, specifically benchmarking the design choices of the learning-based controller in terms of training efficiency and effectiveness. Key aspects analyzed include the dense reward formulation, the parallelized training setup, and the curriculum approach outlined in Section 4.3.

Reward Design

The dense reward formulation for the evoBOT task integrates regularization terms with primary task rewards. This design promotes smoother control and efficient exploration during training. The reward structure includes penalties for excessive acceleration, abrupt torque changes, high energy consumption, and rewards for achieving target velocities. This combined approach encourages the agent to prioritize stability while maintaining performance. Including regularization terms resulted in significantly higher performance and faster training times than using plain task rewards alone. Agents exhibited improved exploration capabilities, enabling them to adapt more readily to dynamic environments and achieve better overall task performance.

Parallel Learning

The parallelized training setup utilizes Isaac Gym to train control policies across multiple instances of evoBOT simultaneously. This strategy accelerates training by allowing the agent to gather diverse experiences concurrently. In this setup, 4096 evoBOT instances are trained in parallel, facilitating rapid experience collection and policy updates. The shared learning across instances enables the neural network to learn from a broader range of scenarios. Training with 4096 robots resulted in significantly faster training times than configurations using 512, 1024, and 2096 environments. The enhanced parallel processing capability directly correlates with an increased convergence rate of the policies, facilitating robust control development in a fraction of the time.

Curriculum Learning

The curriculum learning approach is designed to progressively increase task difficulty, which benefits the evoBOT's dynamic balance and locomotion tasks. By systematically introducing more complex challenges, the agent develops its skills in a structured manner. The curriculum adjusts key parameters such as command velocities, arm motion ranges, and payload weights throughout the training process. This gradual escalation helps the agent build a strong foundation before tackling more demanding scenarios. Implementing curriculum learning resulted in faster training and higher final performance on challenging tasks. The agent effectively adapted to increased velocities, arm angles, and packet weights, demonstrating improved stability and control as it advanced through the curriculum stages.

Neural Network Structure

This analysis compares different neural network structures used during training (see Section 4.3) to determine if the learned policies can be deployed on a small robot micro-

Table 4.4.: **Neural network structures** and their influence on the overall policy performance.

Neurons	Weights	Survival ↑	Δ Lin Vel ↓	Δ Ang Vel ↓	Δ Actions ↓
$512 \times 256 \times 128$	169216	99.6 ± 0.8	0.33 ± 0.09	0.02 ± 0.0	4.18 ± 1.08
$256 \times 128 \times 64$	43648	99.2 ± 1.6	0.43 ± 0.13	0.14 ± 0.17	5.71 ± 2.2
$128 \times 64 \times 32$	11584	98.8 ± 3.43	0.36 ± 0.07	0.05 ± 0.18	3.7 ± 2.11
$64 \times 32 \times 16$	3232	97.6 ± 4.8	0.44 ± 0.09	0.15 ± 0.01	2.6 ± 1.46
64×32	864	98.2 ± 3.12	0.55 ± 0.13	0.22 ± 0.18	1.6 ± 1.1

controller, thereby reducing sensor and actuator communication delays. Various fully connected neural networks with three hidden layers were evaluated, featuring neuron configurations of $2^n[64, 32, 16]$ where $n \in [0, 3]$, as well as a two-layered network with $[64, 32]$ neurons, as shown in Table 4.4. The performance of the trained policies for balancing success and dynamic locomotion is consistent across all three-layered networks with $n \geq 2$. In contrast, smaller networks exhibit increased survival rates or tracking errors, likely due to their inability to accurately represent the nonlinearities in highly dynamic motions.

Control Frequencies

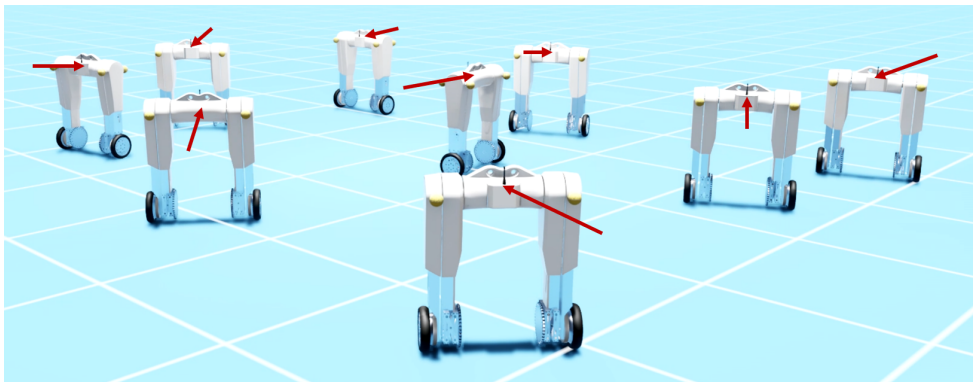
Different control frequencies are analyzed to determine suitable inference rates for the neural network during highly dynamic motions. A range of control frequencies from 25 Hz to 500 Hz was examined. It was found that dynamic balancing requires control frequencies of at least 100 Hz, with optimal performance achieved at 200 Hz. This requirement arises because the highly dynamic nature of the learning task necessitates a minimum update rate for stabilization. Conversely, control frequencies of 500 Hz or higher can lead to minimal state changes, potentially overwhelming the stochastic nature of the learning process.

4.4.3. Robustness Tests on Unseen Scenarios

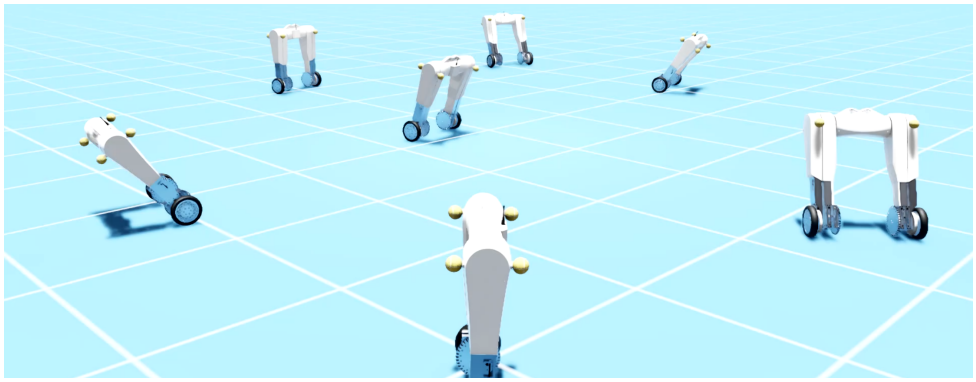
This section demonstrates the robustness of the trained policy for the evoBOT, mainly focusing on configurations not encountered during training. The experiments evaluate the policy's adaptability to new tasks and variants, encompassing various challenging scenarios. The robustness tests include random external pushes, high velocities, heavy payloads, unknown terrains, and communication delays, with visualizations provided in Fig. 4.15.

External Forces

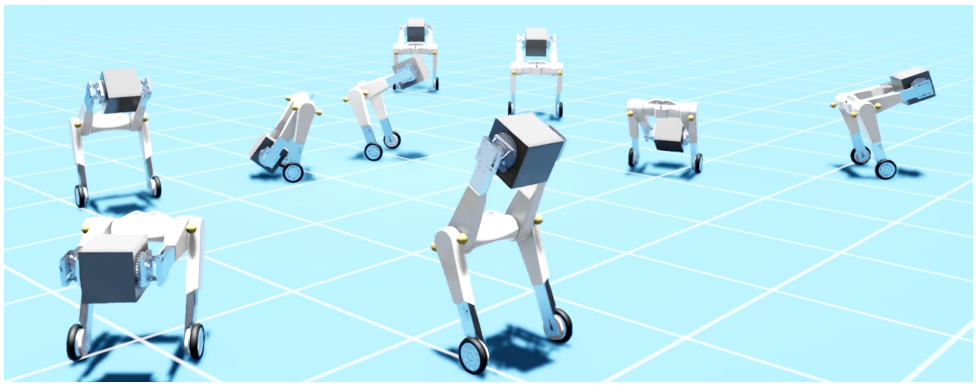
The policy's resilience to disturbances is assessed through experiments involving random external pushes, both linear and angular. These pushes simulate unpredictable environmental interactions that the robot might encounter in real-world scenarios. The trained policy maintained stability and balance during these disturbances, demonstrating effective control and recovery strategies. This robustness is crucial for operating in dynamic environments where external forces vary significantly.



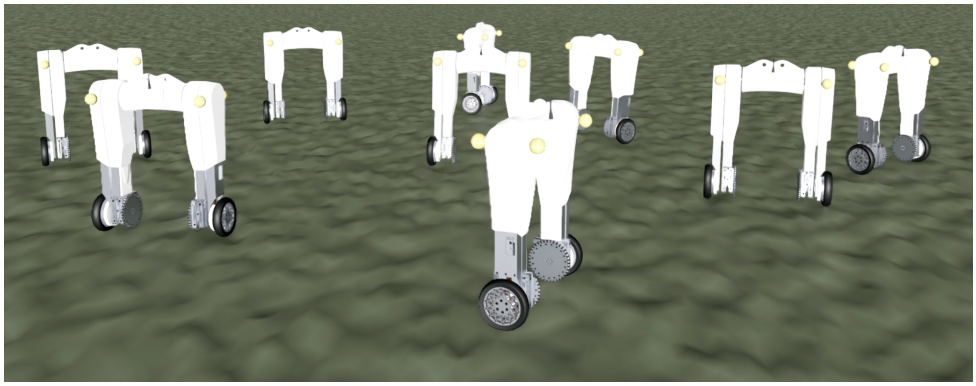
(a) External Forces.



(b) Higher Velocities.



(c) Heavier Payloads.



(d) Unknown Terrains.

Figure 4.15.: **Robustness tests** of the control policy on unseen scenarios. Various challenging scenarios demonstrate the adaptability and generalization of the trained policy to new tasks.

Higher Velocities

The trained policy is further assessed under increased velocity conditions, where the corresponding commands exceed the trained maximum of 1 m/s. This test examines the policy's ability to adapt to rapid movements and maintain control under high-speed conditions. Results indicate that the policy performed well, effectively managing balance and locomotion even at elevated speeds. This adaptability is essential for tasks that require quick maneuvers and responsive control.

Heavier Payloads

To assess the policy's robustness in handling increased weight, experiments were conducted with a payload of 40 kg, double the training payload of 20 kg. This scenario tests the policy's stability and control under heavier loads. The trained policy exhibited strong performance, effectively managing the additional weight without significantly degrading balance or locomotion capabilities. This demonstrates the policy's adaptability to varying operational conditions.

Unknown Terrains

The policy's performance on unknown terrains, specifically rough and stony surfaces, was also evaluated. These tests simulate real-world conditions where the robot might encounter unpredictable ground characteristics. The trained policy successfully navigated these challenging surfaces, showcasing its robustness and ability to adapt to different terrain types. This capability is vital for applications in diverse environments with varying terrain conditions.

Communication Delays

Finally, the impact of communication delays on the policy's performance was examined, simulating delays of up to 50 ms, typical in 5G communication scenarios [199]. These delays can affect the responsiveness of the control system, making it critical to evaluate the policy's robustness under such conditions. The trained policy demonstrated resilience to these communication delays, maintaining effective control and stability despite the latency. This adaptability is essential for applications that rely on real-time communication in dynamic environments.

4.5. Sim-to-Real Transfer

This section outlines the approach and results for transferring policies trained in simulation to the real robot. It begins with a discussion of domain randomization techniques to enhance the generalization of the policies for real-world applications. An ablation study is presented to benchmark the impact of these techniques on overall policy performance. Next, a deployment pipeline is introduced, designed for resource-constrained hardware to facilitate real-time control applications. Finally, the performance of the learning-based controller is evaluated, focusing on both velocity and torque control on the real robot.

4.5.1. Domain Randomization

Introducing variability in simulation parameters enhances the generalization of trained policies, making them more robust for real-world deployment. This section outlines several approaches that expose learning algorithms to a broader range of scenarios, improving their adaptability to unforeseen conditions. These approaches include dynamics randomization, noise injection, delay randomization, and external disturbances. Additionally, an ablation study is conducted to evaluate the impact of these techniques on overall policy performance.

Dynamics Randomization

To address discrepancies in dynamics modeling, dynamics randomization is implemented to enhance the simulation model. Variations in robot link masses are introduced, adjusted by $\pm 20\%$ to account for unmodeled cabling and small electronic components. Additionally, Gaussian noise is applied to the gravity vector, permitting variations of up to $+5\%$. For motor dynamics, both the stiffness and damping of the implicit PD controllers are randomized by $\pm 50\%$, while the identified acceleration ramp is subject to an uncertainty of $\pm 20\%$. Physical ground parameters such as static and dynamic friction and restitution are randomized within $[0.5, 1.0]$ and $[0.0, 0.5]$, respectively, to develop robust policies adaptable to various floor types.

Noise Injection

To enhance the robustness of the policy, noise is introduced to all observations before they are presented to the learning algorithm. This noise is derived from real-world sensor data measurements (see Fig. 4.8), with different scalings of the measured noise variances applied in each environment. This noise injection is essential for training policies that execute highly dynamic motions. Given the high control frequencies, minor state changes occur relative to the injected noise, necessitating that the policy learns to distinguish between noise and actual state changes that demand rapid action adjustments.

Delay Randomization

To account for varying delays experienced in real robot deployments, measurements of different delay ranges are taken from the robot. During simulation training, a buffer of the last observations is maintained, and data is randomly sampled from this buffer for policy training. The findings indicate that feasible delay ranges can extend to $10\,\mathrm{ms}$, aligning with the measured delays in real-world scenarios. This approach ensures the policy can effectively handle the latency inherent in sensor data communication.

External Disturbances

External disturbances are incorporated into the training process to enhance the robustness of the policies in dynamic environments. Randomized pushes are introduced, applying linear and angular forces to the robot base. This approach allows the robot to learn how to dynamically recover from challenging initial poses, fostering more robust behaviors and improving its adaptability to unpredictable conditions.

Table 4.5.: **Sim-to-real methods** and their influence on the overall policy performance.

Sim-to-Real Methods	**Survival ↑**	**Δ Lin Vel ↓**	**Δ Ang Vel ↓**	**Δ Actions ↓**
Baseline (none)	100.0 ± 0.0	0.26 ± 0.09	0.06 ± 0.01	2.67 ± 0.6
Random Pushes	98.33 ± 2.36	0.24 ± 0.02	0.05 ± 0.0	2.39 ± 2.24
Dynamics Randomization	98.67 ± 1.89	0.22 ± 0.02	0.07 ± 0.02	5.32 ± 2.15
Noise Injection	99.33 ± 0.94	0.28 ± 0.03	0.05 ± 0.0	4.45 ± 2.99
Delay Randomization	85.0 ± 19.1	0.62 ± 0.33	0.1 ± 0.04	0.59 ± 0.29

Sim-to-Real Ablation

An ablation study is conducted to assess the impact of domain randomization techniques on the sim-to-real gap. Policies are iteratively trained by adding different components aimed at sim-to-real transfer, with performance evaluated over 1000 trials for each configuration (see Table 4.5). Five training runs with varying random seeds are analyzed regarding survival rates, action rates, and velocity tracking accuracy. The results, summarized in Table 4.5, indicate that the best policy performance is achieved without any sim-to-real methods activated (baseline). Nevertheless, policies with activated random pushes, dynamics randomization, and noise injection also demonstrate strong performance. In contrast, benchmarking policies with delayed observations reveals a significant increase in task difficulty, likely due to the rapid state changes required for effective control in highly dynamic scenarios.

4.5.2. Deployment Pipeline

To transfer trained policies from simulation to the real evoBOT, the neural networks must be deployed on hardware. This section provides an overview of the available hardware options for evoBOT, the conversion of the learning-based controller, and its integration onto real-time hardware.

Embedded Hardware

The evoBOT includes several key electronic components essential for its functionality. A microcontroller is responsible for real-time control and motor actuation, enabling quick responses during dynamic motions. In contrast, an embedded computer manages higher-level processing and decision-making tasks. Additionally, the robot features an IMU, which provides real-time data critical for motion tracking and stabilization. The trained policies are implemented directly on the microcontroller, allowing for real-time control and rapid responses by facilitating direct connections to sensor data and motor actuation.

Neural Network Conversion

The firmware, developed in MATLAB/Simulink, generates C code for deployment on the robot's microcontroller. Since MATLAB/Simulink does not support direct import

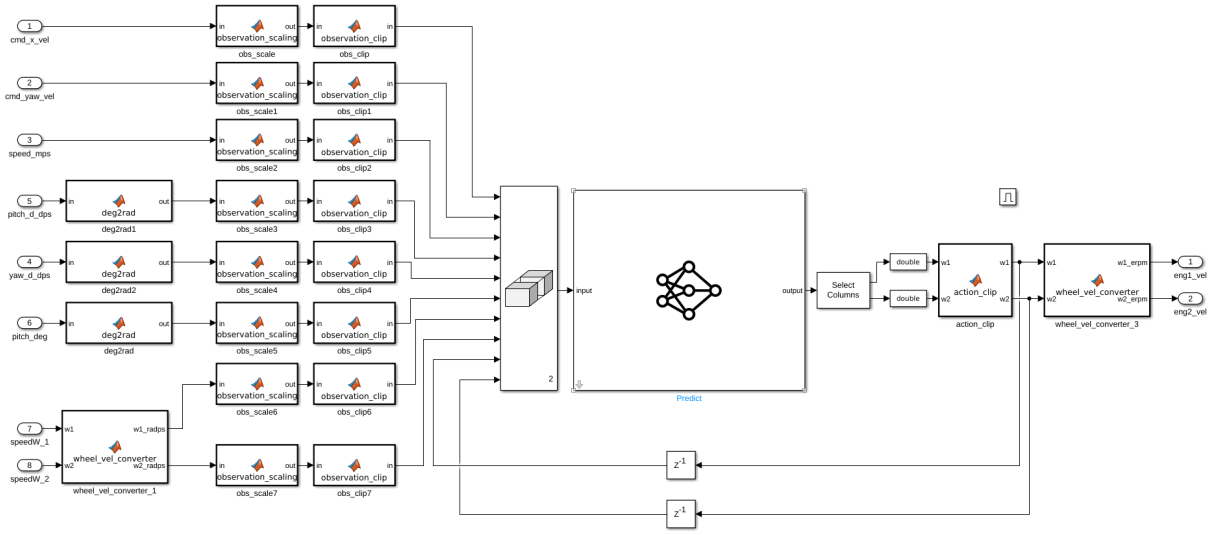


Figure 4.16.: **Deployment pipeline** for the trained neural network via MATLAB/Simulink. The generated C code is deployed directly on the microcontroller of evoBOT for real-time inference.

of PyTorch files for neural networks, the policy is first converted to the Open Neural Network Exchange (ONNX) format. The converted ONNX model is then imported into MATLAB/Simulink and integrated using the predict block for inference. The neural network is linked to the relevant sensor inputs and actuator outputs, incorporating scaling commands that align with the training setup (see Fig. 4.16).

Firmware Integration

The trained neural network is integrated at the firmware level, where sensor processing and communication occur. This integration allows the generated C code, including the neural network, to be deployed on the microcontroller. Although microcontrollers typically offer faster communication with sensors and actuators, they are also resource-constrained compared to embedded computers. While the real-time inference speed is sufficient, the static memory allocated on the microcontroller limits the neural network size. Therefore, a neural network with two hidden layers is utilized, enabling real-time integration of the learning-based controller onto the robot's embedded hardware.

4.5.3. Performance Evaluation

The following evaluates the performance of the trained policies deployed on the real evoBOT, based on the described sim-to-real methods and the deployment pipeline for the trained neural networks. It begins with an overview of the experimental setup, followed by an analysis of the policies operating at both velocity and torque control levels.

Experimental Setup

A two-layered neural network is chosen for the sim-to-real transfer to accommodate the size constraints of the microcontroller, based on findings from evaluation studies. The

policies are trained and deployed at an inference rate of 200 Hz, while low-level motor control operates at 1 kHz. The sim-to-real ablation study (see Table 4.5) indicates that the learned control policies are highly sensitive to delays due to rapid state changes. Consequently, control policies utilize all proposed sim-to-real methods, excluding delay randomization.

Velocity Control

Initially, a learning-based controller that operates at the motor speed level is deployed on the real robot. In this configuration, the motor velocity commands generated by the trained neural network are converted to torques using a low-level PD controller. This initial transfer of trained policies effectively enables the dynamic stabilization of the robot. An analysis of the robot's base velocities shows that the average velocities achieved closely match the commanded target velocities for locomotion tasks. However, higher action rates are observed in the real system compared to the simulation. Further analysis reveals that the tracking accuracy of the low-level PD controllers is less effective for small velocity commands, contributing to a more significant sim-to-real gap. This hardware limitation mainly affects the balancing task, leading to increased energy consumption despite initial attempts to replicate this behavior in simulation to reduce the gap.

Torque Control

Next, a learning-based controller that operates directly at the torque level is deployed on the real robot. This approach simplifies the interface, as the neural network can directly produce torques sent to the motors, eliminating the need for a dedicated PD controller and addressing the actuator tracking accuracy limitations discussed earlier. Like the velocity control case, an analysis of the robot's base velocities indicates that the average velocities align closely with the commanded target velocities for locomotion tasks. However, the torque interface significantly enhances the performance of the balancing task, improving both smoothness and energy efficiency. Real-world experiments demonstrate a high robustness of the policies against external disturbances, such as random pushes, confirming the successful transfer of behaviors from simulation to the real robot.

4.6. Concluding Remarks

This chapter presented a comprehensive case study on learning-based control methods for the evoBOT, a dynamic two-wheeled robot developed by Fraunhofer IML. A detailed physics simulation model was established to accurately capture the kinematic and dynamic aspects of the evoBOT, serving as a foundation for effective policy training. Real-world data was utilized to optimize the sim-to-real gap, incorporating techniques such as actuator dynamics modeling, friction analysis, and noise injection. The performance analysis revealed that the trained policies achieved high velocity tracking and actuator responsiveness levels, effectively stabilizing the robot during various dynamic tasks. Insights from evaluating the Guided RL design choices indicated that dense reward structures, parallel learning, and task curricula significantly improved training efficiency and policy

effectiveness. Robustness tests demonstrated the policies' adaptability to unseen scenarios, including handling external disturbances, navigating unknown terrains, and managing higher payloads. Finally, the sim-to-real pipeline illustrated the successful transfer of trained policies to the real robot, including an initial setup for real-time critical control applications. Key results from this chapter, including the simulation model and training code, have been released as open-source and are available in the latest release of NVIDIA Isaac SIM. Building on these findings, the next chapter will explore the role of action spaces and their impact on dynamic control tasks, further investigating how a generalized action space can enhance the performance and adaptability of robotic systems.

5

Generalized Action Space for Robot Control

The case study on learning-based control (see Chapter 4) demonstrates that dynamic robot learning tasks are significantly influenced by the control commands a neural network can direct to a real robot, commonly referred to as the action space in RL. Typically, robots operate within an action space defined by physics. For instance, electric actuators utilize current to track torque through a high-frequency control loop. While RL policies can directly output torque values, many research papers use alternative action spaces such as joint position, joint velocity, or task-space setpoints. These values are then converted into torques using feedback laws. However, practitioners often rely on intuition when selecting action spaces in RL. Therefore, it would be beneficial for the community to have a method for identifying the most efficient and effective action space for their robotic tasks. Consequently, this chapter addresses the following research question:

Research Question III: How does action space design affect the performance of learning-based control strategies across various robot morphologies and learning tasks?

This chapter introduces the concept of a generalized action space for robot control, aiming to create a comprehensive framework that addresses the complexities of robot interactions within dynamic environments. It begins with a general mathematical formulation, presenting a generalized parameterization of action space at the torque level, covering specific cases of the parameterization, and the importance of temporal considerations in action space design (Section 5.1). Next, the experimental setup for the benchmarking study is described, which includes various robot platforms and tasks, the policy training methodology, and strategies for sim-to-real transfer (Section 5.2). Following up, the results of this benchmarking study are discussed, focusing on applying the proposed action space to a range of robots, detailing initial exploration behaviors and the effects of varying control frequencies (Section 5.3). Finally, the chapter discusses the implications of these findings and offers practical guidelines for the robot learning community when designing action spaces for new robotic tasks (Section 5.4).

Disclaimer: Parts of this chapter are based on a prior publication of this thesis [60].

5.1. Mathematical Formulation

This section introduces the concept of a generalized action space for robot control, aiming to provide a unified approach for evaluating action space design in RL at the torque level. First, the motivation and mathematical formalization of the generalized parametrization of action spaces are presented. Next, several specific cases of this generalized parametrization are discussed, including well-known action spaces such as position, velocity, and torque control. Finally, the temporal aspects of action space design in RL are addressed, emphasizing the importance of control frequencies for robot learning tasks.

5.1.1. Generalized Parameterization of Action Space

Practitioners must thoughtfully choose the action space where learning occurs, often relying on intuition to guide their decisions. For example, a wheeled robot is typically associated with a wheel velocity action space. In contrast, a legged robot is linked to joint positions, and a manipulator operates within Cartesian space targets. In well-established tasks, the field has often converged on standardized action spaces. For instance, position control action spaces are commonly used for learning legged locomotion.

Decomposition in Policy and Controller

In robot motor control, the design of the action space in RL is generally divided into two components: the policy network and a low-level controller (see Fig. 5.1:A). In this setup, the actions generated by the policy network are translated into torques using a PD Controller. These feedback laws must apply to simulation environments and real robot operations to effectively control the robot's movements.

Universal Transformation into Motor Torques

In a generalized form for RL, the transformation $\mathbf{T}$ from the policy output $\mathbf{a}_t$, state $\mathbf{s}_t$, and historical information $\mathbf{h}_t := \sum_{k=[0..t-1]} \mathbf{a}_k$ into motor torque commands τ_t can be expressed as:

$$\boldsymbol{\tau_t} = \mathbf{T}(\mathbf{a}_t, \mathbf{s}_t, \mathbf{h}_t) \tag{5.1.1}$$

This formulation includes both task-space action spaces and configuration-based action spaces. Given its significance in sim-to-real RL for dynamic motions, this study focuses on configuration-based action spaces. When the command update and torque update operate at the same frequency (i.e. $f^{\text{Ctrl}} = f^{\text{RL}}$, see Section 5.1.3), it becomes possible to combine them to express any of these action spaces as a linear mapping from the state and action to torque.

Parametrization for Configuration-based Tasks

This leads to the following general parametrization for configuration-based action spaces, which represents a specific instance of the formulation described above:

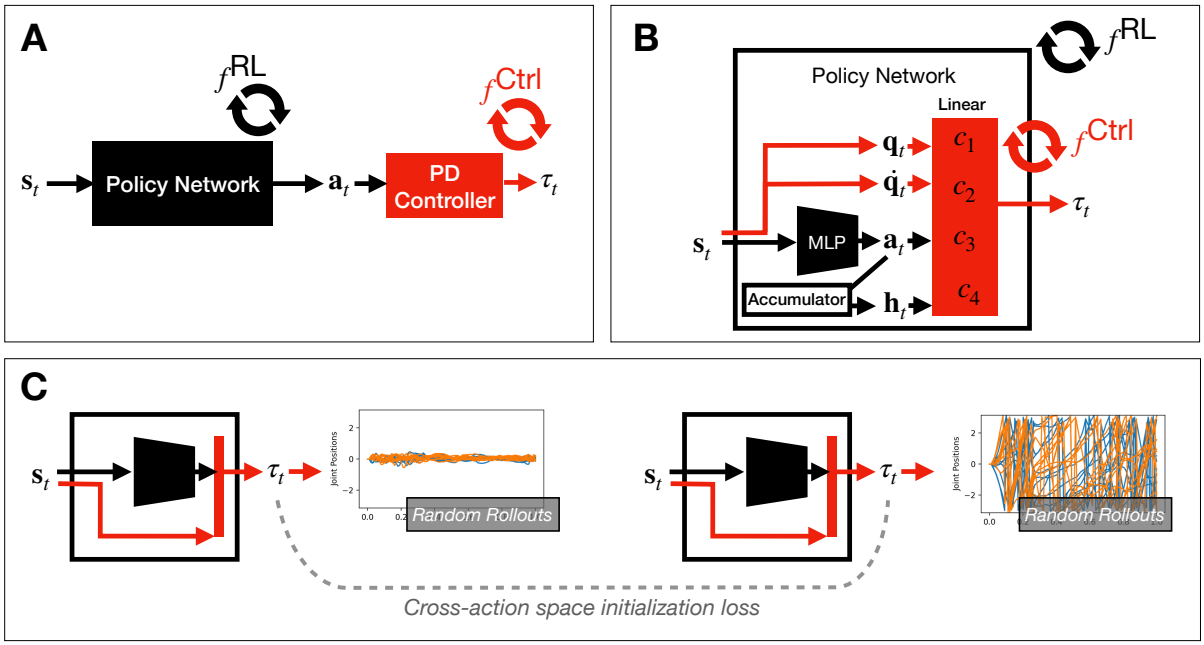


Figure 5.1.: **Generalized action space** for robot control tasks (based on prior publication [60]). **A.** Classical decomposition of policy and controller. **B.** Generalized parameterization of action spaces, where the low-level controller can be represented as a linear layer within the network. **C.** Seamless transitions between action spaces and novel analysis of their structural differences.

$$\boldsymbol{\tau}_t = \mathbf{T}_\mathbf{J}(\mathbf{a}_t, \mathbf{s}_t, \mathbf{h}_t), \tag{5.1.2}$$

$$\mathbf{T}_\mathbf{J}(\mathbf{a}_t, \mathbf{s}_t, \mathbf{h}_t) = c_1\mathbf{q}_t + c_2\dot{\mathbf{q}}_t + c_3\mathbf{a}_t + c_4\mathbf{h}_t. \tag{5.1.3}$$

This set of parameters c_1, c_2, c_3, c_4 can be interpreted as the final linear layer $\phi_A(\cdot)$ of the policy network $\pi(s_t)$ (see Fig. 5.1:B). In particular, these coefficients for $\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{a}_t, \mathbf{h}_t$ represent linear combinations of joint positions, velocities, actions, and historical information, respectively, when the learning frequency f^{RL} matches the control frequency f^{Ctrl}. The coefficients c_1, c_2, c_3, c_4 serve to linearly map these variables to the torque command τ_t. The choice among different action spaces corresponds to manually initializing these policy weights. This interpretation allows us to evaluate whether the advantages of an action space are related to the architecture itself, meaning that allowing the policy to learn parameters leads to better performance, or whether they are linked to initialization, suggesting that appropriate initialization of the network parameters improves performance. Furthermore, it allows for pre-training one action space to enhance initial exploration in another action space, such as for smooth sim-to-real transfer (see Fig. 5.1:C).

5.1.2. Special Cases of the Generalized Parametrization

Table 5.1 presents several action spaces in this generalized parameterization. Each action space is defined as a linear function of the state and action, which is independent of each joint. The command update connects the policy output $\mathbf{a}_t$ to the desired state, which can

be ($\mathbf{q}_t^{\text{des}}$, $\dot{\mathbf{q}}_t^{\text{des}}$, or $\boldsymbol{\tau}_t$). In contrast, the torque update relates the command to the desired state.

Formalization of Common Action Spaces

Common action spaces in RL for robotics include joint position, joint velocity, and torque control (see Chapter 2). In position action spaces, the command update directly translates the policy actions into the desired joint positions. The torque update in this scenario converts the desired state into the motor's torque target, utilizing the stiffness and damping coefficients, respectively. The network outputs correspond to the desired joint velocities for velocity action spaces. In this case, the torque update function acts as a speed-tracking controller that generates the appropriate motor torques. No torque update is required when using torque control because the policy outputs directly represent the desired motor torques.

Formalization of Delta Action Spaces

In addition to the typical action spaces, some studies implement delta variants for controlling position and velocity. The policy output is incorporated as an offset in the command update in these delta action spaces. The network's action can either be applied as a delta to the previous desired state (e.g., $\mathbf{q}_{t-1}^{\text{des}}$), which represents the cumulative sum of all previous actions $\mathbf{h}_t$ (referred to as multi-step integrator, or MS), or as delta to the current state (known as a one-step integrator, or OS).
As a result, the torque updates for delta position control are equivalent to those for position control. However, the command update varies depending on whether the action is integrated as an offset to the previous desired joint positions (MS) or the current state (OS). Similarly, torque updates for delta velocity and velocity action spaces are the same. Still, the command updates for these cases also involve integrating action offsets to the desired joint velocities (MS) or the current state (OS).

Relationships between Action Spaces

Under this generalized parameterization, several interesting relationships between the action spaces can be observed. First, it is evident that RL policies operating closer to the motor control level require fewer parameters to tune, as the torque update rules become simpler. For example, in the case of torque control, the policy network is directly mapped to the torques. Conversely, additional parameters are needed for torque updates in joint position action spaces.
Furthermore, delta action spaces facilitate an implicit implementation of higher-order action spaces. For instance, delta position control action spaces present another method for effective velocity control. In contrast, delta velocity control pertains to acceleration control, operating at a dynamic level similar to direct torque control.
Interestingly, some relationships may be counterintuitive. For example, delta position control with a one-step integrator is equivalent to velocity control, while delta velocity control with a one-step integrator is akin to torque control. Consequently, the generalized

Table 5.1.: **Special cases of the generalized parametrization** of action spaces (based on prior publication [60]). The corresponding command and torque updates, including the generalized parameter set, are provided for each action space. MS = Multi-step, OS = One-step integrator.

Action Space	**Command Update**	**Torque Update**	$[\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}, \mathbf{c_4}]$
Position	$\mathbf{q}_t^{\text{des}} \leftarrow \mathbf{a}_t$	$\boldsymbol{\tau}_t \leftarrow K_p(\mathbf{q}_t^{\text{des}} - \mathbf{q}_t) - K_d\dot{\mathbf{q}}_t$	[-K_p,-K_d,K_p,0]
Δ Position (MS)	$\mathbf{q}_t^{\text{des}} \leftarrow \mathbf{a}_t + \mathbf{q}_{t-1}^{\text{des}}$	$\boldsymbol{\tau}_t \leftarrow K_p(\mathbf{q}_t^{\text{des}} - \mathbf{q}_t) - K_d\dot{\mathbf{q}}_t$	[-K_p,-K_d,0,K_p]
Δ Position (OS)	$\mathbf{q}_t^{\text{des}} \leftarrow \mathbf{a}_t + \mathbf{q}_t$	$\boldsymbol{\tau}_t \leftarrow K_p(\mathbf{q}_t^{\text{des}} - \mathbf{q}_t) - K_d\dot{\mathbf{q}}_t$	[0, -K_d, K_p, 0]
Velocity	$\dot{\mathbf{q}}_t^{\text{des}} \leftarrow \mathbf{a}_t$	$\boldsymbol{\tau}_t \leftarrow K_d(\dot{\mathbf{q}}_t^{\text{des}} - \dot{\mathbf{q}}_t)$	[0, -K_d, K_d, 0]
Δ Velocity (MS)	$\dot{\mathbf{q}}_t^{\text{des}} \leftarrow \mathbf{a}_t + \dot{\mathbf{q}}_{t-1}^{\text{des}}$	$\boldsymbol{\tau}_t \leftarrow K_d(\dot{\mathbf{q}}_t^{\text{des}} - \dot{\mathbf{q}}_t)$	[0, -K_d, 0, K_d]
Δ Velocity (OS)	$\dot{\mathbf{q}}_t^{\text{des}} \leftarrow \mathbf{a}_t + \dot{\mathbf{q}}_t$	$\boldsymbol{\tau}_t \leftarrow K_d(\dot{\mathbf{q}}_t^{\text{des}} - \dot{\mathbf{q}}_t)$	$[0, 0, K_d, 0]$
Torque	$\boldsymbol{\tau}_t \leftarrow \mathbf{a}_t$	–	$[0, 0, 1, 0]$

action space allows for a novel analysis of structural differences and seamless transitions between them.

5.1.3. Temporal Aspects of Action Space Design

The generalized action space for robot control suggests that all action spaces can be represented as a linear function applied to the state and policy output (see Fig. 5.1). However, this representation does not fully account for the policy's behavior between timesteps, mainly when the linear function is executed at a higher frequency than the policy evaluation, which is common in real robot systems. To address this, a higher-frequency module is sometimes used to bridge the gap between simulation and reality.
For example, an actuator with a high gear ratio may demonstrate a misalignment between the torques in the real world and those in simulation. When torque control is not critical, a well-tuned low-level position controller can help mitigate the need for precise torque modeling in simulations. However, in scenarios where force plays a vital role, such as locomotion and contact-rich manipulation, successful implementations aim to minimize the torque sim-to-real gap by utilizing quasi-direct drive motors that are easier to model, either analytically or through learning-based system identification.
The benchmarking study of action spaces (see Section 5.3) focuses on these force-intensive tasks and examines whether high-frequency low-level control provides advantages when actuation torque is accurately modeled. The following descriptions detail each relevant frequency, including the physics, control, and learning frequency.

Physics Frequency

In physics simulations, including those used for robotics, a physics frequency denoted as f^{Phys} approximates the physics stepping. This frequency is relatively insignificant for real robots since physical interactions occur continuously at an infinitely high rate. Because the physics frequency is highly dependent on the simulator's physics engine, it was not examined in detail during the benchmarking study. Instead, the physics frequency is set

as high as possible while balancing it with computational efficiency.

Control Frequency

The control frequency, represented as f^{Ctrl}, influences the updates for torques (see Table 5.1). A lower control frequency limits the variety of action sequences the policy can execute by determining how often actions can change. On the other hand, it can also enhance stability during the learning process, as shorter trajectories simplify the task of credit attribution. Various control frequencies are tested in the benchmarking process, while maintaining consistent practical discount factors and batch sizes.

Learning Frequency

The learning frequency, denoted as f^{RL}, refers to the rate at which command updates occur (see Table 5.1). The same action is generally maintained for several timesteps when this frequency is lower than the control frequency. This approach helps to shorten trajectories, facilitates credit attribution, and accelerates computational learning by decreasing the number of policy inferences required. In such cases, torque may be updated multiple times between policy evaluations since it depends on the state and is recalculated at the control frequency. High frame skipping is often justified in physical systems because the feedback law for the action space computes quickly and relies solely on the local state of each joint. This allows for high-frequency implementation without communication overhead between the motors and a central processor.

5.2. Experimental Setup

This section outlines the experimental setup for the benchmarking study, which is based on the generalized parameterization of action spaces (refer to Section 5.1). It specifically details the policy training configuration and provides an overview of the robot platforms and tasks involved in the study (see Fig. 5.2 and Section 5.3).

5.2.1. Policy Training Setup

The PPO algorithm is utilized for training the RL policies, which is recognized for its effectiveness in sim-to-real motor control [174, 81]. Detailed policy training and network architecture information can be found in the appendix. Training is conducted on five continuous control tasks, including three from the OmniIsaacGymEnvs suite (Ball Balance, Cartpole, and Franka Cabinet) [123], as well as two custom locomotion tasks for the evoBOT and Go1 robots. The primary focus is on the evoBOT and Go1, where the reward structure incorporates velocity tracking and regularization terms. Training parameters and reward formulations are adapted from previous studies [127, 99], ensuring that all terms remain independent of action representations for a fair comparison. The simulation is conducted using NVIDIA Isaac Sim.

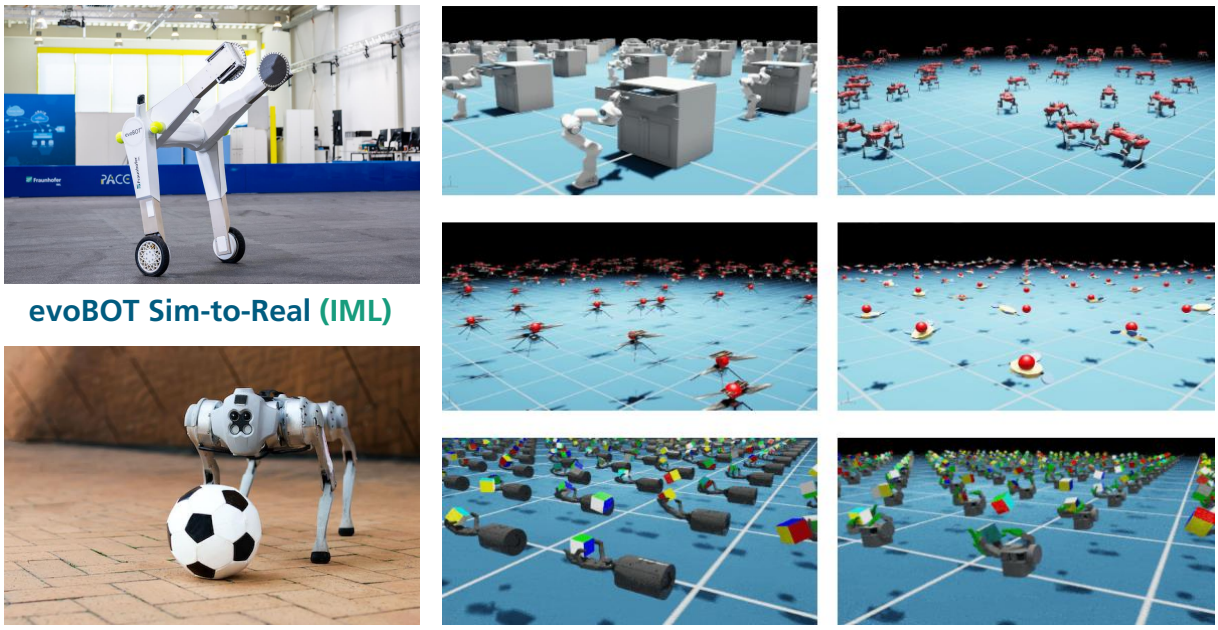


Figure 5.2.: **Overview of robot learning tasks** for the benchmarking study on action spaces. A subset of the Omniverse Isaac Gym suite [74] and the evoBOT [99] and Go1 [73] platforms are analyzed to guide practitioners in selecting the appropriate action space for new tasks.

5.2.2. Robot Platforms & Tasks

The benchmarking study includes two real robots to evaluate the choice of action spaces: the Unitree Go1 and the evoBOT (see Fig. 5.2). The Unitree Go1 is a quadrupedal robot equipped with twelve electric actuators. It stands 40 cm tall and weights 12 kg. The evoBOT is a two-wheeled robot based on the principle of a compound inverted pendulum. It features eight electric actuators, stands 80 cm tall (excluding the arms), and has a total weight of 50 kg. While the study incorporates robot learning tasks from Omniverse Isaac Gym, the primary focus of the experiments is on these two robot platforms.

5.2.3. Sim-to-Real Transfer

Domain randomization techniques are utilized to transfer trained policies from simulation to the real world. Gaussian noise is added to the sensor observations to account for uncertainty in the robot's sensing and actuation, and a lag model is implemented to simulate processing delays. Additionally, ground friction is randomized to ensure the robot can adapt to varied terrains and disturbances, and random pushes are applied to the robot during training. Detailed information about the domain randomization parameters and their ranges for each task is provided in the appendix.

5.3. Benchmarking Study

This section presents a study on the generalized action space for robot control. The study includes experiments designed to evaluate performance within this action space. Additionally, it examines how these action spaces enhance performance, focusing on initial exploration behavior, expressive capacity, and the impact of control frequency selection. Most of the experiments are conducted in simulation to facilitate scaling and reproducibility. To ensure the environment accurately reflects a simulation-to-real scenario, the highest-performing learned policies were tested on the Go1 and evoBOT robots in the real world.

5.3.1. Performance Evaluation

The performance analysis develops control policies for the Go1 and evoBOT robots using 100 different parameter sets within the generalized action space. The primary goal of this experiment is to analyze trends in action space parameters for the two robots, each exhibiting distinct dynamics, rather than to identify a single optimal action space for each task. To accomplish this, the investigation does not center on employing the most efficient search techniques, such as Bayesian optimization. Instead, the objective is to explore various action spaces to reveal trends. Consequently, a grid search approach is conducted, with parameter ranges selected to encompass the settings from OmniIsaacGymEnvs, representing the optimized default action spaces for comparable robots.

Trends Across the Generalized Action Space

Figure 5.3 presents the results from the sweep over the generalized action space in the evoBOT and Go1 locomotion tasks. Each entry in the table corresponds to a unique combination of generalized parameters, representing a specific action space. The color gradient, ranging from green (highest performance) to red (lowest performance), visualizes the normalized return across three seeds for a policy trained in each action space.
The color gradient indicates that the optimal action space varies significantly depending on the learning task. Even for the two locomotion tasks, the Go1 and evoBOT demonstrate nearly opposite trends in how changes to action space parameters affect performance. For the evoBOT locomotion task, position control action spaces (found in the upper part of the table) perform poorly, while the best-performing action spaces for the Go1 task are located within these same spaces. In contrast, the evoBOT performs optimally in torque and velocity control action spaces, where the Go1's performance declines.

Selection of Action Space Parameters

The learning dynamics are highly sensitive to the selection of action space parameters. For the evoBOT, introducing a nonzero value for c_1 (related to a position control loop) significantly degrades performance, as does increasing c_2 (which pertains to the damping parameter). Conversely, the Go1 robot struggles to achieve high rewards without a sufficiently elevated c_1. This variation arises from the distinct natural dynamics of each robot. The Go1 benefits from a tendency to maintain a nominal standing position, as effectively compensating for gravity is a key aspect of its task. Its optimal strategy

Table 5.2.: **Performance evaluation of the evoBOT task** for various special cases of the generalized action space. The normalized return, linear and angular velocity errors, and the corresponding parameter set are shown for the best-performing action space (see Fig. 5.3).

Action Space	**Normalized Return**	**Δ Lin Vel**	**Δ Ang Vel**	**[c_1, c_2, c_3, c_4]**
Position	0.54	0.51	0.43	[-10, -0.5, 30, 0]
ΔPosition (MS)	0.81	0.17	0.37	[-10, -2, 0, 3]
ΔPosition (OS)	0.64	0.30	0.75	[-10, 0, 0, 3]
Velocity	0.99	0.05	0.02	[0, -1, 30, 0]
ΔVelocity (MS)	1.0	0.03	0.03	[0, -2, 0, 1]
ΔVelocity (OS)	0.91	0.11	0.09	[0, 0, 30, 0]
Torque	1.0	0.04	0.02	[0, 0, 10, 0]

involves oscillating its joints around these nominal positions. On the other hand, the evoBOT requires continuous wheel rotation for effective locomotion; consequently, the bias introduced by a nonzero c_1 can negatively impact its performance.

Task-Related Performance Metrics

In addition to the trends observed in the generalized action space regarding normalized return, utilizing task-specific metrics for performance assessment is beneficial. For locomotion tasks, evaluating a policy's ability to follow a specified velocity target involves examining the tracking errors in both linear and angular velocities.

Table 5.2 presents these task-related performance metrics for selected special cases within the sweep of the generalized action space (see Fig. 5.3). Specifically, the best-performing action spaces for each special case (as shown in Table 5.1) have been selected for further evaluation. The results indicate that the poor performance observed in the position and delta position action spaces is also reflected in the velocity tracking performance. However, integrating a damping term, as seen in the delta position multi-step case, results in linear velocity errors of less than 20 cm/s. In contrast, significantly better performance is observed from the velocity, delta velocity multi-step, and torque action spaces, yielding optimal velocity tracking results of under 5 cm/s and 3 rad/s, respectively.

5.3.2. Initial Exploration Behavior

The evaluation of the generalized action space has revealed that the performance of a policy can be highly sensitive to the selection of action space parameters. One possible reason for the significant influence of action space design on training performance is that it enhances initialization behavior, facilitating more effective exploration during the training process. For example, Hwangbo et al. found that using a position control action space leads to effective standing behavior. At the same time, directly commanding torques makes the robot more prone to wriggling and falling during quadrupedal locomotion tasks.

	c_3=	10	15	20	30
c_1=-40	c_2=-3.0	0.47±0.00	0.50±0.01	0.53±0.00	0.54±0.01
c_1=-40	c_2=-2.0	0.48±0.00	0.50±0.00	0.53±0.00	0.56±0.01
c_1=-40	c_2=-1.0	0.49±0.00	0.51±0.01	0.56±0.01	0.68±0.01
c_1=-40	c_2=-0.5	0.51±0.01	0.53±0.00	0.63±0.02	0.61±0.01
c_1=-40	c_2=0.0	0.47±0.01	0.50±0.02	0.49±0.02	0.51±0.02
c_1=-30	c_2=-3.0	0.49±0.01	0.51±0.01	0.53±0.01	0.54±0.00
c_1=-30	c_2=-2.0	0.50±0.00	0.53±0.00	0.55±0.01	0.57±0.01
c_1=-30	c_2=-1.0	0.51±0.00	0.55±0.00	0.57±0.01	0.62±0.02
c_1=-30	c_2=-0.5	0.51±0.01	0.57±0.01	0.59±0.01	0.60±0.01
c_1=-30	c_2=0.0	0.49±0.00	0.50±0.02	0.48±0.02	0.52±0.01
c_1=-20	c_2=-3.0	0.51±0.01	0.53±0.00	0.54±0.00	0.56±0.00
c_1=-20	c_2=-2.0	0.51±0.00	0.53±0.01	0.54±0.01	0.59±0.01
c_1=-20	c_2=-1.0	0.52±0.01	0.56±0.01	0.58±0.01	0.63±0.00
c_1=-20	c_2=-0.5	0.53±0.00	0.57±0.00	0.59±0.01	0.62±0.00
c_1=-20	c_2=0.0	0.53±0.00	0.59±0.02	0.59±0.02	0.59±0.01
c_1=-10	c_2=-3.0	0.54±0.01	0.55±0.00	0.55±0.00	0.57±0.01
c_1=-10	c_2=-2.0	0.53±0.01	0.54±0.00	0.54±0.00	0.59±0.01
c_1=-10	c_2=-1.0	0.52±0.00	0.54±0.01	0.56±0.02	0.64±0.01
c_1=-10	c_2=-0.5	0.53±0.01	0.57±0.00	0.63±0.00	0.65±0.01
c_1=-10	c_2=0.0	0.55±0.00	0.60±0.01	0.62±0.01	0.63±0.02
c_1=0	c_2=-3.0	0.71±0.01	0.78±0.00	0.84±0.01	0.94±0.00
c_1=0	c_2=-2.0	0.76±0.02	0.88±0.01	0.94±0.02	0.97±0.03
c_1=0	c_2=-1.0	0.94±0.01	0.99±0.00	1.00±0.00	0.99±0.01
c_1=0	c_2=-0.5	1.00±0.00	0.99±0.01	1.00±0.00	0.97±0.03
c_1=0	c_2=0.0	0.99±0.01	1.00±0.00	0.99±0.01	0.98±0.01

(a) evoBOT

	c_3=	10	15	20	30
c_1=-40	c_2=-3.0	0.22±0.01	0.26±0.01	0.39±0.00	0.36±0.01
c_1=-40	c_2=-2.0	0.54±0.03	0.62±0.01	0.64±0.01	0.58±0.02
c_1=-40	c_2=-1.0	0.91±0.00	0.96±0.00	0.96±0.02	0.85±0.01
c_1=-40	c_2=-0.5	0.94±0.01	0.99±0.01	0.84±0.02	0.71±0.02
c_1=-40	c_2=0.0	0.91±0.01	0.84±0.02	0.46±0.06	0.27±0.05
c_1=-30	c_2=-3.0	0.17±0.01	0.27±0.01	0.39±0.00	0.43±0.09
c_1=-30	c_2=-2.0	0.55±0.01	0.64±0.00	0.64±0.00	0.57±0.07
c_1=-30	c_2=-1.0	0.96±0.01	0.99±0.01	0.97±0.01	0.80±0.03
c_1=-30	c_2=-0.5	0.98±0.01	0.97±0.01	0.77±0.02	0.66±0.05
c_1=-30	c_2=0.0	0.94±0.00	0.76±0.05	0.51±0.04	0.30±0.15
c_1=-20	c_2=-3.0	0.16±0.03	0.34±0.02	0.44±0.06	0.65±0.04
c_1=-20	c_2=-2.0	0.47±0.01	0.65±0.01	0.64±0.00	0.78±0.00
c_1=-20	c_2=-1.0	0.95±0.01	1.00±0.01	0.96±0.00	0.68±0.02
c_1=-20	c_2=-0.5	1.00±0.01	0.94±0.02	0.78±0.03	0.60±0.00
c_1=-20	c_2=0.0	0.95±0.01	0.71±0.02	0.52±0.09	0.52±0.05
c_1=-10	c_2=-3.0	0.04±0.03	0.32±0.17	0.40±0.18	0.69±0.02
c_1=-10	c_2=-2.0	0.62±0.01	0.70±0.04	0.67±0.03	0.67±0.01
c_1=-10	c_2=-1.0	0.95±0.01	0.82±0.03	0.73±0.02	0.62±0.03
c_1=-10	c_2=-0.5	0.96±0.04	0.74±0.01	0.64±0.02	0.53±0.05
c_1=-10	c_2=0.0	0.72±0.02	0.62±0.01	0.57±0.01	0.43±0.06
c_1=0	c_2=-3.0	0.52±0.02	0.59±0.02	0.61±0.07	0.48±0.04
c_1=0	c_2=-2.0	0.61±0.01	0.59±0.02	0.54±0.03	0.51±0.02
c_1=0	c_2=-1.0	0.56±0.03	0.59±0.04	0.61±0.05	0.54±0.03
c_1=0	c_2=-0.5	0.61±0.00	0.53±0.03	0.50±0.01	0.48±0.02
c_1=0	c_2=0.0	0.54±0.02	0.40±0.09	0.39±0.07	0.43±0.05

(b) Go1

Figure 5.3.: **Sweep over the generalized action space** with 100 different parameterizations (based on prior publication [60]). The mean and standard error of the normalized return across three seeds are reported for each unique combination of the generalized action space.

Random Policy Rollouts

Figure 5.4 shows the initial exploration behaviors of random policies across a position and torque action space for the Go1 and evoBOT tasks. It displays the data from random rollouts for joint position, velocity, and acceleration of multiple joints of the robots. Both time series plots and histograms are provided to analyze the range of motions the random policy explores.

To experimentally assess the effect of initialization, an intervention is designed to modify the initial exploration of the policy. Therefore, a teacher policy $\pi_{\theta_T}^{c_1,c_2,c_3,c_4}$ and a student policy $\pi_{\theta_S}^{c_1,c_2,c_3,c_4}$ are randomly initialized. The student policy is trained using supervised learning to mimic the torque outputs of the randomly initialized teacher, but with a different internal action space

$$\min_{\theta_S} E[(\pi_{\theta_T}^{c_1,c_2,c_3,c_4}(s) - \pi_{\theta_S}^{c_1,c_2,c_3,c_4}(s))^2], \quad \text{for } s \sim \pi_{\theta_T}^{c_1,c_2,c_3,c_4}(s), \tag{5.3.1}$$

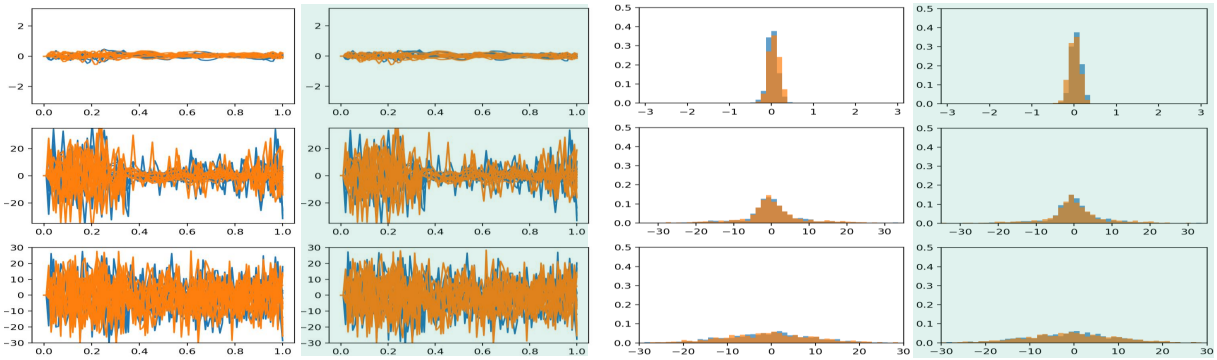
obtained by rolling out the teacher in the training environment. This approach promotes the alignment of the student's rollouts, which operate within a single action space, with the initial exploratory behavior of the other. The green-highlighted blocks in Fig. 5.4 show that the random rollouts from the student policy share characteristics similar to each teacher's. As a result, the teacher-student framework facilitates a seamless transfer between action spaces within this generalized parameterization.

Pre-Trained Policy Initializations

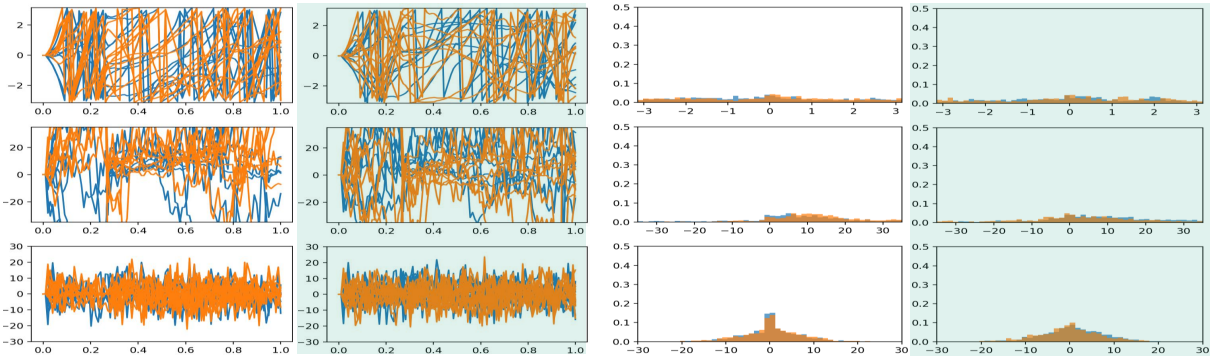
To evaluate the impact of initial exploration on learning dynamics, Figure 5.5 analyzes four configurations: torque from scratch, torque with PD-like initialization, position from scratch, and position with torque-like initialization. In this context, "X with Y-like initialization" means that the X policy is pretrained to mimic rollouts from a randomly initialized Y policy, followed by training the X policy using RL.

For the Go1 task, the torque policy with position-like initialization significantly improves final performance, bringing it close to the performance achieved with a position-controlled policy. Conversely, for evoBOT, where the torque action space is more effective, pre-initializing a PD controller with torque-like initialization results in some performance gains, although it does not completely recover the final performance level.

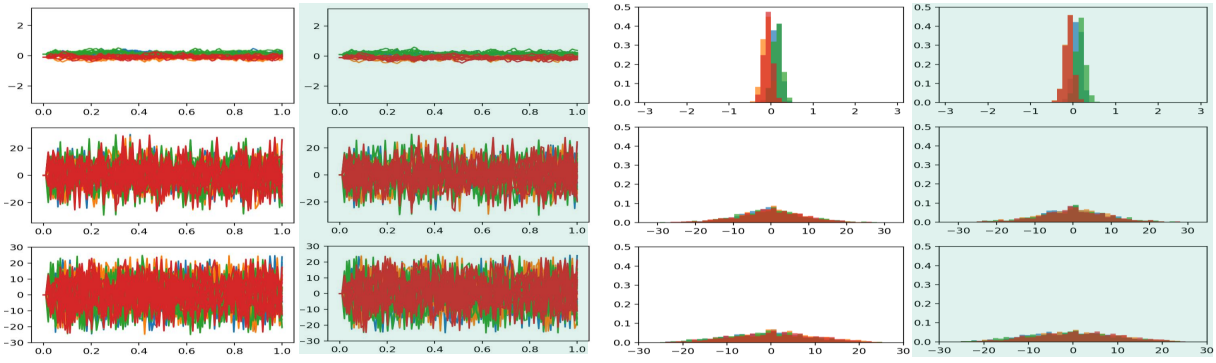
Additional tasks include the FrankaCabinet, BallBalance, and Cartpole, derived from the OmniIsaacGymEnvs suite. Findings indicate that for specific tasks (such as Go1 and evoBOT), the performance gap between action spaces can be partially or fully mitigated by modifying the initial exploration behavior. However, for other tasks, this factor appears to have less influence. Notably, the FrankaCabinet achieves the highest performance with a torque policy that uses PD-like initialization. This suggests that the task may benefit from initially exploring using a PD policy, which provides gravity compensation, before transitioning to a different range of movements.



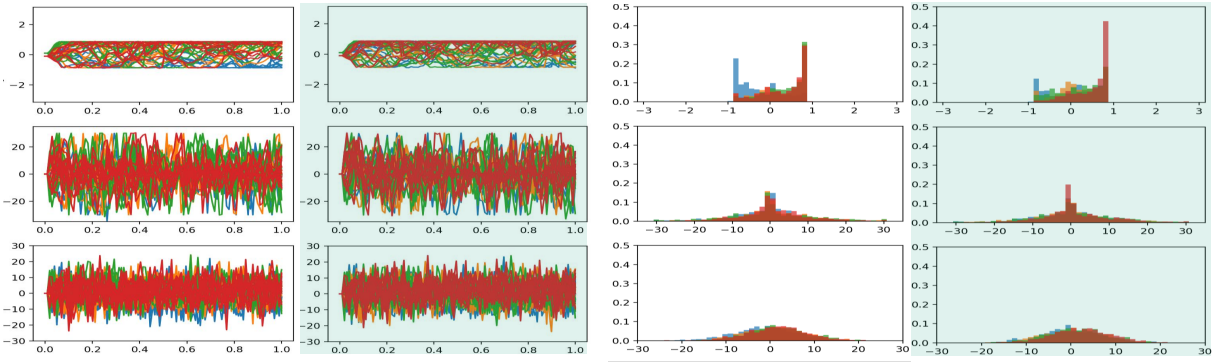
(a) evoBOT Position Control.



(b) evoBOT Torque Control.



(c) Go1 Position Control.



(d) Go1 Torque Control.

Figure 5.4.: **Random rollout behavior** of various action spaces (based on prior publication [60]). Joint positions, velocities, and torques from a rollout of a random policy are shown together with the histogram. Green shading indicates pre-training with another action space.

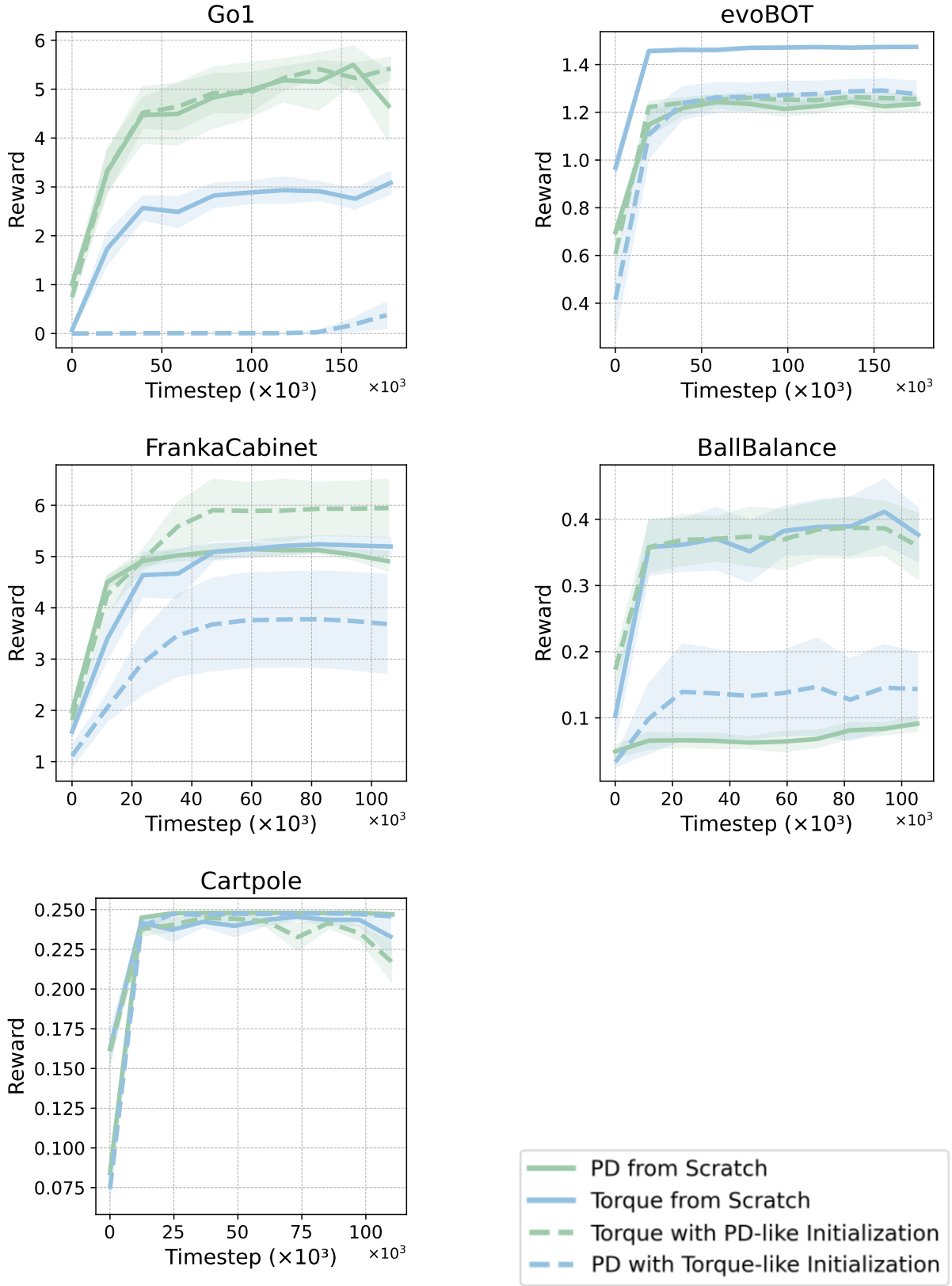


Figure 5.5.: **Initial exploratory behavior** plays a major role in the performance of different action spaces. For dashed lines, the policy underwent a pretraining stage where it was trained through imitation learning to match the random rollouts sampled from another action space.

Table 5.3.: **Expressive capacity** plays a negligible role in the performance of various action spaces. The best-performing policy identified during the action space exploration acts as the teacher, while the student policies mimic it using designated action space representations.

	Normalized Return	
	evoBOT	Go1
Teacher	1.00 ± 0.00	1.00 ± 0.07
Position Student	1.00 ± 0.00	1.00 ± 0.13
Velocity Student	1.00 ± 0.00	1.03 ± 0.16
Torque Student	1.00 ± 0.00	0.99 ± 0.18

5.3.3. Expressive Capacity

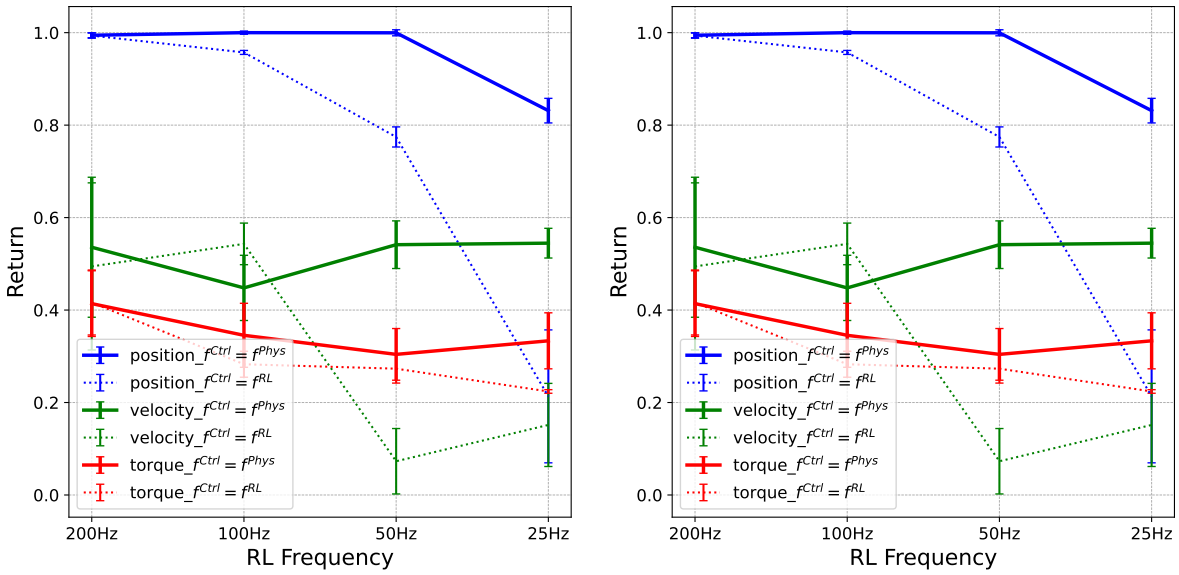
Initial exploration behavior plays a significant role in the performance of different action spaces. Another potential explanation for the relevance of action space design to training performance is that they have different expressive capacities related to their distinct architecture. The findings of this study indicate that this is not true for policies operating at synchronized frequency.

Table 5.3 shows that a policy can be trained with any position, delta position, or torque action space to match the best-performing policy trained in another action space using teacher-student distillation. This is naively true because neural networks are universal function approximators. However, it also illustrates that this comparison lacks factors that would make the policies impossible to imitate, such as varied behavior between timesteps, which will be evaluated in the following experiment.

5.3.4. Behavior Between Timesteps

The impact of temporal factors on policy performance across different action spaces is examined. Policies are trained for the case study at various control frequencies ranging from 25 to 200 Hz for both evoBOT and Go1, with performance evaluated based on total rewards across different action spaces. To ensure comparability, the physics frequency is set to 200 Hz for all experiments, and the RL frequency is implemented using a frame skip of $[1, 2, 4, 8]$. The methodology from [71] is adopted to analyze different control frequencies while keeping the effective discount factor and batch size constant.

Figure 5.6 illustrates that control frequency significantly affects final policy performance. For learning locomotion with evoBOT, higher control frequencies generally result in increased rewards, especially in torque control mode. In the Go1 environment, lower frequencies can also produce satisfactory performance. Furthermore, the behavior between physics substeps can considerably influence final policy outcomes. To differentiate these two effects, the policy performance is analyzed under two conditions: one in which torques are recomputed at each physics substep ($f^{\text{Ctrl}} = f^{\text{Phys}}$) and another in which they remain fixed until the next RL update ($f^{\text{Ctrl}} = f^{\text{RL}}$). Consequently, updating the torques at the physics level, rather than at the RL level, typically yields better performance across most



(a) evoBOT (b) Go1

Figure 5.6.: **Temporal behavior** of policies across multiple action spaces across three seeds with reported mean and standard error of the normalized return (based on prior publication [60]). For various RL frequencies (frame skip), different trends in the policies' performance can be observed. Similarly, torque updates in physics steps also affect final policy performance significantly.

action spaces and tasks. This finding is particularly significant for position control, likely due to the effects of stiffness and damping in PD control.

5.4. Practical Guidelines for Action Space Selection

The results from the benchmarking study (see Section 5.3) indicate that the choice of action space significantly impacts learning performance. However, the analysis across various robot learning tasks reveals no optimal action space applicable to all tasks. Instead, the design of the action space seems to be highly dependent on both the robot and task-specific characteristics. Below are some general guidelines to assist practitioners in designing an appropriate action space for new robot learning tasks:

1. **Consider Robot Dynamics:** When selecting an action space for a new task, evaluate the robot's dynamics and expected movements first. A position-based action space may be beneficial if it is necessary to maintain the joints near a nominal posture by applying torque. For joints requiring continuous rotation, using a higher-level dynamics action space, such as acceleration or torque, may be preferable.

2. **Impact on Exploration:** The chosen action space greatly influences the initial exploration behavior of the policy. Visualizing position, velocity, and torque rollouts from a random policy on the robot can help you understand this impact. Ensure that the selected action space efficiently explores high-reward states within the learning task.

3. **Action Space Parameters:** Learning dynamics are highly sensitive to selecting action space parameters. Choose action space gains that enhance learning dynamics for your specific task. Conducting a sweep over the generalized action space can provide insights into how changing parameters affect policy performance for your particular task.

4. **Control Frequency Selection:** Designing an appropriate action space is closely related to the choice of control frequency. The behavior of the policy between timesteps can significantly influence performance, but these effects may vary depending on the task. Generally, it is advisable to operate the policy at a higher frequency without frame skips.

5. **Sim-to-Real Transfer:** If your physical robot is limited to a specific control interface, the generalized action space can help transition behaviors from one action space to another during an initial imitation phase. To deploy the final policy on the robot with different gains, consider using teacher-student learning to facilitate this conversion.

These guidelines hopefully assist the robot learning community in designing effective action spaces tailored to their particular robot learning tasks.

5.5. Concluding Remarks

This chapter introduces a generalized parameterization for action spaces in RL for robot motor control. It explored the selection of action spaces for various types of robots, including a wheeled-legged robot, a quadruped robot, and a simulated suite of locomotion, manipulation, and control tasks. An extensive benchmarking study was conducted, training over 600 policies using the novel generalized action space to analyze how action space design can enhance performance across different robot learning tasks.
Several important insights emerged from the findings. The choice of action space significantly affects learning outcomes. Different robots exhibited opposite trends when faced with the same locomotion objectives, highlighting the necessity for tailored action space selection that considers the unique dynamics of each robot. Modifying initial exploration behavior can help reduce performance disparities between action spaces for specific tasks. However, this strategy is not universally effective, emphasizing the task-dependent nature of exploration tuning. Despite performance variations among different action spaces, effective policies can often be adapted with similar success. This indicates that while expressive capacity is crucial, it does not impose strict limitations on achieving optimal performance. A significant link was observed between tuning control frequency and action space representation, emphasizing the complex nature of optimizing action spaces with temporal control parameters.
Overall, the results demonstrate that action space selection substantially impacts learning performance, which varies depending on the task. The practical outcomes of selecting an action space are primarily attributed to improved policy initialization and behavior across time steps. These insights provide valuable guidance for enhancing robotic learning and control across various platforms and tasks.

6

Conclusion and Future Directions

This dissertation has explored learning-based control techniques for dynamic robots, focusing on key challenges in the field, including efficient exploration, robust task performance, and bridging the sim-to-real gap. The research aimed to develop methods for integrating knowledge into RL while improving action space design to enhance real-world robotic applications' training efficiency and effectiveness. The two-wheeled mobile robot evoBOT serves as a case study for this work. However, the developed methods and design strategies can be applied to other learning tasks and dynamic robots. Based on the contributions outlined in Section 1.2, the following summarizes how each chapter of this dissertation addresses these challenges (refer to Section 6.1) and presents directions for future research in this area (see Section 6.2).

6.1. Summary of Findings

Guided Reinforcement Learning Chapter 3 introduced the concept of Guided RL, highlighting its potential to improve robotic systems' training efficiency and effectiveness. By systematically integrating external knowledge sources, such as expert demonstrations or historical experiences, into the RL training process, Guided RL not only speeds up the learning curve but also enhances the quality of the learned policies. A key achievement of this chapter was the development of a novel taxonomy for Guided RL, which acts as a modular toolbox. This taxonomy facilitates the incorporation of diverse knowledge sources into the RL pipeline, enriching the learning experience and making it more adaptable to various robotic contexts. The chapter also provided a comprehensive survey of existing Guided RL methods, emphasizing critical aspects such as problem representation, learning strategies, and task structures. This survey highlighted the significance of knowledge transfer from simulation to real-world environments, demonstrating that well-structured guidance can significantly improve sample efficiency. This means that fewer interactions with the environment are required to achieve optimal performance. Quantitative evaluations showed that applying Guided RL methods improved task performance across various robotic applications, reinforcing the framework's relevance in the field. Furthermore, practical guidelines for selecting appropriate Guided RL methods were outlined, stressing the importance of aligning the chosen method with the task's specific requirements and the robot's characteristics. These guidelines recommend considering factors such as the nature

of the task (e.g., locomotion, manipulation), the availability of expert knowledge, and the desired balance between exploration and exploitation during training. By following these recommendations, practitioners can optimize the effectiveness and efficiency of their RL implementations in real-world scenarios.

Case Study on Learning-based Control In Chapter 4, the focus was on developing dynamic motions for the two-wheeled robot evoBOT. The chapter detailed the implementation of learning-based control methods specifically designed for this robot, emphasizing the need for rapid training speeds and robust performance in real-world situations. Key findings demonstrated that policies trained using Guided RL achieved high levels of velocity tracking and actuator responsiveness, effectively controlling the robot during various dynamic tasks. A significant achievement was the successful optimization of the sim-to-real gap, which was accomplished through techniques such as modeling actuator dynamics, analyzing friction, and injecting noise. These methods facilitated the effective integration of real-world data into the training process, resulting in policies that were effective in simulation and transferable to the physical robot. Additionally, performance analyses indicated that the trained policies exhibited remarkable adaptability, successfully managing external disturbances, navigating unknown terrains, and handling varying payloads. The chapter also underscored the importance of structured reward designs and task curricula in enhancing training efficiency. The findings revealed that utilizing dense reward structures and parallel learning strategies significantly improved both the convergence speed and effectiveness of the learned policies. Overall, this case study demonstrated the practical implications of Guided RL in developing robust and dynamic control systems for real-world robotic applications.

Generalized Action Space for Robot Control Chapter 5 examined the role of action space design in RL for dynamic robot control. It found that the choice of action space significantly affects learning outcomes, with different robots demonstrating varying performance trends when tackling the same locomotion objectives. This discovery emphasizes the need for a customized approach to action space selection that considers each robot's unique dynamics and operational constraints. A significant achievement was the development of a generalized parameterization for action spaces, which enables the evaluation of learning-based control strategies across different robot morphologies. An extensive benchmarking study involving training over 600 policies showed that modifying the action space could enhance training speed and policy performance across various tasks. While the chapter pointed out that having a rich and expressive action space is essential, it also suggested that this does not create strict limits on achieving optimal performance. Effective policies can often be adapted to different action representations. Furthermore, the chapter offered practical guidelines for selecting action spaces, highlighting the importance of aligning the action space with the robot's dynamics and considering temporal factors, such as control frequency. This systematic analysis provides researchers and practitioners with valuable tools to improve robotic learning and control across diverse platforms and tasks.

6.2. Implications for Future Research

Key results of this work, including publications, the robot simulation model, and the training code, are available for open access to promote further research in this field.

Further Guided RL Applications Guided RL (see Chapter 3) is a modular framework that enhances training efficiency, task effectiveness, and the sim-to-real transfer process by incorporating knowledge into robot learning tasks. The case study (see Chapter 4) evaluated how Guided RL can improve learning-based control of the two-wheeled mobile robot, evoBOT. Integrating multiple methods from the Guided RL framework has demonstrated improvements in training efficiency, enhanced robustness in task performance, and successful transfer of learned policies to the real robot. Future research could explore how Guided RL can enhance robot learning tasks from other domains. Integrating knowledge to improve cognitive reasoning could benefit challenges such as dexterous manipulation in cluttered environments or multi-robot cooperation. Since evoBOT has simple kinematics with just eight joints, studies could also investigate the impact of Guided RL on training for more complex robot morphologies, such as humanoid robots.

Sim-to-Real Transfer for Complex Tasks The case study (see Chapter 4) investigated methods to optimize the sim-to-real transfer for two-wheeled robots. By incorporating real-world data from the robot and benchmarking with a motion capture system, improved model accuracy and successful sim-to-real transfer of trained policies were achieved. However, the sim-to-real experiments conducted in this work were limited to locomotion tasks in controlled lab environments. Future research could focus on evaluating sim-to-real transfer for more complex tasks. For instance, highly dynamic tasks that involve rapid accelerations at the robot's limits, where wheel slippage becomes unavoidable, warrant further investigation. Additionally, experimental trials examining the effects of varying loads on the real robot would be valuable for assessing the generalization capabilities of the learning-based controller.

Automated Action Space Design The action space study (see Chapter 5) explored how robot control can be enhanced across various platforms and tasks by training hundreds of policies within the proposed generalized action space. The results indicate that the selection of the action space significantly influences learning performance, which varies depending on the specific robot learning task. Although this work identified trends in action space design across multiple tasks, these trends do not fully explain the differences in learning dynamics across varying action spaces. Future work could build upon the generalized action space formulation to automate its design. For example, the generalized parameters of the action space could be incorporated into the training process, allowing the agent to fine-tune robot motor control for new tasks automatically. Another promising direction would involve establishing metrics to predict the suitability of an action space for new tasks before training begins, potentially accelerating the automated design of environments, such as using foundation models [122].

Autonomous Environment Interaction The case study (see Chapter 4) developed methods for learning-based control of the two-wheeled robot evoBOT. Specifically, the reinforcement learning controller was trained to accomplish various task variants simultaneously, such as robust balancing, dynamic locomotion, and effective packet handling. This machine learning-based approach facilitates faster iteration cycles and reduces the need for manual tuning in future robot control designs. Future research could focus on integrating additional capabilities into the learning-based controller. While the current controller has demonstrated its ability to manage various objects during locomotion, enhancing it to enable complete control over the arms and grippers could be a future direction for achieving more autonomous interactions with the environment. Similarly, incorporating additional sensor modalities, such as vision or tactile sensing, into the training process could pave the way for even more autonomous interactions of robots with the real world.

Bibliography

[1] **R. Agarwal, D. Schuurmans, and M. Norouzi:** "An Optimistic Perspective on Offline Reinforcement Learning". en. In: *Proceedings of the 37th International Conference on Machine Learning*. ISSN: 2640-3498. Nov. 2020, pp. 104–114.

[2] **M. Akbulut, E. Oztop, M. Y. Seker, H. X, A. Tekden, and E. Ugur:** "ACNMP: Skill Transfer and Task Extrapolation through Learning from Demonstration and Reinforcement Learning via Representation Sharing". en. In: *Proceedings of the 2020 Conference on Robot Learning*. ISSN: 2640-3498. Oct. 2021, pp. 1896–1907.

[3] **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". In: *IEEE Robotics and Automation Letters* 9.6 (June 2024). Conference Name: IEEE Robotics and Automation Letters, pp. 5895–5902. ISSN: 2377-3766. DOI: 10.1109/LRA.2024.3398428.

[4] **A. Allshire, R. Martín-Martín, C. Lin, S. Manuel, S. Savarese, and A. Garg:** "LASER: Learning a Latent Action Space for Efficient Reinforcement Learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2021, pp. 6650–6656. DOI: 10.1109/ICRA48506.2021.9561232.

[5] **A. Anandakrishnan:** "Hierarchical Reinforcement Learning for Agile Loco-Manipulation". MA thesis. TU Dortmund University, Nov. 2024.

[6] **M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba:** "Hindsight Experience Replay". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[7] **K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath:** "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (Nov. 2017). Conference Name: IEEE Signal Processing Magazine, pp. 26–38. ISSN: 1558-0792. DOI: 10.1109/MSP.2017.2743240.

[8] **P.-L. Bacon, J. Harb, and D. Precup:** "The Option-Critic Architecture". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 31.1 (Feb. 2017). Number: 1. ISSN: 2374-3468. DOI: 10.1609/aaai.v31i1.10916.

[9] **D. Baek, A. Purushottam, and J. Ramos:** "Hybrid LMC: Hybrid Learning and Model-based Control for Wheeled Humanoid Robot via Ensemble Deep Reinforcement Learning". In: *2022 IEEE/RSJ International Conference on Intelligent*

Robots and Systems (IROS). ISSN: 2153-0866. Oct. 2022, pp. 9347–9354. DOI: 10.1109/IROS47612.2022.9981913.

[10] **S. Bahl, A. Gupta, and D. Pathak:** "Hierarchical Neural Dynamic Policies". en. In: *Robotics: Science and Systems XVII*. July 2021. ISBN: 978-0-9923747-7-8. DOI: 10.15607/RSS.2021.XVII.023.

[11] **S. Bahl, M. Mukadam, A. Gupta, and D. Pathak:** "Neural Dynamic Policies for End-to-End Sensorimotor Learning". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 5058–5069.

[12] **M. Bain and C. Sammut:** "A framework for behavioural cloning". en. In: *Machine Intelligence 15*. Jan. 2000, pp. 103–129. ISBN: 978-0-19-853867-7 978-1-383-02650-4. DOI: 10.1093/oso/9780198538677.003.0006.

[13] **S. Bajpai:** "Sim-to-Real Transfer Methods for a Dynamically Balancing Robot". MA thesis. TU Dortmund University, Feb. 2023.

[14] **C. Banerjee, Z. Chen, and N. Noman:** "Improved Soft Actor-Critic: Mixing Prioritized Off-Policy Samples With On-Policy Experiences". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.3 (Mar. 2024). Conference Name: IEEE Transactions on Neural Networks and Learning Systems, pp. 3121–3129. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2022.3174051.

[15] **G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap:** "Distributed Distributional Deterministic Policy Gradients". In: arXiv:1804.08617. Apr. 2018. DOI: 10.48550/arXiv.1804.08617.

[16] **A. G. Barto and S. Mahadevan:** "Recent Advances in Hierarchical Reinforcement Learning". en. In: *Discrete Event Dynamic Systems* 13.4 (Oct. 2003), pp. 341–379. ISSN: 1573-7594. DOI: 10.1023/A:1025696116075.

[17] **S. Batra, Z. Huang, A. Petrenko, T. Kumar, A. Molchanov, and G. S. Sukhatme:** "Decentralized Control of Quadrotor Swarms with End-to-end Deep Reinforcement Learning". en. In: *Proceedings of the 5th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2022, pp. 576–586.

[18] **G. Bellegarda, Y. Chen, Z. Liu, and Q. Nguyen:** "Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2022, pp. 10364–10370. DOI: 10.1109/IROS47612.2022.9982132.

[19] **Y. Bengio, J. Louradour, R. Collobert, and J. Weston:** "Curriculum learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. June 2009, pp. 41–48. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553380.

[20] **A. G. Billard, S. Calinon, and R. Dillmann:** "Learning from Humans". en. In: *Springer Handbook of Robotics*. 2016, pp. 1995–2014. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_74.

[21] **G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim:** "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2018, pp. 2245–2252. DOI: 10.1109/IROS.2018.8593885.

[22] **M. Bogdanovic, M. Khadiv, and L. Righetti:** "Learning Variable Impedance Control for Contact Sensitive Tasks". In: *IEEE Robotics and Automation Letters* 5.4 (Oct. 2020). Conference Name: IEEE Robotics and Automation Letters, pp. 6129–6136. ISSN: 2377-3766. DOI: 10.1109/LRA.2020.3011379.

[23] *Boston Dynamics Handle Robot Reimagined for Logistics*. URL: https://bostondynamics.com/ (visited on 04/15/2025).

[24] **K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke:** "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2018, pp. 4243–4250. DOI: 10.1109/ICRA.2018.8460875.

[25] **S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, O. Sushkov, D. Barker, J. Scholz, M. Denil, N. d. Freitas, and Z. Wang:** "Scaling data-driven robotics with reward sketching and batch reinforcement learning". In: arXiv:1909.12200. June 2020. DOI: 10.48550/arXiv.1909.12200.

[26] **C. Celemin, G. Maeda, J. Ruiz-del-Solar, J. Peters, and J. Kober:** "Reinforcement learning of motor skills using Policy Search and human corrective advice". en. In: *The International Journal of Robotics Research* 38.14 (Dec. 2019). Publisher: SAGE Publications Ltd STM, pp. 1560–1580. ISSN: 0278-3649. DOI: 10.1177/0278364919871998.

[27] **L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch:** "Decision Transformer: Reinforcement Learning via Sequence Modeling". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021, pp. 15084–15097.

[28] **S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath:** "Learning Torque Control for Quadrupedal Locomotion". In: arXiv:2203.05194 [cs]. Mar. 2023. DOI: 10.48550/arXiv.2203.05194.

[29] **T. Chen, J. Xu, and P. Agrawal:** "A System for General In-Hand Object Re-Orientation". en. In: *Proceedings of the 5th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2022, pp. 297–307.

[30] **Z. Chen, X. Hu, and A. Owens:** "Structure from Silence: Learning Scene Structure from Ambient Sound". In: arXiv:2111.05846. Nov. 2021. DOI: 10.48550/arXiv.2111.05846.

[31] **X. Cheng, K. Shi, A. Agarwal, and D. Pathak:** "Extreme Parkour with Legged Robots". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. May 2024, pp. 11443–11450. DOI: 10.1109/ICRA57147.2024.10610200.

[32] **H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis:** "Learning Navigation Behaviors End-to-End With AutoRL". In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019). Conference Name: IEEE Robotics and Automation Letters, pp. 2007–2014. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2899918.

[33] **A. Church, J. Lloyd, R. Hadsell, and N. F. Lepora:** "Tactile Sim-to-Real Policy Transfer via Real-to-Sim Image Translation". en. In: *Proceedings of the 5th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2022, pp. 1645–1654.

[34] **J. Collins, S. Chand, A. Vanderkop, and D. Howard:** "A Review of Physics Simulators for Robotic Applications". In: *IEEE Access* 9 (2021). Conference Name: IEEE Access, pp. 51416–51431. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3068769.

[35] **E. Coumans:** "Bullet physics simulation". In: *ACM SIGGRAPH 2015 Courses*. SIGGRAPH '15. July 2015, p. 1. ISBN: 978-1-4503-3634-5. DOI: 10.1145/2776880.2792704.

[36] **M. Dalal, D. Pathak, and R. R. Salakhutdinov:** "Accelerating Robotic Reinforcement Learning via Parameterized Action Primitives". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021, pp. 21847–21859.

[37] **J. Dao, H. Duan, and A. Fern:** "Sim-to-Real Learning for Humanoid Box Loco-Manipulation". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. May 2024, pp. 16930–16936. DOI: 10.1109/ICRA57147.2024.10610977.

[38] **P. N. Dao and Y.-C. Liu:** "Adaptive Reinforcement Learning Strategy with Sliding Mode Control for Unknown and Disturbed Wheeled Inverted Pendulum". en. In: *International Journal of Control, Automation and Systems* 19.2 (Feb. 2021), pp. 1139–1150. ISSN: 2005-4092. DOI: 10.1007/s12555-019-0912-9.

[39] **S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn:** "RoboNet: Large-Scale Multi-Robot Learning". In: arXiv:1910.11215. Jan. 2020. DOI: 10.48550/arXiv.1910.11215.

[40] **H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst:** "Learning Task Space Actions for Bipedal Locomotion". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2021, pp. 1276–1282. DOI: 10.1109/ICRA48506.2021.9561705.

[41] **G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester:** "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis". en. In: *Machine Learning* 110.9 (Sept. 2021), pp. 2419–2468. ISSN: 1573-0565. DOI: 10.1007/s10994-021-05961-4.

[42] **G. Eoh and T.-H. Park:** "Cooperative Object Transportation Using Curriculum-Based Deep Reinforcement Learning". en. In: *Sensors* 21.14 (Jan. 2021). Number: 14 Publisher: Multidisciplinary Digital Publishing Institute, p. 4780. ISSN: 1424-8220. DOI: 10.3390/s21144780.

[43] **T. Erez, Y. Tassa, and E. Todorov:** "Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 1050-4729. May 2015, pp. 4397–4404. DOI: 10.1109/ICRA.2015.7139807.

[44] **A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel:** "Adversarial Motion Priors Make Good Substitutes for Complex Reward Functions". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2022, pp. 25–32. DOI: 10.1109/IROS47612.2022.9981973.

[45] **L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu:** "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures". en. In: *Proceedings of the 35th International Conference on Machine Learning*. ISSN: 2640-3498. July 2018, pp. 1407–1416.

[46] **J. Eßer:** *Embodied AI Explained: Principles, Applications, and Future Perspectives » Lamarr-Blog*. Jan. 2025. URL: https://lamarr-institute.org/blog/embodied-ai-explained/ (visited on 04/07/2025).

[47] **J. Eßer:** *Fabriken der Zukunft: Julian Eßer über Reinforcement Learning in der Robotik*. Feb. 2024. URL: https://aigrid-qlcxw4e9ng.live-website.com/interview/fabriken-der-zukunft-ein-gespraech-mit-julian-esser-ueber-reinforcement-learning-in-der-robotik/ (visited on 04/07/2025).

[48] **J. Eßer:** *Guided Reinforcement Learning – Application to three Robotic Tasks » Lamarr-Blog*. July 2023. URL: https://lamarr-institute.org/blog/reinforcement-learning-application-in-robotics/ (visited on 04/07/2025).

[49] **J. Eßer:** *Guided Reinforcement Learning – Deployed for Dynamic Locomotion of a Two-Wheeled Robot » Lamarr-Blog*. Oct. 2023. URL: https://lamarr-institute.org/blog/guided-reinforcement-learning-dynamic-locomotion/ (visited on 04/07/2025).

[50] **J. Eßer:** *Guided Reinforcement Learning – Towards Efficient and Effective Real-World Robotics » Lamarr-Blog*. July 2023. URL: https://lamarr-instit

ute.org/blog/guided-reinforcement-learning-and-robotics/ (visited on 04/07/2025).

[51] **J. Eßer:** *Introduction to Reinforcement Learning – A Robotics Perspective » Lamarr-Blog*. July 2023. URL: https://lamarr-institute.org/blog/reinforcement-learning-and-robotics/ (visited on 04/07/2025).

[52] **J. Eßer:** *Reinforcement Learning für Roboter - Belohnung als Anreiz zum Lernen | ROBOTIK UND PRODUKTION*. Section: Anwendungen & Lösungen. Mar. 2024. URL: https://robotik-produktion.de/robotik/belohnung-als-anreiz-zum-lernen/ (visited on 04/07/2025).

[53] **J. Eßer:** *Reinforcement Learning: Roboter lernen besser mit Belohnung - IT Daily*. Section: Industrie 4.0 & RPA. Feb. 2024. URL: https://www.it-daily.net/it-management/industrie-rpa/roboter-lernen-besser-mit-belohnung (visited on 04/07/2025).

[54] **J. Eßer:** *Training Code for evoBOT | NVIDIA Omniverse Isaac Gym Environments*. Apr. 2025. URL: https://github.com/julesser/ral-action-spaces/tree/main/omniisaacgymenvs (visited on 04/15/2025).

[55] **J. Eßer, N. Bach, C. Jestel, O. Urbann, and S. Kerner:** "Guided Reinforcement Learning: A Review and Evaluation for Efficient and Effective Real-World Robotics [Survey]". In: *IEEE Robotics & Automation Magazine* 30.2 (June 2023), pp. 67–85. ISSN: 1558-223X. DOI: 10.1109/MRA.2022.3207664.

[56] **J. Eßer, F. Feldmeier, and R. Gasoto:** *evoBOT Simulation Model for Isaac SIM*. Aug. 2024. URL: https://git.openlogisticsfoundation.org/silicon-economy/simulation-model/evobotsimmodel (visited on 04/07/2025).

[57] **J. Eßer, I. Havoitis, S. Kumar, G. Margolis, C. Mastalli, C. Semini, O. Stasse, and O. Urbann:** *Loco-Manipulation Workshop | ICRA 2024*. May 2025. URL: https://sites.google.com/view/loco-manipulation-icra24/home (visited on 02/28/2025).

[58] **J. Eßer and S. Kerner:** *Next-Generation Robotics: Training Agile Loco-Manipulation and Human-Machine Interaction | GTC Spring 2024 | NVIDIA On-Demand*. Mar. 2024. URL: https://www.nvidia.com/en-us/on-demand/session/gtc24-s62865/ (visited on 04/07/2025).

[59] **J. Eßer and S. Kerner:** *Training Highly Dynamic Robots for Complex Tasks in Industrial Applications | GTC Spring 2023 | NVIDIA On-Demand*. Mar. 2023. URL: https://www.nvidia.com/en-us/on-demand/session/gtcspring23-s51823/ (visited on 04/07/2025).

[60] **J. Eßer, G. B. Margolis, O. Urbann, S. Kerner, and P. Agrawal:** "Action Space Design in Reinforcement Learning for Robot Motor Skills". In: *Proceedings of The 8th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2025, pp. 4021–4032.

[61] **C. Eteke, D. Kebüde, and B. Akgün:** "Reward Learning From Very Few Demonstrations". In: *IEEE Transactions on Robotics* 37.3 (June 2021). Conference Name: IEEE Transactions on Robotics, pp. 893–904. ISSN: 1941-0468. DOI: 10.1109/TRO.2020.3038698.

[62] *evoBOT - The Evolution of Autonomous Mobile Robots | Fraunhofer IML.* URL: https://www.iml.fraunhofer.de/en/fields_of_activity/material-flow-systems/iot-and-embedded-systems/evobot.html (visited on 04/06/2025).

[63] **F. Feldmeier:** "Reduction of the Sim-to-real Gap for a Highly Dynamic Robot". MA thesis. TU Dortmund University, Aug. 2024.

[64] **C. Finn, P. Abbeel, and S. Levine:** "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". en. In: *Proceedings of the 34th International Conference on Machine Learning*. ISSN: 2640-3498. July 2017, pp. 1126–1135.

[65] **P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson:** "Implicit Behavioral Cloning". en. In: *Proceedings of the 5th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2022, pp. 158–168.

[66] **C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel:** "Reverse Curriculum Generation for Reinforcement Learning". en. In: *Proceedings of the 1st Annual Conference on Robot Learning*. ISSN: 2640-3498. Oct. 2017, pp. 482–495.

[67] **Z. Fu, X. Cheng, and D. Pathak:** "Deep Whole-Body Control: Learning a Unified Policy for Manipulation and Locomotion". In: *Proceedings of The 6th Conference on Robot Learning*. ISSN: 2640-3498. Mar. 2023, pp. 138–149.

[68] **Z. Fu, A. Kumar, J. Malik, and D. Pathak:** "Minimizing Energy Consumption Leads to the Emergence of Gaits in Legged Robots". In: arXiv:2111.01674. Oct. 2021. DOI: 10.48550/arXiv.2111.01674.

[69] **S. Fujimoto, D. Meger, and D. Precup:** "Off-Policy Deep Reinforcement Learning without Exploration". en. In: *Proceedings of the 36th International Conference on Machine Learning*. ISSN: 2640-3498. May 2019, pp. 2052–2062.

[70] **A. Ganapathi, P. Florence, J. Varley, K. Burns, K. Goldberg, and A. Zeng:** "Implicit Kinematic Policies: Unifying Joint and Cartesian Action Spaces in End-to-End Robot Learning". In: *2022 International Conference on Robotics and Automation (ICRA)*. May 2022, pp. 2656–2662. DOI: 10.1109/ICRA46639.2022.9812165.

[71] **S. Gangapurwala, L. Campanaro, and I. Havoutis:** "Learning Low-Frequency Motion Control for Robust and Dynamic Robot Locomotion". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. May 2023, pp. 5085–5091. DOI: 10.1109/ICRA48891.2023.10160357.

[72] **F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer:** "Sim-to-Real Transfer with Neural-Augmented Robot Simulation". en. In: *Proceedings of The 2nd Conference on Robot Learning*. ISSN: 2640-3498. Oct. 2018, pp. 817–828.

[73] **R. Gordon:** *DribbleBot From MIT Designed to Play Soccer on Varied Terrains*. Apr. 2023. URL: https://www.robotics247.com/article/dribblebot_mit_plays_soccer_varied_terrains (visited on 04/12/2025).

[74] **K. Guo, V. Makoviichuk, and G. State:** *NVIDIA Omniverse Isaac Gym*. June 2022. URL: https://github.com/isaac-sim/OmniIsaacGymEnvs (visited on 04/12/2025).

[75] **T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine:** "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". en. In: *Proceedings of the 35th International Conference on Machine Learning*. ISSN: 2640-3498. July 2018, pp. 1861–1870.

[76] **E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme:** "NeuralSim: Augmenting Differentiable Simulators with Neural Networks". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2021, pp. 9474–9481. DOI: 10.1109/ICRA48506.2021.9560935.

[77] **D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter:** "Learning a State Representation and Navigation in Cluttered and Dynamic Environments". In: *IEEE Robotics and Automation Letters* 6.3 (July 2021). Conference Name: IEEE Robotics and Automation Letters, pp. 5081–5088. ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3068639.

[78] **D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. v. Hasselt, and D. Silver:** "Distributed Prioritized Experience Replay". In: arXiv:1803.00933. Mar. 2018. DOI: 10.48550/arXiv.1803.00933.

[79] **J. Huang, J. Rojas, M. Zimmer, H. Wu, Y. Guan, and P. Weng:** "Hyperparameter Auto-Tuning in Self-Supervised Robotic Learning". In: *IEEE Robotics and Automation Letters* 6.2 (Apr. 2021). Conference Name: IEEE Robotics and Automation Letters, pp. 3537–3544. ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3064509.

[80] **J. Hughes, A. Abdulali, R. Hashem, and F. Iida:** "Embodied Artificial Intelligence: Enabling the Next Intelligence Revolution". en. In: *IOP Conference Series: Materials Science and Engineering* 1261.1 (Oct. 2022). Publisher: IOP Publishing, p. 012001. ISSN: 1757-899X. DOI: 10.1088/1757-899X/1261/1/012001.

[81] **J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter:** "Learning agile and dynamic motor skills for legged robots". en. In: *Science Robotics* 4.26 (Jan. 2019), eaau5872. ISSN: 2470-9476. DOI: 10.1126/scirobotics.aau5872.

[82] **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". en. In: *The International Journal of Robotics Research* 40.4-5 (Apr. 2021). Publisher: SAGE Publications Ltd STM, pp. 698–721. ISSN: 0278-3649. DOI: 10.1177/0278364920987859.

[83] **C. S. Imai, M. Zhang, Y. Zhang, M. Kierebiński, R. Yang, Y. Qin, and X. Wang:** "Vision-Guided Quadrupedal Locomotion in the Wild with Multi-Modal Delay Randomization". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2022, pp. 5556–5563. DOI: 10.1109/IROS47612.2022.9981072.

[84] **M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel:** "Human-level performance in 3D multiplayer games with population-based reinforcement learning". In: *Science* 364.6443 (May 2019). Publisher: American Association for the Advancement of Science, pp. 859–865. DOI: 10.1126/science.aau6249.

[85] **S. James, K. Wada, T. Laidlow, and A. J. Davison:** "Coarse-To-Fine Q-Attention: Efficient Learning for Visual Robotic Manipulation via Discretisation". en. In: 2022, pp. 13739–13748.

[86] **S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis:** "Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks". In: 2019, pp. 12627–12637.

[87] **R. Jangir, N. Hansen, S. Ghosal, M. Jain, and X. Wang:** "Look Closer: Bridging Egocentric and Third-Person Views With Transformers for Robotic Manipulation". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022). Conference Name: IEEE Robotics and Automation Letters, pp. 3046–3053. ISSN: 2377-3766. DOI: 10.1109/LRA.2022.3144512.

[88] **C. Jestel, K. Rösner, N. Dietz, N. Bach, J. Eßer, J. Finke, and O. Urbann:** "MuRoSim – A Fast and Efficient Multi-Robot Simulation for Learning-based Navigation". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. May 2024, pp. 16881–16887. DOI: 10.1109/ICRA57147.2024.10610375.

[89] **G. Ji, J. Mun, H. Kim, and J. Hwangbo:** "Concurrent Training of a Control Policy and a State Estimator for Dynamic and Robust Legged Locomotion". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022). Conference Name: IEEE Robotics and Automation Letters, pp. 4630–4637. ISSN: 2377-3766. DOI: 10.1109/LRA.2022.3151396.

[90] **D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman:** "MT-Opt: Continuous Multi-Task

Robotic Reinforcement Learning at Scale". In: arXiv:2104.08212. Apr. 2021. DOI: 10.48550/arXiv.2104.08212.

[91] **A. Kanwischer and O. Urbann:** "A Machine Learning Approach to Minimization of the Sim-To-Real Gap via Precise Dynamics Modeling of a Fast Moving Robot". In: *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. Dec. 2022, pp. 349–354. DOI: 10.1109/ICARCV57592.2022.10004376.

[92] **S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney:** "Recurrent Experience Replay in Distributed Reinforcement Learning". en. In: Sept. 2018.

[93] **E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza:** "Champion-level drone racing using deep reinforcement learning". en. In: *Nature* 620.7976 (Aug. 2023), pp. 982–987. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-023-06419-4.

[94] **E. Kaufmann, L. Bauersfeld, and D. Scaramuzza:** "A Benchmark Comparison of Learned Control Policies for Agile Quadrotor Flight". In: *2022 International Conference on Robotics and Automation (ICRA)*. May 2022, pp. 10504–10510. DOI: 10.1109/ICRA46639.2022.9811564.

[95] **S. Kerner and J. Eßer:** *Towards a Digital Reality in Logistics Automation: Optimization of Sim-to-Real | GTC Spring 2022 | NVIDIA On-Demand*. Mar. 2022. URL: https://www.nvidia.com/en-us/on-demand/session/gtcspring22-s42559/ (visited on 04/07/2025).

[96] **O. Khatib:** "A unified approach for motion and force control of robot manipulators: The operational space formulation". In: *IEEE Journal on Robotics and Automation* 3.1 (Feb. 1987). Conference Name: IEEE Journal on Robotics and Automation, pp. 43–53. ISSN: 2374-8710. DOI: 10.1109/JRA.1987.1087068.

[97] **V. Klemm, A. Morra, C. Salzmann, F. Tschopp, K. Bodie, L. Gulich, N. Küng, D. Mannhart, C. Pfister, M. Vierneisel, F. Weber, R. Deuber, and R. Siegwart:** "Ascento: A Two-Wheeled Jumping Robot". In: *2019 International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2019, pp. 7515–7521. DOI: 10.1109/ICRA.2019.8793792.

[98] **P. Klink, H. Abdulsamad, B. Belousov, and J. Peters:** "Self-Paced Contextual Reinforcement Learning". en. In: *Proceedings of the Conference on Robot Learning*. ISSN: 2640-3498. May 2020, pp. 513–529.

[99] **P. Klokowski, J. Eßer, N. Gramse, B. Pschera, M. Plitt, F. Feldmeier, S. Bajpai, C. Jestel, N. Bach, O. Urbann, and S. Kerner:** "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)*. ISSN: 2153-0866. Oct. 2023, pp. 10425–10432. DOI: 10.1109/IROS55552.2023.10342128.

[100] **J. Kober, J. A. Bagnell, and J. Peters:** "Reinforcement learning in robotics: A survey". en. In: *The International Journal of Robotics Research* 32.11 (Sept. 2013). Publisher: SAGE Publications Ltd STM, pp. 1238–1274. ISSN: 0278-3649. DOI: 10.1177/0278364913495721.

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

[102] **S. Kolev and E. Todorov:** "Physically consistent state estimation and system identification for contacts". In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. Nov. 2015, pp. 1036–1043. DOI: 10.1109/HUMANOIDS.2015.7363481.

[103] **I. Kostrikov, A. Nair, and S. Levine:** "Offline Reinforcement Learning with Implicit Q-Learning". In: arXiv:2110.06169. Oct. 2021. DOI: 10.48550/arXiv.2110.06169.

[104] **A. Kumar, Z. Fu, D. Pathak, and J. Malik:** "RMA: Rapid Motor Adaptation for Legged Robots". en. In: *arXiv.org*. July 2021.

[105] **N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. J. Pister:** "Low-Level Control of a Quadrotor With Deep Model-Based Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019). Conference Name: IEEE Robotics and Automation Letters, pp. 4224–4230. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2930489.

[106] **M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg:** "DART: Noise Injection for Robust Imitation Learning". en. In: *Proceedings of the 1st Annual Conference on Robot Learning*. ISSN: 2640-3498. Oct. 2017, pp. 143–156.

[107] **H. Le, N. Jiang, A. Agarwal, M. Dudik, Y. Yue, and I. I. I. Hal Daumé:** "Hierarchical Imitation and Reinforcement Learning". en. In: *Proceedings of the 35th International Conference on Machine Learning*. ISSN: 2640-3498. July 2018, pp. 2917–2926.

[108] **A. X. Lee, C. M. Devin, Y. Zhou, T. Lampe, K. Bousmalis, J. T. Springenberg, A. Byravan, A. Abdolmaleki, N. Gileadi, D. Khosid, C. Fantacci, J. E. Chen, A. Raju, R. Jeong, M. Neunert, A. Laurens, S. Saliceti, F. Casarini, M. Riedmiller, R. Hadsell, and F. Nori:** "Beyond Pick-and-Place: Tackling Robotic Stacking of Diverse Shapes". en. In: June 2021.

[109] **J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter:** "Learning quadrupedal locomotion over challenging terrain". en. In: *Science Robotics* 5.47 (Oct. 2020), eabc5986. ISSN: 2470-9476. DOI: 10.1126/scirobotics.abc5986.

[110] **S. Levine, A. Kumar, G. Tucker, and J. Fu:** "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems". In: arXiv:2005.01643. Nov. 2020. DOI: 10.48550/arXiv.2005.01643.

[111] **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: *International Journal of Semantic Computing* 16.03 (Sept. 2022). Publisher: World Scientific Publishing Co., pp. 381–402. ISSN: 1793-351X. DOI: 10.1142/S1793351X22430024.

[112] **C. Li, F. Xia, R. Martín-Martín, and S. Savarese:** "HRL4IN: Hierarchical Reinforcement Learning for Interactive Navigation with Mobile Manipulators". en. In: *Proceedings of the Conference on Robot Learning*. ISSN: 2640-3498. May 2020, pp. 603–616.

[113] **Y. Li:** "Deep Reinforcement Learning: An Overview". en. In: arXiv:1701.07274 [cs]. Nov. 2018.

[114] **J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox:** "GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning". en. In: *Proceedings of The 2nd Conference on Robot Learning*. ISSN: 2640-3498. Oct. 2018, pp. 270–282.

[115] **T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra:** "Continuous control with deep reinforcement learning". en. In: *arXiv.org*. Sept. 2015.

[116] **H.-K. Lim, J.-B. Kim, C.-M. Kim, G.-Y. Hwang, H.-b. Choi, and Y.-H. Han:** "Federated Reinforcement Learning for Controlling Multiple Rotary Inverted Pendulums in Edge Computing Environments". In: *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. Feb. 2020, pp. 463–464. DOI: 10.1109/ICAIIC48513.2020.9065233.

[117] **M. Liu, Z. Chen, X. Cheng, Y. Ji, R.-Z. Qiu, R. Yang, and X. Wang:** "Visual Whole-Body Control for Legged Loco-Manipulation". en. In: *8th Annual Conference on Robot Learning*. Sept. 2024.

[118] **N. Liu, Y. Cai, T. Lu, R. Wang, and S. Wang:** "Real–Sim–Real Transfer for Real-World Robot Control Policy Learning with Deep Reinforcement Learning". en. In: *Applied Sciences* 10.5 (Jan. 2020). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 1555. ISSN: 2076-3417. DOI: 10.3390/app10051555.

[119] **L. Ljung:** "System Identification". en. In: *Signal Analysis and Prediction*. 1998, pp. 163–173. ISBN: 978-1-4612-1768-8. DOI: 10.1007/978-1-4612-1768-8_11.

[120] **K. Lowrey, S. Kolev, J. Dao, A. Rajeswaran, and E. Todorov:** "Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system". In: *2018 IEEE International Conference on Simulation, Model-*

ing, and Programming for Autonomous Robots (SIMPAR). May 2018, pp. 35–42. DOI: 10.1109/SIMPAR.2018.8376268.

[121] **S. Luo, H. Kasaei, and L. Schomaker:** "Accelerating Reinforcement Learning for Reaching Using Continuous Curriculum Learning". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. ISSN: 2161-4407. July 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207427.

[122] **Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar:** "Eureka: Human-Level Reward Design via Coding Large Language Models". en. In: arXiv:2310.12931 [cs]. Apr. 2024.

[123] **V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State:** "Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning". en. In: Aug. 2021.

[124] **H. Mania, A. Guy, and B. Recht:** "Simple random search of static linear policies is competitive for reinforcement learning". In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

[125] **G. B. Margolis and P. Agrawal:** "Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior". en. In: *Proceedings of The 6th Conference on Robot Learning*. ISSN: 2640-3498. Mar. 2023, pp. 22–31.

[126] **G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. b. Kim, and P. Agrawal:** "Learning to Jump from Pixels". en. In: *Proceedings of the 5th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2022, pp. 1025–1034.

[127] **G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal:** "Rapid locomotion via reinforcement learning". In: *The International Journal of Robotics Research* 43.4 (Apr. 2024). Publisher: SAGE Publications Ltd STM, pp. 572–587. ISSN: 0278-3649. DOI: 10.1177/02783649231224053.

[128] **R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg:** "Variable Impedance Control in End-Effector Space: An Action Space for Reinforcement Learning in Contact-Rich Tasks". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Nov. 2019, pp. 1010–1017. DOI: 10.1109/IROS40897.2019.8968201.

[129] **T. Matiisen, A. Oliver, T. Cohen, and J. Schulman:** "Teacher–Student Curriculum Learning". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (Sept. 2020). Conference Name: IEEE Transactions on Neural Networks and Learning Systems, pp. 3732–3740. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2019.2934906.

[130] **B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull:** "Active Domain Randomization". en. In: *Proceedings of the Conference on Robot Learning*. ISSN: 2640-3498. May 2020, pp. 1162–1176.

[131] **A. Melnik, L. Lach, M. Plappert, T. Korthals, R. Haschke, and H. Ritter:** "Using Tactile Sensing to Improve the Sample Efficiency and Performance of Deep Deterministic Policy Gradients for Simulated In-Hand Manipulation Tasks". English. In: *Frontiers in Robotics and AI* 8 (June 2021). Publisher: Frontiers. ISSN: 2296-9144. DOI: 10.3389/frobt.2021.538773.

[132] **T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter:** "Learning robust perceptive locomotion for quadrupedal robots in the wild". In: *Science Robotics* 7.62 (Jan. 2022). Publisher: American Association for the Advancement of Science, eabk2822. DOI: 10.1126/scirobotics.abk2822.

[133] **M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg:** "Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments". In: *IEEE Robotics and Automation Letters* 8.6 (June 2023). Conference Name: IEEE Robotics and Automation Letters, pp. 3740–3747. ISSN: 2377-3766. DOI: 10.1109/LRA.2023.3270034.

[134] **V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu:** "Asynchronous Methods for Deep Reinforcement Learning". In: arXiv:1602.01783. June 2016. DOI: 10.48550/arXiv.1602.01783.

[135] **V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller:** "Playing Atari with Deep Reinforcement Learning". en. In: (2013).

[136] **F. Muratore, M. Gienger, and J. Peters:** "Assessing Transferability From Simulation to Reality for Reinforcement Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.4 (Apr. 2021). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1172–1183. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2019.2952353.

[137] **F. Muratore, T. Gruner, F. Wiese, B. Belousov, M. Gienger, and J. Peters:** "Neural Posterior Domain Randomization". en. In: *Proceedings of the 5th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2022, pp. 1532–1542.

[138] **F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters:** "Robot Learning From Randomized Simulations: A Review". English. In: *Frontiers in Robotics and AI* 9 (Apr. 2022). Publisher: Frontiers. ISSN: 2296-9144. DOI: 10.3389/frobt.2022.799893.

[139] **V. Myers, E. Biyik, N. Anari, and D. Sadigh:** "Learning Multimodal Rewards from Rankings". en. In: *Proceedings of the 5th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2022, pp. 342–352.

[140] **S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko:** "Regularizing Action Policies for Smooth Control with Reinforcement Learning". In: *2021 IEEE*

International Conference on Robotics and Automation (ICRA). ISSN: 2577-087X. May 2021, pp. 1810–1816. DOI: 10.1109/ICRA48506.2021.9561138.

[141] **O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar:** "Multi-Agent Manipulation via Locomotion using Hierarchical Sim2Real". In: arXiv:1908.05224. Oct. 2019. DOI: 10.48550/arXiv.1908.05224.

[142] **O. Nachum, S. (Gu, H. Lee, and S. Levine:** "Data-Efficient Hierarchical Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

[143] **S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone:** "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey". In: *Journal of Machine Learning Research* 21.181 (2020), pp. 1–50. ISSN: 1533-7928.

[144] **S. Nasiriany, H. Liu, and Y. Zhu:** "Augmenting Reinforcement Learning with Behavior Primitives for Diverse Manipulation Tasks". In: *2022 International Conference on Robotics and Automation (ICRA)*. May 2022, pp. 7477–7484. DOI: 10.1109/ICRA46639.2022.9812140.

[145] **G. Ning, X. Zhang, and H. Liao:** "Autonomic Robotic Ultrasound Imaging System Based on Reinforcement Learning". In: *IEEE Transactions on Biomedical Engineering* 68.9 (Sept. 2021). Conference Name: IEEE Transactions on Biomedical Engineering, pp. 2787–2797. ISSN: 1558-2531. DOI: 10.1109/TBME.2021.3054413.

[146] **OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang:** "Solving Rubik's Cube with a Robot Hand". en. In: *arXiv.org*. Oct. 2019.

[147] **OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang:** "Dota 2 with Large Scale Deep Reinforcement Learning". en. In: *arXiv.org*. Dec. 2019.

[148] **T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters:** "An Algorithmic Perspective on Imitation Learning". English. In: *Foundations and Trends® in Robotics* 7.1-2 (Mar. 2018). Publisher: Now Publishers, Inc., pp. 1–179. ISSN: 1935-8253, 1935-8261. DOI: 10.1561/2300000053.

[149] **R. Özalp, N. K. Varol, B. Taşci, and A. Uçar:** "A Review of Deep Reinforcement Learning Algorithms and Comparative Results on Inverted Pendulum System". en. In: *Machine Learning Paradigms*. Vol. 18. Series Title: Learning

and Analytics in Intelligent Systems. 2020, pp. 237–256. ISBN: 978-3-030-49723-1 978-3-030-49724-8. DOI: 10.1007/978-3-030-49724-8_10.

[150] *PACE Lab - Fraunhofer IML*. en. URL: https://www.iml.fraunhofer.de/en/institute_profile/researchhallslaboratories/pace-lab.html (visited on 04/18/2025).

[151] **J. Parker-Holder, R. Rajan, X. Song, A. Biedenkapp, Y. Miao, T. Eimer, B. Zhang, V. Nguyen, R. Calandra, A. Faust, F. Hutter, and M. Lindauer:** "Automated Reinforcement Learning (AutoRL): A Survey and Open Problems". en. In: *Journal of Artificial Intelligence Research* 74 (June 2022), pp. 517–568. ISSN: 1076-9757. DOI: 10.1613/jair.1.13596.

[152] **S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek:** "Hierarchical Reinforcement Learning: A Comprehensive Survey". In: *ACM Comput. Surv.* 54.5 (June 2021), 109:1–109:35. ISSN: 0360-0300. DOI: 10.1145/3453160.

[153] **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)*. ISSN: 2577-087X. May 2018, pp. 3803–3810. DOI: 10.1109/ICRA.2018.8460528.

[154] **X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne:** "DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning". In: *ACM Trans. Graph.* 36.4 (July 2017), 41:1–41:13. ISSN: 0730-0301. DOI: 10.1145/3072959.3073602.

[155] **X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine:** "Learning Agile Robotic Locomotion Skills by Imitating Animals". en. In: *arXiv.org*. Apr. 2020.

[156] **X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa:** "AMP: adversarial motion priors for stylized physics-based character control". In: *ACM Trans. Graph.* 40.4 (July 2021), 144:1–144:20. ISSN: 0730-0301. DOI: 10.1145/3450626.3459670.

[157] **X. B. Peng and M. Van De Panne:** "Learning locomotion skills using DeepRL: does the choice of action space matter?" en. In: *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. July 2017, pp. 1–13. ISBN: 978-1-4503-5091-4. DOI: 10.1145/3099564.3099567.

[158] **K. Pertsch, Y. Lee, and J. Lim:** "Accelerating Reinforcement Learning with Learned Skill Priors". en. In: *Proceedings of the 2020 Conference on Robot Learning*. ISSN: 2640-3498. Oct. 2021, pp. 188–204.

[159] **K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim:** "Demonstration-Guided Reinforcement Learning with Learned Skills". In: arXiv:2107.10253. July 2021. DOI: 10.48550/arXiv.2107.10253.

[160] **A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield:** "Structured Domain Randomization: Bridg-

ing the Reality Gap by Context-Aware Synthetic Data". In: *2019 International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2019, pp. 7249–7255. DOI: 10.1109/ICRA.2019.8794443.

[161] **I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath:** "Real-world humanoid locomotion with reinforcement learning". In: *Science Robotics* 9.89 (Apr. 2024). Publisher: American Association for the Advancement of Science, eadi9579. DOI: 10.1126/scirobotics.adi9579.

[162] **P. Rajendran:** "Learning Dynamic Whole-body Locomotion with Unknown Payloads". MA thesis. TU Dortmund University, Dec. 2024.

[163] **F. Ramos, R. C. Possas, and D. Fox:** "BayesSim: adaptive domain randomization via probabilistic inference for robotics simulators". In: arXiv:1906.01728. June 2019. DOI: 10.48550/arXiv.1906.01728.

[164] **K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari:** "RL-CycleGAN: Reinforcement Learning Aware Simulation-to-Real". In: 2020, pp. 11157–11166.

[165] **D. Reda, T. Tao, and M. Van De Panne:** "Learning to Locomote: Understanding How Environment Design Matters for Deep Reinforcement Learning". en. In: *Motion, Interaction and Games*. Oct. 2020, pp. 1–10. ISBN: 978-1-4503-8171-0. DOI: 10.1145/3424636.3426907.

[166] **D. Rodriguez and S. Behnke:** "DeepWalk: Omnidirectional Bipedal Gait by Deep Reinforcement Learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2021, pp. 3033–3039. DOI: 10.1109/ICRA48506.2021.9561717.

[167] **S. Ross, G. Gordon, and D. Bagnell:** "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". en. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. ISSN: 1938-7228. June 2011, pp. 627–635.

[168] **N. Rudin, D. Hoeller, P. Reist, and M. Hutter:** "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning". en. In: *Proceedings of the 5th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2022, pp. 91–100.

[169] **L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, M. Walczak, J. Garcke, C. Bauckhage, and J. Schuecker:** "Informed Machine Learning – A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems". In: *IEEE Transactions on Knowledge and Data Engineering* 35.1 (Jan. 2023). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 614–633. ISSN: 1558-2191. DOI: 10.1109/TKDE.2021.3079836.

[170] **L. von Rueden, S. Mayer, R. Sifa, C. Bauckhage, and J. Garcke:** "Combining Machine Learning and Simulation to a Hybrid Modelling Approach: Current and Future Directions". en. In: *Advances in Intelligent Data Analysis XVIII*. 2020,

pp. 548–560. ISBN: 978-3-030-44584-3. DOI: 10.1007/978-3-030-44584-3_43.

[171] **T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever:** "Evolution Strategies as a Scalable Alternative to Reinforcement Learning". In: arXiv:1703.03864. Sept. 2017. DOI: 10.48550/arXiv.1703.03864.

[172] **S. Schaal:** "Learning from Demonstration". In: *Advances in Neural Information Processing Systems*. Vol. 9. 1996.

[173] **J. Schrittwieser, T. Hubert, A. Mandhane, M. Barekatain, I. Antonoglou, and D. Silver:** "Online and Offline Reinforcement Learning by Planning with a Learned Model". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021, pp. 27580–27591.

[174] **J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel:** "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: arXiv:1506.02438 [cs]. Oct. 2018. DOI: 10.48550/arXiv.1506.02438.

[175] **J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov:** "Proximal Policy Optimization Algorithms". en. In: *arXiv.org*. July 2017.

[176] **C. Schwarke, V. Klemm, M. v. d. Boon, M. Bjelonic, and M. Hutter:** "Curiosity-Driven Learning of Joint Locomotion and Manipulation Tasks". en. In: *Proceedings of The 7th Conference on Robot Learning*. ISSN: 2640-3498. Dec. 2023, pp. 2594–2610.

[177] **T. Seyde, I. Gilitschenski, W. Schwarting, B. Stellato, M. Riedmiller, M. Wulfmeier, and D. Rus:** "Is Bang-Bang Control All You Need? Solving Continuous Control with Bernoulli Policies". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021, pp. 27209–27221.

[178] **A. Sharma, A. Gupta, S. Levine, K. Hausman, and C. Finn:** "Autonomous Reinforcement Learning via Subgoal Curricula". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021, pp. 18474–18486.

[179] **R. Siegwart:** *Introduction to autonomous mobile robots*. en. 2nd ed. Intelligent robotics and autonomous agents series. MIT Press, 2011. ISBN: 978-0-262-01535-6 978-0-262-29532-1.

[180] **J. Siekmann, Y. Godse, A. Fern, and J. Hurst:** "Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2021, pp. 7309–7315. DOI: 10.1109/ICRA48506.2021.9561814.

[181] **D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis:** "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (Dec. 2018). Publisher: American Association for the Advancement of Science, pp. 1140–1144. DOI: 10.1126/science.aar6404.

[182] **B. Singh, R. Kumar, and V. P. Singh:** "Reinforcement learning in robotic applications: a comprehensive survey". en. In: *Artificial Intelligence Review* 55.2 (Feb. 2022), pp. 945–990. ISSN: 1573-7462. DOI: 10.1007/s10462-021-09997-9.

[183] **J.-P. Sleiman, M. Mittal, and M. Hutter:** "Guided Reinforcement Learning for Robust Multi-Contact Loco-Manipulation". en. In: (2024). Artwork Size: 17 p. Medium: application/pdf Publisher: ETH Zurich, 17 p. DOI: 10.3929/ETHZ-B-000700491.

[184] **L. Smith, I. Kostrikov, and S. Levine:** "A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning". In: arXiv:2208.07860 [cs]. Aug. 2022. DOI: 10.48550/arXiv.2208.07860.

[185] **P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe:** "Curriculum Learning: A Survey". en. In: *International Journal of Computer Vision* 130.6 (June 2022), pp. 1526–1565. ISSN: 1573-1405. DOI: 10.1007/s11263-022-01611-x.

[186] **R. S. Sutton and A. G. Barto:** *Reinforcement Learning, second edition: An Introduction*. en. Google-Books-ID: uWV0DwAAQBAJ. MIT Press, Nov. 2018. ISBN: 978-0-262-35270-3.

[187] **R. S. Sutton, D. Precup, and S. Singh:** "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial Intelligence* 112.1 (Aug. 1999), pp. 181–211. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(99)00052-1.

[188] **J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke:** "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots". en. In: *arXiv.org*. Apr. 2018.

[189] **J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel:** "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Sept. 2017, pp. 23–30. DOI: 10.1109/IROS.2017.8202133.

[190] **E. Todorov, T. Erez, and Y. Tassa:** "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. ISSN: 2153-0866. Oct. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[191] **Y. Tong, H. Liu, and Z. Zhang:** "Advancements in Humanoid Robots: A Comprehensive Review and Future Prospects". In: *IEEE/CAA Journal of Automatica Sinica* 11.2 (Feb. 2024), pp. 301–328. ISSN: 2329-9274. DOI: 10.1109/JAS.2023.124140.

[192] **L. Torabi:** *NVIDIA Isaac SIM*. Jan. 2021. URL: https://docs.isaacsim.omniverse.nvidia.com/latest/index.html (visited on 04/15/2025).

[193] *Triangular AI at the Lamarr Institute*. Feb. 2024. URL: https://lamarr-institute.org/research/ (visited on 04/06/2025).

[194] **Y.-Y. Tsai, H. Xu, Z. Ding, C. Zhang, E. Johns, and B. Huang:** "DROID: Minimizing the Reality Gap Using Single-Shot Human Demonstration". In: *IEEE Robotics and Automation Letters* 6.2 (Apr. 2021). Conference Name: IEEE Robotics and Automation Letters, pp. 3168–3175. ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3062311.

[195] **O. Urbann, J. Eßer, D. Kleingarn, A. Moos, D. Brämer, P. Brömmel, N. Bach, C. Jestel, A. Larisch, and A. Kirchheim:** "A Large-Scale Dataset for Humanoid Robotics Enabling a Novel Data-Driven Fall Prediction". In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. To be published.

[196] **P. Varin, L. Grossman, and S. Kuindersma:** "A Comparison of Action Spaces for Learning Manipulation Tasks". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Nov. 2019, pp. 6015–6021. DOI: 10.1109/IROS40897.2019.8967946.

[197] B. Vogel-Heuser, T. Bauernhansl, and M. Ten Hompel, eds.**:** *Handbuch Industrie 4.0 Bd.3: Logistik*. de. Springer, 2017. ISBN: 978-3-662-53250-8 978-3-662-53251-5. DOI: 10.1007/978-3-662-53251-5.

[198] **E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter:** "Advanced Skills through Multiple Adversarial Motion Priors in Reinforcement Learning". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. May 2023, pp. 5120–5126. DOI: 10.1109/ICRA48891.2023.10160751.

[199] **N. Wagner, J. Eßer, I. Fachrudin Priyanta, F. Kurtz, M. Roidl, and C. Wietfeld:** "Real-Time Predictive Scheduling for Networked Robot Control Using Digital Twins and OpenRAN". In: *IEEE Globecom, Workshop on Digital Twins over NextG Wireless Networks*. Dec. 2024.

[200] **J. Wang, Y. Liu, and B. Li:** "Reinforcement Learning with Perturbed Rewards". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020). Number: 04, pp. 6202–6209. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i04.6086.

[201] **L. Wang, X. Meng, Y. Xiang, and D. Fox:** "Hierarchical Policies for Cluttered-Scene Grasping With Latent Plans". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022). Conference Name: IEEE Robotics and Automation Letters, pp. 2883–2890. ISSN: 2377-3766. DOI: 10.1109/LRA.2022.3143198.

[202] **L. Wang, Y. Xiang, and D. Fox:** "Manipulation Trajectory Optimization with Online Grasp Synthesis and Selection". In: arXiv:1911.10280. May 2020. DOI: 10.48550/arXiv.1911.10280.

[203] **L. Wang, Y. Xiang, W. Yang, A. Mousavian, and D. Fox:** "Goal-Auxiliary Actor-Critic for 6D Robotic Grasping with Point Clouds". en. In: *Proceedings of the 5th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2022, pp. 70–80.

[204] **M. Wang and W. Deng:** "Deep visual domain adaptation: A survey". In: *Neurocomputing* 312 (Oct. 2018), pp. 135–153. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2018.05.083.

[205] **Z. Wang, A. Novikov, K. Zolna, J. S. Merel, J. T. Springenberg, S. E. Reed, B. Shahriari, N. Siegel, C. Gulcehre, N. Heess, and N. de Freitas:** "Critic Regularized Regression". en. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7768–7778.

[206] **W. Whitney, R. Agarwal, K. Cho, and A. Gupta:** "Dynamics-aware Embeddings". In: arXiv:1908.09357. Jan. 2020. DOI: 10.48550/arXiv.1908.09357.

[207] **N. Wilde, E. Bıyık, D. Sadigh, and S. L. Smith:** "Learning Reward Functions from Scale Feedback". In: arXiv:2110.00284. Oct. 2021. DOI: 10.48550/arXiv.2110.00284.

[208] **J. Wong, V. Makoviychuk, A. Anandkumar, and Y. Zhu:** "OSCAR: Data-Driven Operational Space Control for Adaptive and Robust Robot Manipulation". In: *2022 International Conference on Robotics and Automation (ICRA)*. May 2022, pp. 10519–10526. DOI: 10.1109/ICRA46639.2022.9811967.

[209] **Y.-H. Wu, Z.-C. Yu, C.-Y. Li, M.-J. He, B. Hua, and Z.-M. Chen:** "Reinforcement learning in dual-arm trajectory planning for a free-floating space robot". In: *Aerospace Science and Technology* 98 (Mar. 2020), p. 105657. ISSN: 1270-9638. DOI: 10.1016/j.ast.2019.105657.

[210] **F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese:** "Gibson Env: Real-World Perception for Embodied Agents". In: 2018, pp. 9068–9079.

[211] **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)*. ISSN: 2577-087X. May 2021, pp. 4955–4961. DOI: 10.1109/ICRA48506.2021.9560837.

[212] **H. Xu, Y. R. Wang, S. Eppel, A. Aspuru-Guzik, F. Shkurti, and A. Garg:** "Seeing Glass: Joint Point Cloud and Depth Completion for Transparent Objects". In: arXiv:2110.00087. Sept. 2021. DOI: 10.48550/arXiv.2110.00087.

[213] **T.-Y. Yang, T. Zhang, L. Luu, S. Ha, J. Tan, and W. Yu:** "Safe Reinforcement Learning for Legged Locomotion". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Oct. 2022, pp. 2454–2461. DOI: 10.1109/IROS47612.2022.9982038.

[214] **Z. Yang, K. Merrick, L. Jin, and H. A. Abbass:** "Hierarchical Deep Reinforcement Learning for Continuous Action Control". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.11 (Nov. 2018). Conference Name:

IEEE Transactions on Neural Networks and Learning Systems, pp. 5174–5184. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2018.2805379.

[215] **D. Yarats, D. Brandfonbrener, H. Liu, M. Laskin, P. Abbeel, A. Lazaric, and L. Pinto:** "Don't Change the Algorithm, Change the Data: Exploratory Data for Offline Reinforcement Learning". In: arXiv:2201.13425. Apr. 2022. DOI: 10.48550/arXiv.2201.13425.

[216] **W. Yu, J. Tan, C. K. Liu, and G. Turk:** "Preparing for the Unknown: Learning a Universal Policy with Online System Identification". In: arXiv:1702.02453. May 2017. DOI: 10.48550/arXiv.1702.02453.

[217] **Y. Yu:** "Towards Sample Efficient Reinforcement Learning". en. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*. 2018, pp. 5739–5743.

[218] **J. Zhang, L. Tai, P. Yun, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard:** "VR-Goggles for Robots: Real-to-Sim Domain Adaptation for Visual Control". In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019). Conference Name: IEEE Robotics and Automation Letters, pp. 1148–1155. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2894216.

[219] **T. Z. Zhao, V. Kumar, S. Levine, and C. Finn:** "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware". en. In: arXiv:2304.13705 [cs]. Apr. 2023.

[220] **W. Zhao, J. P. Queralta, and T. Westerlund:** "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2020, pp. 737–744. DOI: 10.1109/SSCI47803.2020.9308468.

A
Appendix

A.1. List of Publications

A.1.1. Related Peer-Reviewed Publications

J. Eßer, N. Bach, C. Jestel, O. Urbann, and S. Kerner: "Guided Reinforcement Learning: A Review and Evaluation for Efficient and Effective Real-World Robotics [Survey]". In: *IEEE Robotics & Automation Magazine* 30.2 (June 2023), pp. 67–85. ISSN: 1558-223X. DOI: 10.1109/MRA.2022.3207664.

J. Eßer, G. B. Margolis, O. Urbann, S. Kerner, and P. Agrawal: "Action Space Design in Reinforcement Learning for Robot Motor Skills". In: *Proceedings of The 8th Conference on Robot Learning*. ISSN: 2640-3498. Jan. 2025, pp. 4021–4032.

P. Klokowski, J. Eßer, N. Gramse, B. Pschera, M. Plitt, F. Feldmeier, S. Bajpai, C. Jestel, N. Bach, O. Urbann, and S. Kerner: "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)*. ISSN: 2153-0866. Oct. 2023, pp. 10425–10432. DOI: 10.1109/IROS55552.2023.10342128.

A.1.2. Peer-Reviewed Publications with Co-Authorship

C. Jestel, K. Rösner, N. Dietz, N. Bach, J. Eßer, J. Finke, and O. Urbann: "MuRoSim – A Fast and Efficient Multi-Robot Simulation for Learning-based Navigation". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. May 2024, pp. 16881–16887. DOI: 10.1109/ICRA57147.2024.10610375.

O. Urbann, J. Eßer, D. Kleingarn, A. Moos, D. Brämer, P. Brömmel, N. Bach, C. Jestel, A. Larisch, and A. Kirchheim: "A Large-Scale Dataset for Humanoid Robotics Enabling a Novel Data-Driven Fall Prediction". In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. To be published.

N. Wagner, J. Eßer, I. Fachrudin Priyanta, F. Kurtz, M. Roidl, and C. Wietfeld: "Real-Time Predictive Scheduling for Networked Robot Control Using Digital

Twins and OpenRAN". In: *IEEE Globecom, Workshop on Digital Twins over NextG Wireless Networks*. Dec. 2024.

A.1.3. Related Blogposts & Interviews

J. Eßer: *Embodied AI Explained: Principles, Applications, and Future Perspectives » Lamarr-Blog*. Jan. 2025. URL: https://lamarr-institute.org/blog/embodied-ai-explained/ (visited on 04/07/2025).

J. Eßer: *Fabriken der Zukunft: Julian Eßer über Reinforcement Learning in der Robotik*. Feb. 2024. URL: https://aigrid-qlcxw4e9ng.live-website.com/interview/fabriken-der-zukunft-ein-gespraech-mit-julian-esser-ueber-reinforcement-learning-in-der-robotik/ (visited on 04/07/2025).

J. Eßer: *Guided Reinforcement Learning – Application to three Robotic Tasks » Lamarr-Blog*. July 2023. URL: https://lamarr-institute.org/blog/reinforcement-learning-application-in-robotics/ (visited on 04/07/2025).

J. Eßer: *Guided Reinforcement Learning – Deployed for Dynamic Locomotion of a Two-Wheeled Robot » Lamarr-Blog*. Oct. 2023. URL: https://lamarr-institute.org/blog/guided-reinforcement-learning-dynamic-locomotion/ (visited on 04/07/2025).

J. Eßer: *Guided Reinforcement Learning – Towards Efficient and Effective Real-World Robotics » Lamarr-Blog*. July 2023. URL: https://lamarr-institute.org/blog/guided-reinforcement-learning-and-robotics/ (visited on 04/07/2025).

J. Eßer: *Introduction to Reinforcement Learning – A Robotics Perspective » Lamarr-Blog*. July 2023. URL: https://lamarr-institute.org/blog/reinforcement-learning-and-robotics/ (visited on 04/07/2025).

J. Eßer: *Reinforcement Learning für Roboter - Belohnung als Anreiz zum Lernen | ROBOTIK UND PRODUKTION*. Section: Anwendungen & Lösungen. Mar. 2024. URL: https://robotik-produktion.de/robotik/belohnung-als-anreiz-zum-lernen/ (visited on 04/07/2025).

J. Eßer: *Reinforcement Learning: Roboter lernen besser mit Belohnung - IT Daily*. Section: Industrie 4.0 & RPA. Feb. 2024. URL: https://www.it-daily.net/it-management/industrie-rpa/roboter-lernen-besser-mit-belohnung (visited on 04/07/2025).

A.1.4. Published Software and Talks

J. Eßer: *Training Code for evoBOT | NVIDIA Omniverse Isaac Gym Environments*. Apr. 2025. URL: https://github.com/julesser/ral-action-spaces/tree/main/omniisaacgymenvs (visited on 04/15/2025).

J. Eßer, F. Feldmeier, and R. Gasoto: *evoBOT Simulation Model for Isaac SIM*. Aug. 2024. URL: https://git.openlogisticsfoundation.org/silicon-economy/simulation-model/evobotsimmodel (visited on 04/07/2025).

J. Eßer and S. Kerner: *Next-Generation Robotics: Training Agile Loco-Manipulation and Human-Machine Interaction | GTC Spring 2024 | NVIDIA On-Demand*. Mar. 2024. URL: https://www.nvidia.com/en-us/on-demand/session/gtc24-s62865/ (visited on 04/07/2025).

J. Eßer and S. Kerner: *Training Highly Dynamic Robots for Complex Tasks in Industrial Applications | GTC Spring 2023 | NVIDIA On-Demand*. Mar. 2023. URL: https://www.nvidia.com/en-us/on-demand/session/gtcspring23-s51823/ (visited on 04/07/2025).

S. Kerner and J. Eßer: *Towards a Digital Reality in Logistics Automation: Optimization of Sim-to-Real | GTC Spring 2022 | NVIDIA On-Demand*. Mar. 2022. URL: https://www.nvidia.com/en-us/on-demand/session/gtcspring22-s42559/ (visited on 04/07/2025).

A.1.5. Supervised Master Theses

A. Anandakrishnan: "Hierarchical Reinforcement Learning for Agile Loco-Manipulation". MA thesis. TU Dortmund University, Nov. 2024.

S. Bajpai: "Sim-to-Real Transfer Methods for a Dynamically Balancing Robot". MA thesis. TU Dortmund University, Feb. 2023.

F. Feldmeier: "Reduction of the Sim-to-real Gap for a Highly Dynamic Robot". MA thesis. TU Dortmund University, Aug. 2024.

P. Rajendran: "Learning Dynamic Whole-body Locomotion with Unknown Payloads". MA thesis. TU Dortmund University, Dec. 2024.

A.2. Open-Source Contributions

A.2.1. Guided Reinforcement Learning (Chapter 3)

Survey Study:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9926159

A.2.2. Case Study: Learning-based Control for evoBOT (Chapter 4)

Simulation Model:
https://git.openlogisticsfoundation.org/silicon-economy/simulation-model/evobotsimmodel

Training Code:
https://github.com/julesser/omni-isaac-gym-envs/tree/main/omniisaacgymenvs/tasks/evobot.py

A.2.3. Generalized Action Space for Robot Control (Chapter 5)

Benchmarking Study:
https://openreview.net/pdf?id=GGuNkjQSrk

Training Code:
https://github.com/julesser/corl-action-spaces

A.3. Detailed Training Setup

A.3.1. Hyperparameters

Table A.1.: Hyperparameters used during training with PPO.

Hyperparameter	**Value**
Timesteps per rollout	48
Minibatches per epoch	5
Mini-batch size	32768
Discount factor	0.99
GAE parameter	0.95
Learning rate	adaptive
KL threshold	0.008
Workers	1
Environments per worker	4096

Table A.2.: Neural network architecture for the Go1 task.

Hyperparameter	**Value**
Hidden layer dimensions	$[512, 256, 128]$
Activation function	`elu`

Table A.3.: Neural network architecture for the evoBOT task.

Hyperparameter	**Value**
Hidden layer dimensions	$[128, 64]$
Activation function	`elu`

A.3.2. Reward Formulations

Table A.4.: Reward for the Go1 task (adapted from [125] to eliminate dependence on action representation, e.g. action change penalty is replaced by corresponding smoothness terms).

Term	Form	Coefficient
Lin vel tracking (xy)	$\exp\{-\lvert\mathbf{v}_{xy} - \mathbf{v}_{xy}^{\text{cmd}}\rvert^2/\sigma_{vxy}\}$	1.0
Ang vel tracking (yaw)	$\exp\{-(\boldsymbol{\omega}_z - \boldsymbol{\omega}_z^{\text{cmd}})^2/\sigma_{\omega z}\}$	0.5
Joint acceleration	$\lvert\ddot{\mathbf{q}}_t\rvert^2$	$-2.5e-7$
Delta torque	$\lvert\boldsymbol{\tau}_t - \boldsymbol{\tau}_{t-1}\rvert^2$	$-5e-8$
Delta pos	$\lvert\mathbf{q}_t - \mathbf{q}_{t-1}\rvert^2$	$-4e-3$
Lin vel (z)	$\mathbf{v}_z^2$	-1.0
Ang vel (roll-pitch)	$\lvert\boldsymbol{\omega}_{xy}\rvert^2$	-0.05
Power	$\lvert\boldsymbol{\tau}_t \cdot \dot{\mathbf{q}}_t\rvert^2$	$-2e-5$
Foot clearance	$\sum_{\text{foot}}(\boldsymbol{h}_{z,\text{foot}}^f - \boldsymbol{h}_z^{f,\text{cmd}})^2 C_{\text{foot}}^{\text{cmd}}(\boldsymbol{\theta}^{\text{cmd}}, t)$	-20.0
Base height	$(\boldsymbol{h}_{\boldsymbol{z}} - \boldsymbol{h}_{\text{ref}})^2$	-80.0
Base orientation	$\lvert\boldsymbol{g}_t\rvert^2$	-0.2
Foot contact (force)	$\sum_{\text{foot}}[1 - C_{\text{foot}}^{\text{cmd}}(\boldsymbol{\theta}^{\text{cmd}}, t)] \exp\{-\lvert\mathbf{f}^{\text{foot}}\rvert^2/\sigma_{cf}\}$	0.2
Foot contact (velocity)	$\sum_{\text{foot}}[C_{\text{foot}}^{\text{cmd}}(\boldsymbol{\theta}^{\text{cmd}}, t)] \exp\{-\lvert\mathbf{v}_{xy}^{\text{foot}}\rvert^2/\sigma_{cv}\}$	0.2
Raibert heuristic	$(\mathbf{p}_{x,y,\text{foot}}^f - \mathbf{p}_{x,y,\text{foot}}^{f,\text{cmd}})^2$	-5.0

Table A.5.: Reward for the evoBOT task (adapted from [99]) to eliminate dependence on action representation, e.g. action change penalty is replaced by corresponding smoothness terms).

Term	Form	Coefficient
Lin vel tracking (x)	$\exp\{-\lvert\mathbf{v}_x - \mathbf{v}_x^{\text{cmd}}\rvert^2/\sigma_{vxy}\}$	1.0
Ang vel tracking (yaw)	$\exp\{-(\boldsymbol{\omega}_z - \boldsymbol{\omega}_z^{\text{cmd}})^2/\sigma_{\omega z}\}$	0.5
Joint acceleration	$\lvert\ddot{\mathbf{q}}_t\rvert^2$	$-1e-4$
Delta torque	$\lvert\boldsymbol{\tau}_t - \boldsymbol{\tau}_{t-1}\rvert^2$	$-1e-4$
Power	$\lvert\boldsymbol{\tau}_t \cdot \dot{\mathbf{q}}_t\rvert^2$	$-1e-4$

A.3.3. Sim-to-Real Conditions

Table A.6.: Observations for the Go1 task.

Term	Dim	Scale	Noise Std
Gravity vector	3	1.0	0.01
Velocity command	3	1.0	0.0
Joint positions	12	0.5	0.005
Joint velocities	12	0.05	1.0
Clock variable	1	1.0	0.0

Table A.7.: Observations for the evoBOT task.

Term	Dim	Scale	Noise Std
Velocity command	2	2.0	0.0
Linear velocity (x)	1	2.0	0.0003
Angular velocity (yz)	2	0.25	0.0005
Pitch angle	1	0.5	0.005
Joint positions	2	0.5	0.001
Joint velocities	2	0.05	0.01

Table A.8.: Domain randomization parameters for the Go1 and evoBOT tasks.

Term	Min	Max
Ground friction	0.5	1.0
Ground restitution	0.0	1.0
Initial joint velocities	-0.1	0.1
Random pushes (every 10 seconds)	-0.5	0.5