# Hierarchical Frameworks for Reinforcement Learning-based Robust and Dynamic Bipedal Locomotion

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Guillermo Castillo Martinez, B.S., M.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2024

# Abstract

This dissertation addresses the integration of reinforcement learning (RL) with model-based methods to develop robust and efficient bipedal locomotion strategies for robots, highlighting the significance of these machines in navigating complex terrains and confined spaces. The quest for advanced bipedal robots is driven by their potential utility across various sectors, including terrain exploration, rescue missions, and assistive technologies. However, the inherent complexity of bipedal motion—marked by high-dimensional models, underactuation, and nonlinear dynamics—presents substantial hurdles in crafting effective motion synthesis techniques.

The core of this research lies in devising innovative, learning-based frameworks that synergize the principles of control theory and the adaptability of RL. By encapsulating the hybrid nature of bipedal locomotion and leveraging reduced-order models, these frameworks aim to surmount the challenges of data efficiency and interpretability in controller design. The dissertation unfolds across a series of contributions that collectively enhance the understanding and capabilities of bipedal locomotion, eschewing a chapter-by-chapter exposition for a thematic discussion of key advances.

A significant portion of the work is dedicated to developing RL-based frameworks that compartmentalize the bipedal locomotion challenge into trajectory planning and feedback regulation. Incorporating insights from the physics of walking, these frameworks reduce the complexity of policy inputs, leading to the creation of variable speed locomotion policies without relying on pre-established reference trajectories. These methodologies underscore the feasibility of implementing sophisticated locomotion strategies in real-world robots, as demonstrated on platforms like the Cassie and Digit robots.

Further, the dissertation extends into the realm of data-driven techniques, proposing a novel integration of low-dimensional state representations with RL for crafting robust locomotion policies. This approach underscores the potential of unsupervised learning techniques in distilling complex dynamics into efficient, interpretable models suitable for real-time application.

The dissertation also focuses on terrain-aware locomotion, with the introduction of a hierarchical framework that adeptly combines an efficient encoded latent representation of the terrain with an RL high-level planner and a model-based low-level controller. This strategy exhibits enhanced robustness and sample efficiency, enabling effective navigation over diverse terrains and demonstrating a considerable advance over traditional model-based or model-free approaches.

The challenge of bridging the simulation-to-reality gap is also addressed, highlighting the indispensable role of simulation in the development and testing of locomotion policies. Successful real-world applications of these policies demonstrate the practical viability of the proposed frameworks and their potential for sim-to-real transfer, a critical consideration for deploying robotic systems in indoor and outdoor environments.

In sum, this dissertation contributes significantly to the field of legged robotics by advancing data-efficient RL frameworks for bipedal locomotion, bridging theoretical and empirical insights with practical applications. These developments not only enhance the adaptability and efficiency of bipedal robots but also open avenues for their deployment in varied real-world scenarios. Future work should aim to integrate perception sensors such as depth cameras and point clouds, broadening their applicability and overcoming the extant challenges in real-world implementation, thereby unlocking the full potential of bipedal and humanoid robots in industrial and societal applications.

To my wife, Linda

To my parents, Mercedes and Eduardo

To my brothers, Eduardo, Cristian, and Bryan

# Acknowledgments

I express my gratitude to God for his countless blessings every day of my life and to all the people who have made this possible through their support and motivation, without whom this thesis would not have been possible.

First and foremost, I would like to thank my advisor, Prof. Ayonga Hereid, for his continuous support, patience, and motivation throughout the Ph.D. program. His insights and direction have been crucial to the development of my research, and his encouragement during challenging times made this achievement possible. I would also like to extend my gratitude to Prof. Wei Zhang for his continuous support, advice, and fruitful discussions.

To my parents, Eduardo and Mercedes, whose love and support have been the bedrock of my journey. Your sacrifices and teachings have shaped every step of my personal and academic growth. To my brothers, Eduardo, Cristian, and Bryan, your unconditional love, support, and encouragement have brought joy and happiness to my life. I am proud of all of you.

To the memory of my grandmother Lucrecia for her valuable teachings and love, and to all my uncles and aunts for their continuous support and encouragement, especially Marianita, Miguel, and Miguel Sebastian, who welcomed me with great love and support and made me feel like part of their family.

To my friends in Ecuador and all over the world, whom I have been lucky to meet and share with. Your sincere friendship, support, and advice have been an essential part of this journey.

To my labmates and research collaborators, whose collaboration and camaraderie have significantly enriched my academic experience. The time we have spent working together has not only been instrumental in the progress of my research but has also provided me with a support network that was crucial during challenging times.

To my wife, Linda, who has made this academic journey a wonderful experience filled with valuable and unforgettable moments—thank you for your love, support, encouragement, and care. Sharing this path with you and witnessing firsthand your dedication and excellence has been one of the greatest privileges of my life. You are not only a constant source of love and support but also an enduring inspiration. I am immensely proud of all that you have accomplished and feel profoundly grateful to share this moment of triumph with you. I look forward to all the adventures and successes our future holds, knowing that together, we can achieve anything. I love you with all my heart!

Finally, I would like to extend my gratitude to the Graduate School at The Ohio State University for awarding me the Presidential Fellowship, which has also partially supported this dissertation, and to the Fulbright Commission for providing me with the opportunity to start my graduate studies in the U.S. I am deeply grateful for this great opportunity, which has changed my life in a personal, academic and professional way, opening the door to new and fulfilling challenges.

In closing, I thank all who have been part of this journey. Your collective support has shaped this experience, turning challenges into achievements and dreams into realities.

# Vita

|  |  |
|---|---|
| 2015 | B.S., Electronics and Control Engineer, Escuela Politécnica Nacional, Quito, Ecuador |
| 2017 - 2019 | Fulbright Scholarship |
| 2017 - present | Graduate Research Associate, The Ohio State University. |
| 2022 - 2023 | Graduate Presidential Fellowship, The Ohio State University. |

## Research Publications

[J3] Weng, Bowen and **Castillo, Guillermo A.** and Zhang, Wei and Hereid, Ayonga, "On the Comparability and Optimal Aggressiveness of the Adversarial Scenario-Based Safety Testing of Robots", *IEEE Transactions on Robotics*, Volume 39, 3299-3318, 2023.

[J2] **Castillo, Guillermo A.** and Weng, Bowen and Zhang, Wei and Hereid, Ayonga, "Reinforcement Learning-Based Cascade Motion Policy Design for Robust 3D Bipedal Locomotion", *IEEE Access*, Volume 10, 20135, 2022.

[J1] Krishna*, Lokesh and **Castillo, Guillermo A.*** and Mishra, Utkarsh A. and Hereid, Ayonga and Kolathaya, Shishir, "Linear Policies are Sufficient to Realize Robust Bipedal Walking on Challenging Terrains", *IEEE Robotics and Automation Letters*, Volume 7, 2047-2054, 2022.

[C9] **Castillo, Guillermo A.** and Weng, Bowen and Yang, Shunpeng and Zhang, Wei and Hereid, Ayonga, "Template Model Inspired Task Space Learning for Robust Bipedal Locomotion", *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[C8] Sony, Raghav and **Castillo, Guillermo A.** and Krishna, Lokesh and Hereid, Ayonga and Kolathaya, Shishir, "MELP: Model Embedded Linear Policies for Robust Bipedal Hopping", *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[C7] Peng, Chengyang and Donca, Octavian and **Castillo, Guillermo A.** and Hereid, Ayonga, "Safe Bipedal Path Planning via Control Barrier Functions for Polynomial Shape Obstacles Estimated Using Logistic Regression", *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 3649-3655, 2023.

[C6] Weng, Bowen and **Castillo, Guillermo A.** and Zhang, Wei and Hereid, Ayonga, "On Safety Testing, Validation, and Characterization with Scenario-Sampling: A Case Study of Legged Robots", *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5179-5186, 2022.

[C5] Krishna, Lokesh and Mishra, Utkarsh A. and **Castillo, Guillermo A.** and Hereid, Ayonga and Kolathaya, Shishir, "Learning Linear Policies for Robust Bipedal Locomotion on Terrains with Varying Slopes", *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5159-5164, 2021.

[C4] **Castillo, Guillermo A.** and Weng, Bowen and Zhang, Wei and Hereid, Ayonga, "Robust Feedback Motion Policy Design Using Reinforcement Learning on a 3D Digit Bipedal Robot", *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[C3] **Castillo, Guillermo A.** and Weng, Bowen and Stewart, Terrence C. and Zhang, Wei and Hereid, Ayonga, "Velocity Regulation of 3D Bipedal Walking Robots with Uncertain Dynamics Through Adaptive Neural Network Controller", *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7703-7709, 2021.

[C2] **Castillo, Guillermo A.** and Weng, Bowen and Zhang, Wei and Hereid, Ayonga, "Hybrid Zero Dynamics Inspired Feedback Control Policy Design for 3D Bipedal Locomotion using Reinforcement Learning", *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[C1] **Castillo, Guillermo A.** and Weng, Bowen and Zhang, Wei and Hereid, Ayonga, "Reinforcement Learning Meets Hybrid Zero Dynamics: A Case Study for RABBIT", *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

## Fields of Study

Major Field: Electrical and Computer Engineering

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

| | |
|---|---|
| **ALIP** | Angular Momentum-based Linear Inverted Pendulum |
| **API** | Application Programming Interface |
| **CBF** | Control Barrier Function |
| **CBR** | Composite Rigid Body |
| **CMAC** | Cerebellum Model Articulation Controller |
| **CNN** | Convolutional Neural Networks |
| **COM** | Center of Mass |
| **COT** | Cost of Transport |
| **CVAE** | Convolutional Variational Autoencoder |
| **DDPG** | Deep Deterministic Policy Gradient |
| **DOF** | Degrees Of Freedom |
| **EoM** | Equations of Motion |
| **ES** | Evolution Strategies |
| **FL** | Feedback Linearization |
| **FK** | Forward Kinematics |
| **FSM** | Finite State Machine |
| **G-HZD** | Generalized Hybrid Zero Dynamics |
| **GRF** | Ground Reaction Force |
| **HL** | High-Level |
| **HZD** | Hybrid Zero Dynamics |

| | |
|---|---|
| **HZDRL** | Hybrid Zero Dynamics Reinforcement Learning |
| **ID-QP** | Inverse Dynamics with Quadratic Programming |
| **IK** | Inverse Kinematics |
| **IMU** | Inertial Measurement Unit |
| **JSON** | JavaScript Object Notation |
| **KL** | Kullback-Leibler |
| **LIP** | Linear Inverted Pendulum |
| **LL** | Low-Level |
| **MCTS** | Monte Carlo Tree Search |
| **MBPO** | Model-Based Policy Optimization |
| **MBRL** | Model-based Reinforcement Learning |
| **MBVE** | Model-Based Value Estimation |
| **MFRL** | Model-free Reinforcement Learning |
| **MLP** | Multi-Layer Perceptron |
| **MPC** | Model Predictive Control |
| **MSE** | Mean Squared Error |
| **NN** | Neural Network |
| **NES** | Natural Evolution Strategies |
| **PCA** | Principal Component Analysis |
| **PD** | Proportional-Derivative |
| **PI** | Proportional-Integral |
| **PPO** | Proximal Policy Optimization |
| **QP** | Quadratic Programming |
| **RL** | Reinforcement Learning |
| **RNN** | Recurrent Neural Network |
| **ROS** | Robotic Operative System |

| | |
|---|---|
| **S2S** | Step-to-Step |
| **SAC** | Soft Actor-Critic |
| **SOTA** | State-of-the-Art |
| **SVG** | Stochastic Value Gradients |
| **TM** | Template Model |
| **TSC** | Task Space Controller |
| **TSID** | Task Space Inverse Dynamics |
| **UDP** | User Datagram Protocol |
| **VAE** | Variational Autoencoder |
| **CVAE** | Convolutional Variational Autoencoder |
| **WPG** | Walking Pattern Generator |

# Chapter 1: Introduction

## 1.1 Motivation

The anthropomorphic design of bipedal and humanoid robots provides unique advantages for navigating challenging terrains and operating in restricted environments. The allure of these robots lies in their potential applications across diverse fields such as terrain exploration, rescue missions, assistive robotics, and more, making them a focal point for both academia and industry. However, the inherent complexity of bipedal locomotion, characterized by high-dimensional models, underactuation, contact constraints, and the nonlinear and hybrid nature of its dynamics, significantly increases the complexity of synthesizing feasible robot motions and has historically hindered the rapid development of biped robots compared to their counterparts, such as quadruped robots.

The journey of bipedal machines traces back to the late 19th century with the introduction of The Steam Man, a pioneering invention by Georges Moore in 1893. This early biped machine, powered by a 0.5 hp gas-fired boiler, astoundingly achieved reasonable walking speeds [5]. As technology evolved, so did the sophistication of bipedal robots. By 1969, Ichiro Kato at Waseda University in Japan unveiled the WAP-1, a robot equipped with artificial rubber muscles powered pneumatically. However, its initial iterations suffered from a glaring limitation - an exceedingly slow pace. This challenge was later addressed with the development of WL-9DR in 1983, a hydraulically actuated robot boasting ten degrees of freedom and larger feet [6].

Throughout the next decades, there was a notable escalation in the creation of two-legged robots, fueled by technological progress and a refined grasp of bipedal mechanics. However, many of these machines faced challenges in dynamic movement and adapting to their surroundings. Notable progress was made in the field during the 80s with the contributions of Mark Raibert's Leg Lab at MIT on dynamic locomotion. Their robots showed impressive agile maneuvers such as jumping, hopping, and running [7].

Over the last few years, recent innovations in the field have given rise to bipeds that can perform intricate dynamic maneuvers, not just limited to walking but also encompassing actions like running, jumping, and adapting to disturbances. A standout instance of such cutting-edge bipeds is the Atlas robot from Boston Dynamics and the Cassie robot from Agility Robotics.

More recently, several renowned companies have joined the race in the development of humanoids and bipedal robots, including Tesla, with the recent unveiling of their humanoid robot Optimus, Apptronik with their robot Apollo, and Figure AI with their version of a humanoid robot. According to the latest market research, in 2022, the bipedal robot market was valued at USD 1.5 billion and is estimated to reach USD 17.3 billion by 2027 [8]. These facts show the increasing popularity of bipedal and humanoid robots and their potential to become active agents in our daily lives.

A *bipedal locomotion controller* or *locomotion policy* takes the information measured by the robot's sensors to compute the torque needed at the robot's joints to achieve a desired control objective while fulfilling some safety and feasibility requirements. Although important progress has been made in bipedal locomotion in terms of mobility and control, bipedal robots still lack the robustness and versatility they need for most real-world applications.

Significant research efforts have been made in the control theory and machine learning communities separately to develop controllers that can exploit the particular features of legged robots, such as underactuation and compliance. However, little work has been conducted to develop a complete framework that integrates machine learning techniques, formal methods from control theory, and insights from the physics of bipedal locomotion. The successful integration of these three components will provide new methods for the development of data-efficient and interpretable learning-based controllers for robust and agile bipedal locomotion.

This thesis addresses the challenges of such integration by studying several learning-based frameworks that combine the potential of reinforcement learning (RL) with model-based methods to design control policies that realize robust and dynamic locomotion in challenging terrains. The research presented in this work attempts to find insightful answers to the questions: **(Q1)** What is the best design choice for a learning framework that provides the best trade-off between model-free and model-based methods for bipedal locomotion? **(Q2)** How can we exploit the benefits of RL-based methods while maintaining high data efficiency? **(Q3)** Can we efficiently integrate perception into RL frameworks for locomotion?

Model-based methods generally rely on optimization techniques that use the full-order model of the robot to accurately capture the underlying dynamics, which yields more natural dynamic walking behaviors. Model-free methods such as RL do not rely on mathematical models but rather on significant amounts of data obtained through the agent's interaction with the environment -generally through simulations- to learn policies that achieve the desired tasks. With the success of deep learning algorithms in tackling challenging control problems, RL-based approaches have been proposed to develop bipedal locomotion controllers. The desired control objectives can be characterized either as reward functions or hard constraints in the RL formulation.

Traditional end-to-end RL methods use policies that take all the information from the robot's sensors and directly output the desired joint torques or angles. However, using RL policies to generate torque outputs can produce non-realistic behaviors [9, 10], while policies that generate joint angles can be challenging to learn from scratch and heavily rely on particular hardware designs [11, 12]. In the last decade, more structured learning frameworks have proposed the use of reference trajectories to guide the learning process. The reference data may be obtained from motion data sets or the solution of model-based optimization problems. There are two main drawbacks to these approaches. First, end-to-end approaches "see" the bipedal system as a black box without considering the underlying dynamics or hybrid nature of walking. The resulting learned policies lack interpretability and are difficult to analyze and fine-tune for deployment on real robots. Second, using the complete robot's state, i.e., all the sensors' information as inputs of the policy, and all the joint torques or angles as the output, results in big-sized policies with hundreds of thousands of parameters.

Without any insights from the system, the policy requires large amounts of data (millions of samples) to learn a stable walking gait [13].

## 1.2 Contributions and Thesis Outline

This dissertation addresses the problems mentioned aforehead by integrating insights from control theory and reduced-order models into the learning process of the policy. In particular, I have proposed sample efficient RL-based frameworks that decouple the problem of bipedal locomotion in a modular structure where we can combine the benefits of RL to exploit the natural dynamics of walking with the robustness of heuristic-based regulations and model-based control design. Thanks to this decoupled structure, we can draw important concepts from control theory largely used in the design of locomotion controllers, such as template models and Hybrid Zero Dynamics.

In Chapter 2, we provide an overview of the bipedal locomotion problem with some important background about the historical development of bipedal and humanoid robots and the principal model-based and learning-based methods used for the design of robust locomotion controllers.

Chapter 3 focuses on the foundations and categorization of reinforcement learning (RL) algorithms, an important group of the learning-based methods mentioned in Chapter 2. The chapter begins with a comprehensive overview and mathematical formulation of reinforcement learning, providing the theoretical background necessary to understand its application in the field of bipedal locomotion. This chapter then looks into the categorization of RL algorithms, which is crucial for identifying the most suitable approaches for our specific applications. The chapter also covers Proximal

Policy Optimization (PPO), a state-of-the-art algorithm known for its balance of sample efficiency and performance, and its relevance to our proposed frameworks. Additionally, Evolution Strategies are discussed, offering an alternative perspective on solving RL problems that could complement traditional methods. This chapter lays a strong foundation for understanding the RL-based frameworks proposed in later chapters, particularly their design choices and the rationale behind selecting specific algorithms for our bipedal locomotion solutions.

In Chapter 4, we propose a sample efficient RL-based framework that decouples the problem of bipedal locomotion in two stages of a cascade structure, trajectory motion planning, and feedback regulation [14, 15]. The motion planning module takes inspiration from the Hybrid Zero Dynamics (HZD), a formal control framework extensively used in the control of bipedal robots [16, 17].

By incorporating the physical insights of bipedal walking, such as its hybrid nature and symmetric motion, along with the virtual constraints, the proposed framework significantly reduces the dimension of the policy input, which is parameterized by a simple neural network (NN). The feedback regulation module compensates the reference trajectories using intuitive and effective regulators that enhance the stability and robustness of the walking gait. To the best of our knowledge, the proposed method produced the first variable speed locomotion policy for a 3D robot that is learned without using previously known reference trajectories or training separate policies for different speeds. The framework was successfully validated in the 3D bipedal robot Cassie, which is a known testbed for advanced control techniques in bipedal locomotion. Moreover, the learned policy is the smallest NN ever reported for dynamic locomotion with the robot Cassie, with about 20 times fewer parameters than other state-of-the-art RL-based approaches applied on the same robot. Finally, we extended the proposed framework to humanoid robots (bipedal robots with arms) and implemented it on real hardware (sim-to-real) using the humanoid robot Digit [18].

Chapter 5 introduces a task space learning-based framework that completely decouples the bipedal locomotion problem into a hierarchical structure with two modules: a task-space high-level planner policy and a tasks-space low-level controller. The high-level policy uses a reduced-order state to learn task-space actions based on the desired objective. Then, the low-level task-space controller takes the high-level policy actions to compute the desired torques at the joint level. The work presented in Chapter 5 improves on some limitations of the methods discussed in Chapter 4, such as the entangled coupling between the high-level policy and low-level controller, the reduced interpretability of the learned policy, and limited locomotion behaviors, e.g., walking at low speeds. Different from traditional end-to-end learning approaches, our HL policy takes insights from the angular momentum-based linear inverted pendulum (ALIP) to carefully design the observation and action spaces of the Markov Decision Process (MDP). This simple yet effective design creates an insightful mapping between a low-dimensional state that effectively captures the complex dynamics of bipedal locomotion and a set of task space outputs that shape the walking gait of the robot. The HL policy is agnostic to the task space LL controller, which increases the flexibility of the design and generalization of the framework to other bipedal robots. This hierarchical design results in a learning-based framework with improved performance, data efficiency, and robustness compared with the ALIP model-based approach and state-of-the-art learning-based frameworks for bipedal locomotion.

In Chapter 6, we explore the integration of data-driven techniques with reinforcement learning (RL) to enhance the control of bipedal robots. Addressing a pivotal question in RL—the optimal selection of observation states- this chapter proposes a novel framework for bipedal walking that integrates a data-driven learned low-dimensional state representation with robust gait planning using RL. Inspired by the hierarchical structure proposed in Chapter 5, our approach utilizes an autoencoder to extract a low-dimensional, effective state representation of the full-order system dynamics. This improvement in state representation learning is coupled with reinforcement learning and a task space feedback controller to develop robust locomotion policies. The contributions of this work are twofold: first, it demonstrates that the complex dynamics of bipedal robots can be effectively captured in a compact, low-dimensional learned latent space. Second, it showcases how these learned latent variables can be employed to learn robust locomotion policies through RL. We highlight the potential of data-driven methods, such as unsupervised learning and representation learning, in extracting relevant features from high-dimensional data, leading to more efficient and interpretable state representations applicable to real humanoid robots.

Chapter 7 introduces the main challenges associated with bipedal locomotion in unstructured terrains and the transition from robust blind locomotion to robust terrain-aware locomotion. An extension of the methods developed in Chapter 5 is presented to integrate the robot's perception of the terrain through exteroceptive feedback. To this end, we propose a hierarchical framework for terrain-aware locomotion that merges an RL-based high-level policy for the real-time generation of task space commands and a model-based low-level controller for task space trajectory tracking. The high-level policy leverages insights from reduced-order template models and latent representations of the robot's surrounding terrain to thoughtfully design the observation and action spaces of the Markov Decision Process (MDP). This hierarchical design shows superior robustness and sample efficiency compared to traditional pure model-based or model-free approaches. The trained policy successfully traverses challenging terrains such as slopes, stairs, and terrains with random shapes and heights while maintaining the walking gait's robustness and stability, even in challenging conditions.

Chapter 8 addresses the challenge of sim-to-real transfer in reinforcement learning, focusing on bipedal robots where real-world data collection is often impractical due to cost and safety concerns. The chapter highlights the benefits of simulation-based training, leveraging advanced physics engines and parallelization for efficient data generation. Central to the discussion is the sim-to-real gap, a key hurdle in applying RL to real-world robotics, especially in the intricate domain of bipedal locomotion. We provide details about our learning pipeline, including important considerations that help the sim-to-real transfer of our trained policies. Furthermore, this chapter shows the implementation details of our controllers in hardware with the robot Digit, with a special focus on the integration of the high-level RL policies and the low-level feedback controllers through the Robotic Operative System (ROS), showing the successful sim-to-real transfer of the policies trained in simulation to the hardware Digit. We demonstrate the robustness of the policies to external disturbances and unstructured environments, which makes our frameworks feasible to deploy on real hardware and intense testing conditions. The chapter concludes by emphasizing the importance of consistent and standardized methods in disturbance rejection and robustness, showcasing a preliminary study with a linear impactor for safety performance testing in legged robots. These contributions are pivotal in advancing the field towards the reproducibility of research and more reliable and standardized approaches in robotic locomotion.

Finally, some conclusions regarding the methods proposed in this dissertation along with future directions of inquiry are presented in Chapter 9.

I expect the methods proposed in my dissertation to contribute to the development of data-efficient RL frameworks that produce safe, robust, and interpretable policies for bipedal locomotion. The ultimate goal of my research is to unlock the potential of bipedal and humanoid robots to navigate robustly in the real world. This achievement will enable the safe deployment of these systems in essential areas of our society with applications in industry, human-robot cooperation, caregiving, and emergency response, among others.

# Chapter 2: Literature review

The development of bipedal and humanoid robots is a significant area of research and industry in robotics, combining elements of mechanical engineering, control systems, and artificial intelligence. This chapter examines the progress in the field of bipedal and humanoid locomotion, a journey from basic designs struggling with stability to advanced models capable of imitating complex human movements. This literature review serves a dual purpose: first, to provide a detailed overview of the advancements in bipedal robotic locomotion, and second, to lay the groundwork for the discussions on the latest data-driven methods and reinforcement learning approaches that are at the forefront of current research in robust bipedal locomotion.

In reviewing this history, we aim to understand how bipedal robots have evolved and the various challenges and achievements in this field. This background is essential for appreciating the current state of technology and for setting the context for the newer, data-driven approaches explored in later chapters.

## 2.1 History of bipedal locomotion

Although the origin of bipedal robotics could be traced back to the mid-20th century, there were earlier efforts to replicate the locomotion of humans with automated machines. The earliest forms of such devices would likely be mechanical automata, which date back to ancient civilizations.

### 2.1.1 Early Developments

One of the earliest examples of a walking automaton is the "Mechanical Monk," completed in the 1560s. This automaton, possibly created by Juanelo Turriano, utilized wheels for movement while simulating walking. The monk was driven by a key-wound spring and is capable of walking in the path of a square, imitating prayer movements, and performing gestures such as striking its chest, nodding its head, and kissing a cross [19]. n ancient China, a notable account of an automaton is found in the "Lie Zi" text, which describes a meeting between King Mu of Zhou and a mechanical engineer known as Yan Shi. Yan Shi presented the king with a very realistic and detailed life-size, human-shaped automaton. This figure could walk, move its head, sing, and even make advances toward the ladies in attendance [20]. In 1893 Georges Moore invented The Steam Man, a pioneering biped machine, supported by a carriage-like structure, and powered by a 0.5 hp gas-fired boiler, achieving reasonable walking speeds [5]. In 1939, the engineers from Westinghouse introduced the robot Elektro at the New York World's Fair. Elektro, who also relied on a wheeled base for mobility, could perform 26 different tricks, including "walking" and talking, showcasing the evolving complexity of autonomous machines. While these automata and robots couldn't walk or balance solely on their legs, they represent significant milestones in the quest to replicate human and animal movements, highlighting the extraordinary creativity and engineering prowess of their creators across different civilizations [21].

The first significant step towards bipedal robot locomotion was the WABOT-1, developed by Waseda University, Japan, in 1973, which could walk in a human-like manner, albeit slowly and without the dynamic balance of a human [22]. WABOT-1 could recognize objects by vision, understand voice commands, speak, and manipulate objects with two hands. Although the level of the technologies was not so matured, we can certainly consider WABOT-1 the first humanoid robot.

In the United States, the research at the Massachusetts Institute of Technology (MIT) led to the development of the MIT Leg Laboratory in the 1980s, led by Marc Raibert. Raibert's work focused on dynamic walking and running robots, laying the groundwork for many principles of dynamic locomotion[7]. Raibert's contributions continue to be foundational in contemporary research on humanoid and general legged locomotion. His conceptual frameworks serve as essential building blocks that inspire ongoing research in this field, and various control laws he developed have been integrated into numerous subsequent advancements [23–25].

### 2.1.2 Technological Advancements and Major Milestones

In the 1990s, Japan and the United States saw a surge in the development of bipedal robots with more sophisticated control systems. In Japan Honda's P2 and P3 robots evolved into ASIMO in 2000, a robot that could walk, run, and climb stairs with a smooth, human-like gait, employing real-time adaptive motion planning [26]. Concurrently, Sony's robot QRIO, introduced in 2003, demonstrated advanced bipedal motions and stability through the use of gyro sensors and pressure-sensitive feet [27].

The HRP (Humanoid Robotics Project), run by the Japanese Ministry of Economy, Trade and Industry, made significant strides in the development of humanoid robots. The HRP-2 robot, introduced in 2003, brought significant improvements with the implementation of the Zero Moment Point (ZMP) method for balance, and it was the first human-size humanoid robot that could walk, lie down and get up [28, 29].

In the United States, the 2000s marked the emergence of dynamically advanced bipedal robots. Boston Dynamics, a spin-off company from Raibert's Leg Laboratory continued to push the boundaries of robotic capabilities, particularly in dynamic and agile movements. The company was a pivotal moment in translating academic research into practical, real-world applications. Boston Dynamics' BigDog, funded by DARPA (Defense Advanced Research Projects Agency), showcased exceptional balance and agility in rough terrains, laying the groundwork for their later humanoid robot, Atlas, which achieved impressive feats of balance and agility, including jumping and running [30].

The DARPA Robotics Challenge (DRC) in 2015 was another significant milestone. The DRC was a prize competition intended to speed the development of advanced robotic hardware, software, sensors and control interfaces so that robots might assist humans in responding to future natural and man-made disasters. It led to the development of robots like IHMC's Atlas-based Running Man, KAIST's HUBO, and Tartan Rescue's Chimp, which were designed for disaster response and demonstrated how bipedal robots could navigate complex environments and manipulate objects [31].

### 2.1.3 Current State and Challenges

The current landscape in bipedal robotics features an array of advanced humanoid robots. In the last decade, several companies have joined the race of humanoid robots, with special focus on creating a general purpose robot that can be deployed in any environment to perfom any task. Some of these companies, like Agility Robotics and Apptronik, emerged as spin-offs from research labs. Agility Robotics' Digit is used in different research labs in the United States, and has been recently deployed in warehouses to perform logistic tasks like moving totes and boxes. Apptronik's robot Apollo has also showcased similar applications for warehouses. Other companies, like Toyota, has focused in showcasing their advancements The Toyota T-HR3, introduced in 2017, showcases advancements in human-robot interaction and teleoperation [32].

More recently, several renowned companies announced the development of their own humanoid robots including Tesla, with the recent unveiling of their humanoid robot Optimus, Unitree's robot H1, and Figure AI with their version of a humanoid robot. A complete list of all the companies with their respective robots is shown in Table 2.1 and most of them are shown in Fig. 2.1.

Despite these advancements, challenges remain, particularly in energy efficiency, adaptability to varied terrains, and achieving human-like autonomy and decision-making. The ongoing research in model-based and model-free methods for bipedal and humanoid locomotion aims to address these challenges by creating more adaptable and efficient bipedal robots.

## 2.2 Model-based methods for bipedal locomotion

The main objective of a successful locomotion gait in a bipedal robot is to maintain balance at all times while walking or standing. The problem of gait generation thus consists of planning trajectories that result in a balanced gait and then realizing them on the robot. Two main walking modes are possible: static and dynamic walking. In static walking, the projection of the robot's Center of Mass (CoM) on the ground

| **Company** | **Robot** |
|---|---|
| Agility Robotics | Digit |
| Apptronik | Apollo |
| Agibot | Raise A1 |
| Boardwalk Robotics | Nadia |
| Figure AI | Figure 01 |
| Boston Dynamics | Atlas |
| Fourier Intelligence | GR1 |
| Kepler Exploration Robot Co | Kepler |
| PAL Robotics | Kangaroo |
| Sanctuary AI | Phoenix |
| Toyota | T-HR3 |
| Ubtech | Walker X |
| Unitree Robotics | H1 |
| Xiaomi | Cyberone |
| Xpeng | PX5 |
| 1x | Neo |

Table 2.1: Humanoid robots by company

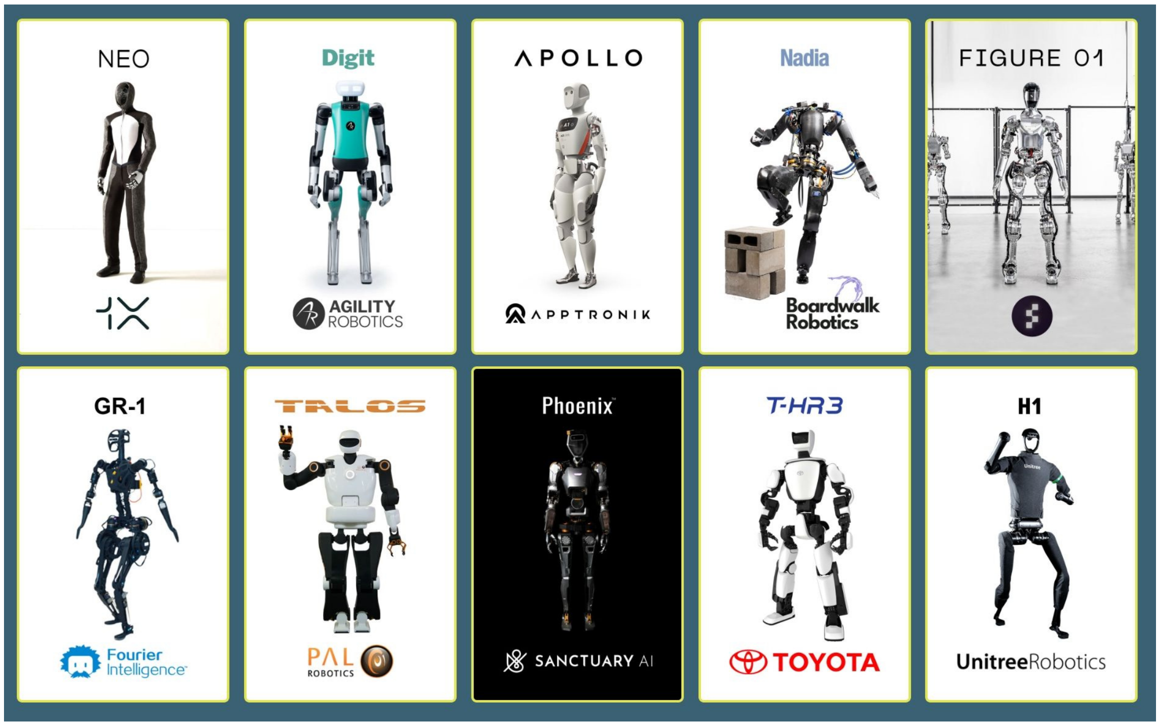

Figure 2.1: Humanoid robots in the industry. Taken from [1]

must always be inside the support polygon, defined as the convex hull of the robot's feet. This results in a slow gait that does not resemble the dynamic way humans walk. The first successful experiments involving bipedal humanoid locomotion were achieved through this mode, e.g., for the WABOT-1 [22].

In dynamic walking, the CoM projection is not constrained to lie above the support polygon of the robot, allowing for faster motions. Here, the dynamic balance must be ensured by keeping the zero momentum point (ZMP) inside the support area. The ZMP, introduced by Vukobratović in 1972 [33], is defined as the point where horizontal components of the moments of the ground reaction forces are equal to zero [29]. This means the robot can maintain balance without falling over while walking or performing other movements.

The introduction of the ZMP marked an important point in the history of bipedal locomotion and opened the door to more dynamic behaviors such as faster walking speeds, running, stair climbing, and even walking with no actuators.

### 2.2.1 Dynamic models of bipedal locomotion and reduced order systems

Dynamic models for bipedal locomotion in robotics are essential for understanding, simulating, and controlling the complex movements of bipedal robots. These models are used to capture the intricate dynamics involved in walking, running, and other locomotion interactions. The way in which the walking system is modeled plays an important role in the planning and control approaches that realize the desired locomotion gaits.

A convenient method for modeling the robotic locomotion system is to construct a representation of the robot in a general position and then enforce ground contacts through forces arising from the associated holonomic constraints that are imposed at the feet. This is often referred to as the floating-base model of the robot [2, 34]. Assuming the robot as a structure of rigid body linkages, the continuous dynamics of a bipedal robot can be represented in the Lagrangian form:

$$D(q)\ddot{q} + H(q, \dot{q}) = Bu + J_h^T(q)\lambda \tag{2.1}$$

$$J(q)\ddot{q} + \dot{J}(q, \dot{q})\dot{q} = 0, \tag{2.2}$$

where $q$ is the vector of generalized coordinates that represents the configuration of the robot, $D(q)$ is the inertia matrix, $H(q, \dot{q}) = C(q, \dot{q})\dot{q} + G(q) + F$ is the vector sum of the Coriolis, centripetal, gravitational, and additional non-conservative forces, $B$ is the actuation matrix, $J_h(q) = \frac{\partial h}{\partial q}(q)$ is the Jacobian of the holonomic constraints $h(q)$, and $\lambda$ are the corresponding wrenches at the holonomic constraints.

However, using the full-order dynamics of the system to plan and execute locomotion trajectories is challenging and computationally intensive due to the high-dimensional, nonlinear, and hybrid nature of legged systems. Therefore, significant efforts have been made in the robotics and biomechanics communities to determine simplified models that allow capturing the most salient features of the full-order model in a reduced set of differential equations. These simplified models are known as reduced-order models (ROMs), and they are crucial for the efficient and robust design of locomotion gaits, particularly for real-time or online applications. The selection of an appropriate reduced-order model is dependent on various factors, including the specific requirements of the intended application, available computational resources, and the constraints inherent to the system.

Fig. 2.2 shows the decomposition of the dynamics models from the most complete e.g., whole body considering the full-order dynamics, to the most simple one, e.g.,

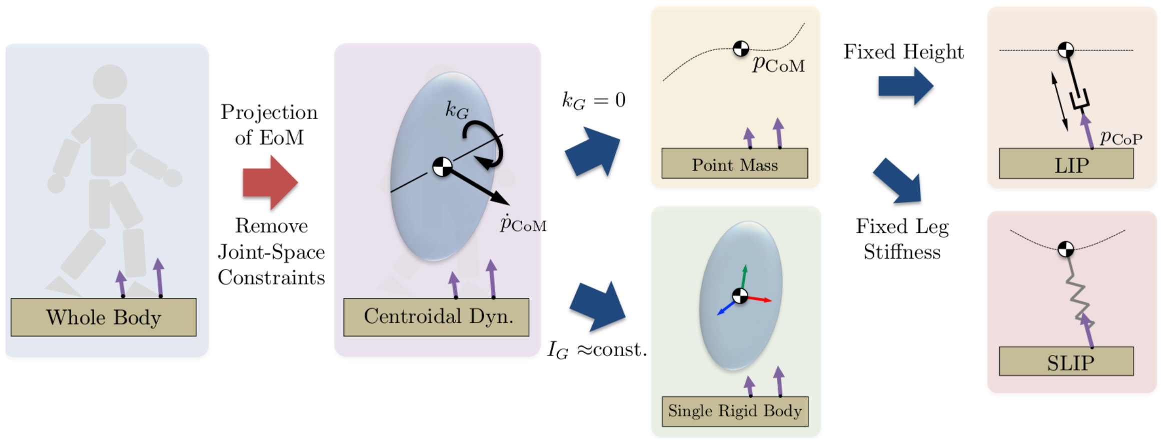

Figure 2.2: Reduced order models (ROMs) used in bipedal locomotion. Published in [2]

linear inverted pendulum (LIP). The centroidal dynamics model considers a reduced problem posed over the trajectories for the CoM position $p_{CoM}$, angular momentum $k_G$ contact locations, and contact forces $\lambda_i$. This order model reduction means losing details about the limbs of the robot by projecting their effects onto the CoM without explicitly considering their kinematics [35, 36]

The single-rigid-body (SRB) model is a simplification of the centroidal dynamics model where the total rotational inertia of the system $I_G$ about the CoM is considered approximately invariant. The SRB model treats the robot as a single, unified mass. It simplifies the robot's body to one rigid entity without articulating its individual limbs or segments while being able to characterize the linear and rotational dynamics that are required to achieve highly aggressive maneuvers [37]. Therefore, this ROM is useful in scenarios where the overall motion of the robot (like jumping or falling) is more critical than the specific movements of its limbs [38]. However, the SRB model is a better choice for systems dynamics where the limbs mass is significantly less compared with the torso mass, which is why it has been mostly used on quadruped robots [38, 38, 39] and more recently in bipedal robots with a hardware design close to that mass distribution such as the robot Cassie from Agility Robotics [37] and the MIT´s humanoid robot [25, 40].

By imposing additional restrictions to the SRB model, such as removing the orientation dynamics, e.g., $K_G = 0$, and keeping the CoM height constant, a simpler but effective model can be realized, the Linear Inverted Pendulum (LIP). The LIP model is a widely used dynamic model for analyzing and designing walking patterns in bipedal robots and humanoid robotics. This model simplifies a walking robot to a point mass (representing the robot's CoM) supported by an inverted pendulum with a a massless telescopic leg that can vary in length but remains always straight [41]. The point of contact with the ground is the Center of Pressure (CoP). These assumptions result in a model with linear differential equations, making it computationally efficient and easier to handle compared to nonlinear systems. The LIP model has been extensively used to simplify the design process for the online generation of the CoM and CoP trajectories for 2D and 3D bipedal walking [42–45]. A common approach in the early days of bipedal locomotion was to use predefined ZMP reference trajectories and generate dynamically feasible trajectories for the robot's CoM [46, 47] Latter approaches proposed the generation of both the ZMP and CoM trajectories with online adaptation methods [48, 49]. Kajita et al. [41] applied the output regulation by the preview control, which is a variation of Model Predictive Control, to the tracking of the referential trajectory of the ZMP. This was a significant step that opened the door for addressing the bipedal locomotion problem through optimization techniques and Model Predictive Control.

## 2.3 Data-driven methods for bipedal locomotion

Data-driven methods for bipedal locomotion in robotics represent a significant shift from traditional model-based approaches, focusing instead on learning or optimizing control strategies directly from data or experience. Although the foundational theory underpinning data-driven algorithms has been established for decades, it is only in recent years, with significant advances in computing power and data availability, that these methods have become feasible for application in more complex systems with high DOF, such as bipedal robots. The increased computational capacity has allowed for the processing of vast amounts of data required by these algorithms, facilitating real-time learning and decision-making.

Additionally, the integration of sophisticated machine learning techniques, including deep learning, has further enabled the effective application of data-driven methods in handling the intricate dynamics and control challenges inherent in bipedal locomotion. This convergence of computational power and advanced algorithmic approaches has not only made it possible to implement these methods in more complex scenarios but also opened up new possibilities for adaptive and intelligent robotic systems that can learn and operate autonomously in a variety of environments.

The literature about data-driven methods for control is vast and, in some cases, can become confusing because of the blurred line between different methods and their applications in particular settings of the bipedal locomotion problem. However, we could distinguish 3 main groups: supervised learning-based approaches, unsupervised learning-based approaches, and reinforcement learning-based approaches.

### 2.3.1 Supervised Learning

In supervised learning, the algorithm is trained on a dataset containing input-output pairs. The goal is to learn a mapping from inputs to outputs, which, in the case of bipedal locomotion, might be mapping sensory inputs to motor actions either as direct target positions or as a parameterized trajectory from which the targets can be derived. In [50], the training and testing sets consist of controllers based on the full dynamic model of the robot, virtual constraints, and parameter optimization to meet torque limits, friction cone, and environmental conditions. Supervised learning is used to extract a low-dimensional state-variable realization of the open-loop trajectories, which is embedded in the original model in a way that makes the desired walking motions locally exponentially stable.

### 2.3.2 Unsupervised Learning

Unsupervised learning, on the other hand, learns patterns and structures from data that does not have labeled outputs. The focus is on discovering the underlying structure of the data. Mitchel et al. [51] use an unsupervised learning approach to train a variational autoencoder to learn a latent space capturing the key stance phases constituting a particular gait for a quadruped robot. A drive signal can be used in the latent space to synthesize a continuous variety of trot styles. Starke et al. [52] extracts a multi-dimensional phase space from full-body motion data, which results in a periodic embedding that can be used to synthesize neural motion controllers for a diverse set of tasks, including locomotion skills.

Supervised and unsupervised learning methods have also been applied to enhance the performance of dynamic model-based controllers. This approach allows the designer to choose particular modules of their system architecture to improve through the harnessing of collected data, which makes the method tailored to very specific applications. Early research work from Hu et al. [53, 54] proposed a self-organizing CMAC (Cerebellum Model Articulation Controller) neural network mechanism to identify the unmodelled dynamics of a planar bipedal robot and a Virtual Model Controller ensuring asymptotic system stability in a Lyapunov sense.

More recently, [55–57] use data collected from simulation and hardware experiments gaits to optimize existing controller parameters. Ahn et al. [58] train a centroidal inertia network to use a Composite Rigid Body (CBR) model within a trajectory optimization framework. In [59], a step-to-step linear dynamics model is learned through supervised learning from the undisturbed walking behaviors of the robot. This model is then used to synthesize a controller with the walking step size as the control input. This approach is extended in [60], where a data-driven representation of the robot's step-to-step dynamics is learned online through classical adaptive control methods. The desired discrete foot placement on the robot is thereby realized by proper continuous output synthesis capturing the data-driven S2S controller coupled with a low-level tracking controller. A similar approach is presented in [61] where an unsupervised learning approach is used to implement a neural network-based adaptive term to reduce the model mismatch between the template LIP model and the physical robot. The adaptive foot planner is then combined with an online QP-based inverse kinematics solver to realize stable and periodic walking for the robot Digit in both simulation and hardware.

### 2.3.3 Reinforcement Learning

The main limitation of supervised and unsupervised learning-based methods is that they require significant amounts of data, and they may suffer from generalization limitations to scenarios outside of the data distribution. Moreover, obtaining this data can be time-consuming, expensive, and sometimes impractical. As an alternative, Reinforcement Learning-based approaches can discover solutions and locomotion strategies on their own, which is particularly useful in situations where labeled data is scarce or difficult to obtain. Unlike supervised learning, RL does not require a labeled dataset. Instead, it learns from the consequences of its actions through interaction with an environment, receiving rewards or penalties. In chapter 3, we explain the foundations of Reinforcement Learning in detail.In this subsection, we intend to provide a general overview of how these learning-based approaches have been applied in the field of bipedal locomotion.

Even though Reinforcement Learning has been applied to bipedal locomotion for decades [62–65], its popularity has increased significantly during the last years, showing impressive results for legged robots in both simulation and hardware experiments.

Some early approaches used RL to learn or modify walking patterns for simple planar bipedal robots using Central Pattern Generator-based controllers [62] or Poincaré maps [65, 66]. By exploiting the mechanical design of passive dynamic walker robots, Tedrake et al. achieve dynamic walking in a 3D robot by learning a feedback linear control policy updated with a stochastic policy gradient algorithm [12, 64].

One of the main limitations in the earlier stage of RL was the scalability of the algorithms for systems with high-order degrees of freedom and continuous action spaces. With higher degrees of freedom, the dimension of the state grows significantly, which results in significantly more complex policies that require intense computational resources. With the recent development in computing technology and learning algorithms, state-of-the-art policy gradient methods have been applied to the bipedal locomotion problem to find policies that map from the observation space to the action space in order to achieve a continuous walking motion. This learning setup is known as end-to-end learning.

In an end-to-end learning framework for a bipedal robot system, the observation space generally consists of the base position, orientation, velocity and angular velocity, and the joint's position and velocity, while the action space can be the joint's motor torques or the joint target positions. Some of the initial breakthroughs of RL for locomotion use end-to-end approaches with policy gradient algorithms such as PPO and DDPG to learn torque output policies, showing impressive performance in simulation [9], [67].

However, directly learning torques can produce undesired behaviors and unnecessary stress on the robot's joints. Even small errors in the input can lead to significant and potentially harmful actions, resulting in policies with high risk of unstable or erratic behavior. Although there have been some efforts to improve these issues by using pre-trained torque networks with gravity compensation [68], most of the state-of-the-art end-to-end learning frameworks use the joint target position as the policy output.

Position output-based end-to-end learning approaches demonstrated some of the first impressive results in hardware with the quadruped robot Anymal [69] and the bipedal robot Cassie [13, 70]. Some of these approaches rely on prior knowledge of a good reference trajectory for the robot joints either by learning small compensations added to the reference trajectory [70] or by using the reference trajectory as part of the reward function [13, 71]. The latter approach, known as imitation learning, has been extensively used in the computer animation community, exhibiting promising results to realize policies that generalize to different tasks without the burden of custom-designed rewards for each particular task. The generalization advantages of imitation learning were pushed forward with the introduction of adversarial motion priors by Peng et al. [72], which uses Generative Adversarial Imitation Learning [73] to allow single policies to perform different tasks while imitating behaviors from large unstructured motion datasets [72, 74] and even generalize to behaviors that are not present in the motion dataset [75, 76].

While some reference trajectories for imitation learning approaches can be obtained from human motion datasets such as [77, 78], they can also be obtained from videos [79] or from solving a trajectory optimization problem with the reduced or full-order model of the robot [37, 80, 81].

Other RL frameworks have proposed to use principled and periodic reward functions based on the physics nature of the locomotion problem to train reference-free policies that learn whole-body push-recovery strategies [82], or diverse locomotion gaits for walking, running, galloping and hopping [83], showing remarkable hardware results in the robot Cassie.

The work in [13, 83] established the foundation for several end-to-end learning bipedal locomotion frameworks with goal-oriented tasks such as velocity tracking [13], high-speed running [84], fast 90°turning [85], highly-dynamic maneuvers [37], robust walking under unsensed dynamic loads [86], walking across stepping stones [87, 88], loco-manipulation [89], and perceptive locomotion on irregular terrains [90].

While end-to-end RL has achieved impressive results, it also has some important limitations, including the lack of interpretability of the trained policy, high computational demands, and challenges in ensuring safe exploration during training. Moreover, end-to-end RL methods combined with deep neural networks can be sampling inefficient (millions of data samples) and are usually over-parameterized (thousands of tunable parameters) as they do not consider the underlying physics of bipedal walking. To address these challenges, some works have proposed to combine RL with model-based kinematic or dynamic controllers commonly used in the field of locomotion. The main idea behind these combined frameworks is to reduce the complexity of the learning problem by integrating control frameworks for legged locomotion, such as hybrid zero dynamics, inverse kinematics, inverse dynamics, and whole-body controllers.

Duan et al. [91] use RL learn task space actions, i.e. the policy outputs residual setpoints in the task space of the feet relative to the robot base. The residual setpoints are added to reference signals, and the regulated trajectories are tracked by an inverse dynamics control of the robot's feet relative to the base. In [92], a 3D reactive stepper is proposed to approximate the 3D capture regions for the full robot dynamics using deep RL. A Q-learning algorithm is used along a LIPM-based stepper to learn the optimal next step location for a biped robot with point feet. The CoM trajectories are generated using the nonlinear inverted pendulum dynamics, and the task space trajectories are tracked using a whole-body controller. Ahn et al. proposed a safe and data-efficient learning framework by combining a walking pattern generator (WPG) with a neural network and a safety controller. The gait generation system employs an analytical approach based on the WPG that enhances efficient learning, while the neural network is optimized to maximize long-term benefits. Concurrently, the safety controller encourages safe exploration by utilizing step capturability and control-barrier functions [93]. The work in [94] introduces a learning framework for humanoid loco-manipulation by combining imitation learning from human-teleoperated demonstrations in task space form with a whole body controller that generates the torques to execute the target commands while complying with the robot dynamics.

Most of the work presented in this dissertation fits in this category. We have proposed different architectures that borrow ideas from control techniques for legged locomotion to exploit the nature of the walking dynamics to reduce the policy complexity, improve the sample efficiency of learning the learning process, and realize robust controllers for bipedal walking under challenging terrains. Our work in [18] borrows ideas from the hybrid zero dynamics theory to propose an RL framework that learns robust walking gaits from scratch for 3D bipedal and humanoid robots by exploiting insights from the hybrid and symmetric nature of dynamic walking to significantly reduce the dimension of policy state and action spaces, improving the sample efficiency of the learning process and robustness of the walking gait. In [95], we present a hierarchical framework that takes insights from the angular momentum-based linear inverted pendulum (ALIP) model to combine an RL-based high-level planner policy for the online generation of task space commands with a model-based low-level controller to track the desired task space trajectories to achieve high-speed locomotion in challenging terrains with slopes up to 20 degrees.

# Chapter 3: Reinforcement Learning

In the quest to achieve robust and efficient bipedal robots, data-driven methods have emerged as a promising avenue. Unlike traditional model-based approaches, which rely heavily on predefined equations of motion and dynamics, data-driven methods leverage vast amounts of data to learn and optimize walking behaviors. Two primary advantages of data-driven approaches are their adaptability to diverse scenarios and their potential to discover novel walking strategies that might not be evident in traditional model-based approaches. To achieve this goal, these methods harness the significant developments achieved in modern computational resources for machine learning and high-fidelity simulators.

This chapter introduces the details of Reinforcement Learning (RL), a major branch of machine learning that focuses on training agents to make sequences of decisions by interacting with an environment. In essence, RL provides a mechanism for an agent to learn to accomplish a specific objective by engaging with its surrounding environment. Instead of being explicitly instructed on the steps to achieve the objective, the agent must determine the most rewarding actions through a process of experimentation. Sometimes, the agent's actions can influence not just the immediate reward but also the agent's next state and all the subsequent rewards. These two characteristics, trial-and-error search, and delayed reward, are the most critical features of reinforcement learning [3], making it one of the most promising fields of research towards generalized artificial intelligence and autonomy.

## 3.1 Overview and Mathematical Formulation

The fundamental components of the reinforcement learning framework include:

**Agent:** The decision-maker in the RL framework. It observes the environment, takes actions based on its policy, and learns from the feedback received in the form of rewards. Usually, the agent is represented by the robot being trained to achieve a particular task.

**Environment:** Everything that the agent interacts with and observes. It responds to the agent's actions by presenting a new state and a reward. In RL, the environment is typically modeled as a Markov Decision Process (MDP), where the probability of transitioning to the next state depends only on the current state and action.

**State ($s$):** A representation of the current configuration of the agent within the environment. In the robotics field, the state is usually determined by the set of measurements obtained from the robot and the environment.

**Action ($a$):** Decisions made by the agent that affect the environment. The set of all possible actions is termed the action space.

**Reward ($r$):** A scalar feedback signal received after taking an action in a particular state. It indicates the immediate benefit of that action.

**Policy ($\pi$):** Defines the agent's behavior. It's a mapping from states $s$ to actions $a$, denoted by

$$\pi(s, a) = \mathsf{P}(a = a | s = s), \tag{3.1}$$

which is the probability of taking an action $a$ given a state $s$, to maximize the total future rewards $r$. In a simple scenario, the policy might be a straightforward look-up table based on discrete states and actions, defined within spaces $\mathcal{S}$ and $\mathcal{A}$, respectively. However, for most applications, creating and learning such a policy is excessively complex and costly, especially when dealing with continuous state and action spaces. Therefore, the policy is usually approximated as an approximate function parameterized by a vector of parameters $\theta$:

$$\pi(s,a) \sim \pi(s,a,\theta) \equiv \pi_\theta(s,a). \tag{3.2}$$

In deep reinforcement learning, this function approximation is generally executed through a deep neural network. This network effectively encapsulates the complex relationship between states, actions, and their outcomes, translating them into a manageable, lower-dimensional representation. This approach allows for efficient learning and decision-making in environments with vast or continuous state and action spaces, where traditional tabular methods are impractical.

**Markov Decision Process (MDP):** The problem of RL is formalized using ideas from dynamical systems theory. As previously stated above, the agent's interaction with the environment can be modeled as a finite MDP. MDPs are a classical formalization of sequential decision-making, where actions influence not just immediate rewards but also subsequent situations or states and, through those, future rewards. [3]. Importantly, this process meets the *Markov property*, which can be described as the independence of the future from the past, given the present. Therefore, an MDP generalizes the notion of a Markov process to include actions and rewards, making it suitable for decision making and control. Formally, a MDP can be represented as the tuple:

$$\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathsf{P}, \mathcal{R}, \xi, \gamma), \tag{3.3}$$

where $\mathcal{S}$ is the feature state space, and $\mathcal{A}$ is the feasible action space. Specifically, as shown in Fig. 3.1 given $s_t \in \mathcal{S}$ at time $t$, the agent takes an action $a_t \in \mathcal{A}$, which takes it into the next state $s_{t+1} \in \mathcal{S}$ according to the transition probability $P$

$$P(s', s, a) = \mathsf{P}(s_{t+1} = s' | s_t = s, a_t = a), \tag{3.4}$$

and the reward function $R$

$$R(s', s, a) = \mathsf{P}(r_{t+1} | s_{t+1} = s', s_t = s, a_t = a). \tag{3.5}$$

Here, $\xi$ denotes the distribution of the initial state $s_0 \in \mathcal{S}$, and $\gamma \in (0, 1)$ denotes the discount factor.

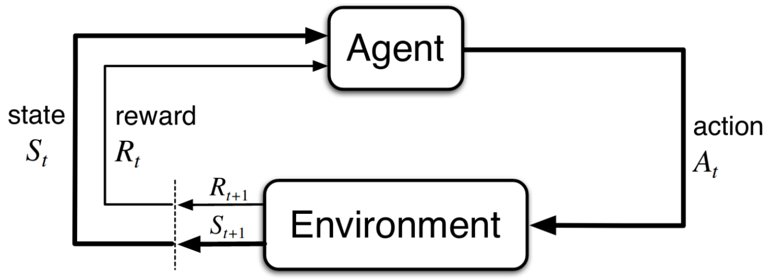

Figure 3.1: Agent-environment interaction in a Markov decision process [3]

**Discount Factor ($\gamma$):** A factor that determines the present value of future rewards. A discount factor close to 0 makes the agent "short-sighted," valuing only immediate rewards, while a factor close to 1 makes the agent "far-sighted," considering long-term rewards.

**Value Function ($V$):** Represents the expected cumulative reward an agent can obtain starting from a particular state. It provides a measure of the "value" or desirability of being in a given state. Given a policy $\pi$, the value function is defined by:

$$V_\pi(s) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s \right], \tag{3.6}$$

where $\mathbb{E}_\pi$ denotes the expected value of a random variable given that the agent follows policy $\pi$ at time $t$.

**Action-Value Function $Q$:** This function, commonly referred to as the Q-function, represents the expected cumulative reward of taking a particular action in a given state and then following a specific policy for all future timesteps.

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s, a_t = a \right], \tag{3.7}$$

The goal of the RL framework is to find an optimal motion policy $\pi^* : \mathcal{S} \to \mathcal{A}$ that maximizes a long-term accumulated reward, which is equivalent to find a policy that maximizes the value function $V(s)$ for all s $\in \mathcal{S}$. Thus, the RL problem can be formally defined as

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s \right], \gamma \in (0,1], \\ \text{subject to} \quad & s_{t+1} = \mathsf{P}(s_t, a_t), \end{aligned} \tag{3.8}$$

One of the most important properties of the value function is that the value at a state $s$ may be written recursively as

$$V_\pi(s) = \mathbb{E}_\pi \left[ R_t + \sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | s_{t+1} = s' \right], \tag{3.9}$$

which results in the optimal value function

$$V_\pi(s) = \max_\pi \mathbb{E}_\pi \left[ R_t + \gamma V_\pi(s') \right], \tag{3.10}$$

where $s' = s_{t+1}$ is the next state after taking action $a_t$ from state $s = s_t$, and the expectation is over actions selected from the optimal policy $\pi$. This equation embodies Bellman's principle of optimality, and it serves as a central result that forms the basis of modern RL.

## 3.2 Categorization of RL algorithms

Reinforcement learning sits at the crossroads of machine learning and control theory, dealing with complex optimization problems. Machine learning optimizes the model parameters to fit training data, while control theory optimizes control performance metrics within dynamic constraints [96]. RL's primary aim is to learn an optimal policy $\pi$ navigating through various approaches like model-based and model-free RL, each with its unique strategies and challenges.

### 3.2.1 Model-base vs model-free RL

A main categorization in RL is given between model-based RL (MBRL) and model-free RL (MFRL) methods. In model-based RL, the agent attempts to learn models of the environment's dynamics alongside the policy. By constructing these models, model-based RL techniques enable the agent to simulate future states and rewards, thus facilitating more informed decision-making. This approach often leads to greater sample efficiency, as the agent can learn from fewer interactions by extrapolating from its internal models.

In model-free RL, the dynamics and reward function are treated as unknown and are not explicitly learned. Instead, the agent's learning focuses on directly deriving the optimal policy or value functions from interactions with the environment. Model-free RL is particularly useful in environments where the underlying dynamics are either too complex to model or unnecessary for learning an effective policy.

While model-based RL methods have higher sample efficiency than their counterpart, their downside is the challenge of model bias, where an agent performs optimally in the learned model but may fail drastically in the actual environment. Conversely, model-free RL methods do not use an environmental model. While they may sacrifice the sample efficiency benefits of model-based methods, they are generally simpler to implement and adapt, which is the reason why they are more popular and widely used for real-world robotics applications.

**Model-based RL algorithms**

Model-based RL methods have many distinct strategies for integrating models into the learning process. This versatility makes it difficult to give a straightforward classification of these methods. Nevertheless, we can differentiate at least three important applications.

***i) Simulation of the environment:*** The world model, i.e., the learned model of the environment, is used for data-augmentation techniques by generating artificial trajectories for the state transitions. DYNA [97] and MBPO [98] are two good examples of this technique.

***ii) Assistance for learning algorithms:*** Leverage smooth functions for the state transition and rewards to perform gradient-based optimization on entire trajectories as shown in Stochastic Value Gradients (SVG) [99] and Dreamer [100, 101]

***iii) Strengthening the policy:*** Leverage internal planners that simulate the environment using the learned world model before picking the best action. Monte Carlo Tree Search (MCTS) [102] is an example of this type of algorithm.

In general, these methods range from pure planning [103] and expert iteration [104, 105] to innovative approaches like data augmentation for model-free methods [106,107] and embedding planning loops into policies [108], demonstrating the versatility and adaptability of model-based RL techniques in various scenarios.

### Model-free RL algorithms

There are three main different ways to apply model-free RL: value-based methods, policy-based methods, and actor-critic methods.

Value-based methods, such as Q-learning and SARSA, determine the policy $\pi(s)$ by estimating the value function $V(s)$ and/or state-value function $Q(s,a)$ [3]. These functions represent the expected total reward from following the current policy $\pi$ starting from the present state. On the other hand, policy-based methods like the REINFORCE algorithm directly learn a parametrized policy $\pi_\theta$, avoiding the need to evaluate $V(s)$ and $Q(s,a)$. Finally, Actor-Critic methods merge these two approaches, using the value function and Q-function estimates to achieve more stable updates of $\pi_\theta$. Examples of this method include Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC).

Value iteration, such as Q-learning [109], takes advantage of the recursive form of the Bellman equation to establish an off-policy algorithm to learn the action-value function $Q^\pi(s_t,a_t)$. While such methods have shown promising performance on complex tasks with high-dimensional state space, their applications in high-dimension action space are limited unless the maximization over action can be efficiently computed during each iteration. For robotic applications with continuous high-dimensional action space, policy gradient methods are more commonly adopted [9].

**Policy gradient methods**

Policy gradient methods are one of the most common and powerful techniques to optimize a policy that is parameterized, as in equation 3.2. Moreover, these methods are well-suited for tasks with continuous action spaces [110]. When the policy $\pi$ is parameterized by $\theta$, it is possible to use gradient optimization on the parameters to improve the policy much faster than through traditional iteration [96]. The parameterization may be a multi-layer neural network, in which case the policy is known as a deep policy network, although other representations and function approximations may be useful. In any case, instead of extracting the policy as the argument maximizing the value or quality functions, it is possible to directly optimize the parameters $\theta$, for example through gradient descent or stochastic gradient descent. The value function $V_\pi(s)$, depending on a policy then becomes $V(s, \theta)$, and a similar modification is possible for the quality function $Q$.

Since the work presented in this dissertation focuses on continuous and high-dimensional action spaces, we use state-of-the-art policy gradient methods that estimate the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{p(x_t;\theta)} \left[ r(x_t) \nabla_\theta \log p(x_t;\theta) \right], x_t = (s_t, a_t), \tag{3.11}$$

either implicitly or explicitly, such as Evolution Strategies (ES) [111] and Proximal Policy Optimization (PPO) [10], respectively. The policy is iteratively improved through simulation rollouts followed by gradient ascent with respect to the objective.

### 3.2.2 Proximal Policy Optimization

Policy gradient methods compute an estimator of the policy gradient in the form of equation 3.11, and plug it into a stochastic gradient ascent algorithm.

PPO is a policy gradient method that proposes a novel objective function that uses multiple epochs of stochastic gradient ascent to perform each policy update. The objective function is of the form

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[ \min(g_t(\theta)A_t, clip(g_t(\theta), 1-\epsilon, 1+\epsilon)A_t) \right], \tag{3.12}$$

where $A_t$ is the so-called advantage estimation [67]. The policy $\pi_\theta(a_t|s_t)$ is improved by a modified probability ratio of $g_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ controlled by the clip ratio $\epsilon$. PPO has the stability and reliability of trust-region methods but is more straightforward to implement and presents better overall performance [10].

### 3.2.3 Evolution Strategies

Evolution strategies are a class of black box optimization algorithms inspired by natural evolution [111] that use a set of candidate solutions (population) to evaluate a problem, updating and improving the set of candidate solutions using random search.

Although ES and RL are seen as distinct categories within the broader field of machine learning and artificial intelligence, both can be applied to similar problems, such as optimization and control, which is the reason why ES has received attention in the last years as a scalable alternative to Reinforcement Learning. However, their methodological approaches are completely different. While RL focuses on learning a direct policy or value function through interaction with the environment, ES is about evolving a population of solutions based on their performance.

In ES, at every iteration (generation), a population of parameter vectors (genomes) is perturbed (mutated) and, optionally, recombined (merged) via crossover. Then, an objective function evaluates the reward (fitness) of each resulting offspring, and through some form of selection, the individuals with higher reward are chosen to produce the individuals in the next generation. More specifically, a population of $N$ policies $\pi(s|\theta_i)_{i=1}^{N}$ are sampled following $\theta_i \sim \mathcal{N}(\theta_\mu, \sigma)$. The normal distribution of policy parameterized on $\theta_\mu$ and $\sigma$ is then improved through estimated gradient using the evaluation results (rewards) from the sampled policies.

This algorithm is repeated until the objective is fully optimized. There are several variations of this algorithm that differ in the method to represent the population and perform mutation and recombination. For the training process in some of the frameworks discussed in this dissertation, we will use a version of natural evolution strategies (NES) developed by OpenAI, which has been applied to standard RL benchmarks [112]. In their work, the stochastic reward experienced over a full episode of agent interaction is represented by a fitness function $F(\cdot)$, and $\theta$ is the vector of parameters of the stochastic policy $\pi_\theta$. The objective function is defined in terms of $\theta$ and it is optimized over $\theta$ directly using stochastic gradient ascent with the score function estimator

$$\nabla_\theta \mathbb{E}_{\epsilon \sim N(0,I)} F(\theta + \sigma\epsilon) = \frac{1}{\sigma} \mathbb{E}_{\epsilon \sim N(0,I)} F(\theta + \sigma\epsilon)\epsilon \tag{3.13}$$

# Chapter 4: Hybrid Zero Dynamics Inspired Reinforcement Learning for Robust Bipedal Locomotion

## 4.1 Introduction and Motivation

Bipedal locomotion's most common learning objective is a feedback control policy that directly maps the state inputs to the torque control output or the joint angles. Typically, this policy is constructed in an end-to-end manner, and the learned policy serves the general purpose of stability maintenance (i.e., walking without falling). Various learning methods have shown effectiveness in learning an end-to-end control policy. Policy gradient-based approaches such as DDPG and PPO have demonstrated competitive performance for general robotic locomotion tasks in simulations with end-to-end learning using torque output policies [9, 10] and real-world experiments (typically combined with dynamics randomization) using torque output-based end-to-end learning [12, 69] and joint angle-based end-to-end learning [11, 113].

Some more advanced methods also seek to achieve velocity tracking [70], push-recovery [82], and walking in various terrain conditions [114] through more structured frameworks. The velocity tracking policy from [70] relies on prior knowledge of a good joint reference trajectory and only learns small compensations added to the known reference trajectory. Siekmann et al. proposed to combine PPO with recurrent neural network (RNN) for learning the direct control policy for Cassie [13]. Some also extend the deep reinforcement learning approach with provided guidance for motion mimicking [71, 115].

Another learning objective is to acquire a reference tracking trajectory of a selected anchor point (e.g., the center of gravity point of the upper torso). A lower-level controller then seeks to track such a learned reference through basic model information such as kinematics. Morimoto et.al. [65, 66] learned the Poincare map of the periodic walking pattern and applied the method to two 2D bipedal robots. Some recent work has proposed to learn the joint-level trajectory as the reference motion through supervised learning [116] and reinforcement learning [14, 15, 117]. The authors in [24] learn linear policies that map the reduced robot's state to parameterized elliptical trajectories for the robot's feet. These approaches often simplify the design of the lower-level tracking, which can be as simple as a PD controller.

Despite the empirical success, most of the aforementioned learning-based approaches are sampling inefficient (millions of data samples) and are usually over-parameterized (thousands of tunable parameters). It is also worth emphasizing that the reference-trajectory-learning approach makes it easier to induce gait symmetry and smooth control signals within the bounded admissible space. On the other hand, the end-to-end approach is difficult to handle symmetry and torque constraints, hence may lead to unnatural walking gaits and sparky control signals [118].

In this chapter, we propose a trajectory-based RL framework to address some of the challenges found in the learning of bipedal locomotion. By decoupling the problem of bipedal locomotion as a two-phase process: trajectory planning and feedback regulation, we propose a modular solution that incorporates the physical insights of dynamic locomotion and its hybrid nature into the learning process of the policy. In particular, we leverage the exploration potential of RL algorithms to find reference trajectories for dynamic locomotion using a reduced state of the robot. Then, we improve these reference trajectories using feedback regulation to obtain stable and robust walking gaits. This decoupled structure significantly simplifies the neural network's complexity, enhancing sampling efficiency and robustness of the learned policy.

A method similar to ours is presented in [119], where the authors propose a decoupled structure that uses DRL to learn a Finite State Machine (FSM) based policy that outputs reference trajectories for particular joints of the robot. A simple linear balance feedback controller is then used on top of the reference trajectories to produce robust locomotion. In our proposed work, we compute continuous joint-space trajectories by means of 5th-order Bézier Polynomials. In addition, we use different high-level commands, e.g., desired velocity tracking, as part of the reduced-order state of our learning framework, whereas [119] uses the full-order state of the robot in addition to desired gait parameters: step length, step duration, and maximum swing foot height during a step.

Our proposed method is evaluated with different robot models, including simulation of the bipedal robots Rabbit, Cassie, and Digit. In addition, we show that the proposed controller structure can be used to transfer the learned policy successfully to hardware with minimal tuning. The resulting controller is extensively tested in hardware with the Digit robot, showing effective velocity tracking performance, and robustness to different disturbances such as external adversarial forces and uneven terrains.

Preliminary results of this work were presented in conference papers [15], [23]. In this work, we extend the preliminary results to further increase the efficiency of the learning method, consider an additional degree of freedom to include constrained arm's motion into the walking gait, include additional regulations to improve the performance of the controller, and perform an extensive series of indoors and outdoors experiments to demonstrate the good performance of the learned policy on hardware. Our contribution can be summarized as follows:

- We propose a complete RL framework to learn robust and stable walking gaits from scratch for 3D bipedal robots. The method takes insights from the hybrid and symmetric nature of dynamic walking to significantly reduce the state and action spaces of the policy, enhancing the sample efficiency of the learning process and robustness of the walking gait.

- We design a regulator policy that uses simple but effective feedback regulators to improve the stability and robustness of the learned walking gait. Different from the earlier conference version, we also develop an estimator of the terrain slope to improve the swing foot orientation regulator, which is the key to successful outdoor experiments. Moreover, we add a stance foot regulation that facilitates velocity tracking on hardware.

- We demonstrate that the proposed framework can be easily extended to robots with different DoF and morphology. We use the proposed learning framework to control the bipedal robot Cassie (no arm joints) and the humanoid robot Digit (with arm joints). The results show the same RL framework learns stable walking gaits for both robots. The results have also been validated extensively in both simulation and hardware.

- We conduct extensive experiments to test the performance of the policy on real hardware, demonstrating the learned policy has a good tracking performance on the desired waking velocity and the desired torso orientation. These results enable the application of the proposed RL framework with confidence for terrain navigation in indoor and outdoor environments. Most of the learning frameworks for bipedal locomotion proposed in the literature do not provide details about the performance of the learned policy for tracking high-level commands like the torso's desired velocity and orientation.

## 4.2 Preliminaries and Problem Formulation

### 4.2.1 Bipedal Robot Model

Bipedal locomotion consists of a collection of phases of continuous dynamics with discrete events that trigger the transitions between these continuous dynamics phases; formally, modeling both continuous and discrete dynamics together results in a hybrid system model. The configuration space $\mathcal{Q}$ of a robot can typically be represented by a floating-base generalized coordinate system, defined as

$$q = [p_b, \phi_b, q_r] \in \mathcal{Q}, \tag{4.1}$$

where $p_b = (q_x, q_y, q_z) \in \mathbb{R}^3$ denotes the relative position of the robot's base, $\phi_b \in SO(3)$ denotes the orientation of the robot's base frame, and $q_r \in \mathbb{R}^m$ denotes the relative angles of articulated joints. Throughout this work, we use $\dot{p}_b = (v_x, v_y, v_z)$ to represent the velocity of the robot's frame, $\phi_b = (q_\psi, q_\theta, q_\phi)$ as the Euler's angle representation (roll, pitch, yaw) of the robot's base orientation, and $\dot{\phi}_b = (\dot{q}_\psi, \dot{q}_\theta, \dot{q}_\phi)$ represents the angular velocity of the robot's base.

Letting $x = (q, \dot{q}) \in \mathcal{X}$ denote the robot states, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ a vector of actuator inputs, and $\omega \in \Omega \subseteq \mathbb{R}^w$ a vector of disturbances and uncertainties, the hybrid system model for bipedal locomotion can be defined as

$$\Sigma : \begin{cases} \dot{x} & = & f(x, u; \omega) & x \notin \mathcal{H} \\ x^+ & = & \Delta(x^-) & x^- \in \mathcal{H}, \end{cases} \tag{4.2}$$

where $f$ represents the continuous dynamics. The switching surface $\mathcal{H}$ is typically the (hyper-) surface of points corresponding to the height of the swing leg above the ground being zero, and $\Delta : \mathcal{H} \to \mathcal{X}$, the reset map or impact map [120], determines the post-impact state values $x^+$ just after switching as a function of the pre-impact state values $x^-$ just before switching.

### 4.2.2 Bipedal Locomotion Problem

In general, the bipedal locomotion problem seeks to establish a motion control policy $\pi : \mathcal{X} \times \mathcal{C} \to \mathcal{U}$ with $\mathcal{C}$ being a set of high-level locomotion commands, such that some properties are achieved. For example, the desired properties may include (i) following commands, (ii) maintaining feasibility condition, (iii) satisfying admissibility condition, (iv) exhibiting naturalistic locomotion, and (v) robustness against uncertainties and disturbances. Here, we mathematically define the aforementioned properties as follows to define the bipedal locomotion problem formally.

**Following Command.** We would like the robot to follow specific high-level commands, such as desired velocities or target locations. In this work, we are particularly interested in velocity tracking, which can be defined as the asymptotic convergence to the desired velocity profile $v^d(t)$, given as,

$$\lim_{t \to \infty} \|\bar{v}(x) - v^d(t)\| = 0, \tag{4.3}$$

where $\bar{v}(x)$ denotes the average velocity over a walking step.

**State Feasibility Condition.** Let $\mathcal{Z} \subseteq \mathcal{X}$ be a set of forbidden states that are prohibitive for the robot. Hence the feasibility criterion is equivalent to ensuring the set $\mathcal{X}^* = \mathcal{X} \setminus \mathcal{Z}$ forward invariant given the dynamics model (4.2), i.e.,

$$\forall x(0) \in \mathcal{X}^* \to x(t) \in \mathcal{X}^*, \quad \forall t \geq 0. \tag{4.4}$$

**Input Admissibility Condition.** Let $\mathcal{U}^*$ be the nominal admissible actuator input set of the robot determined by the actuators' physical capability. The admissibility criterion requires the actuator inputs are persistently feasible, i.e.,

$$\pi(x, c) \in \mathcal{U}^* \quad \forall x \in \mathcal{X}^*, c \in \mathcal{C}. \tag{4.5}$$

**Naturalistic Locomotion.** Moreover, the bipedal applications also expect naturalistic motion for various causes (e.g., environment adaptation and energy efficiency). Examples of naturalistic motion include maintaining the upper-body straight, the Center-of-Mass (CoM) within the support polygon described by the robot feet, avoiding the collision of the robot's feet with each other, etc. In this work, we consider the torso angle limits and the constrained Center-of-Mass (CoM) position to characterize the naturalistic behavior. In particular, let $\theta_{\text{tor}}(x) : \mathcal{X} \to SO(3)$ represents the orientation of the robot's torso, the following constraint is expected to be satisfied:

$$\theta_{\text{tor}}(x) \in \Theta, \quad \forall x \in \mathcal{X}, \tag{4.6}$$

where $\Theta \in SO(3)$ represents the admissible range for the roll, yaw and pitch angles of the robot's torso. In addition, let $p_{\text{com}} : \mathcal{X} \to \mathbb{R}^3$ be the CoM position with respect

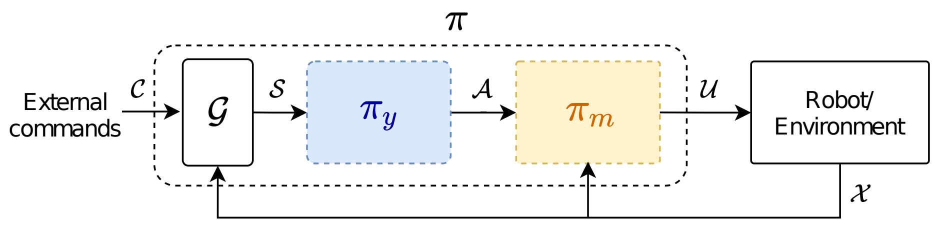

Figure 4.1: The cascaded structure of the proposed motion control policy framework for bipedal locomotion.

to the stance foot in the Cartesian coordinate. The following condition confines the projection of CoM within a enclosed region determined by both feet and the height of CoM within a certain threshold:

$$p_{\text{com}}(x) \in P \quad \forall x \in \mathcal{X}, \tag{4.7}$$

where $P \subset \mathbb{R}^3$ represents the admissible CoM range.

**Problem 1.** ***The Bipedal Locomotion Problem:*** *Consider the robot model in* (4.2)*, the Bipedal Locomotion Problem seeks to establish a motion control policy* $\pi : \mathcal{X} \times \mathcal{C} \to \mathcal{U}$*, such that the criterion defined in* (4.3) - (4.7) *are satisfied with the presence of model uncertainty and external disturbance.*

In practice, solving the above problem is challenging as the hybrid dynamical system in (4.2) is too complex to have a model-based solution that guarantees the satisfaction of all desired properties. Moreover, the various properties specified cannot be satisfied simultaneously in principle (e.g., the velocity tracking requirement may be relaxed in exchange for the safety assurance). In this work, we propose to solve the problem using a cascaded structure that combines reinforcement learning (RL) based motion planning and model-based feedback control design.

### 4.2.3 Cascaded Motion Control Framework

Our proposed approach takes inspiration from the generalized Hybrid Zero Dynamics (G-HZD) framework presented in [50,121]. As shown in Figure 4.1, the motion control policy $\pi$ in Problem 1 consists of a feature selection module, $\mathcal{G}$, and two cascaded policies: a motion policy $\pi_y$ and a feedback control policy $\pi_m$. To clearly identify the objectives of this work, we formally define the proposed motion control framework as follows, where the design of each component will be presented in detail in the following sections.

**Problem 2.** ***Cascade Motion Control Policy Design:*** *The motion control policy $\pi$ in Problem 1 can be designed as*

$$\pi = \pi_m(\cdot) \circ \pi_y(\cdot) \circ \mathcal{G}(\cdot). \tag{4.8}$$

*The feature selection module $\mathcal{G} : \mathcal{X} \times \mathcal{C} \to \mathcal{S}$ maps the full-order states and external commands to a reduced-dimensional feature states $s \in \mathcal{S}$. The motion policy $\pi_y(\cdot) : \mathcal{S} \to \mathcal{A}$ will be designed to generate feasible joint actions $a \in \mathcal{A}$, with $\mathcal{A}$ being the action space, that satisfy the conditions defined in* (4.3) - (4.7)*. Finally, the feedback control policy $\pi_m(\cdot) : \mathcal{X} \times \mathcal{A} \to \mathcal{U}$ converts the joint action commands to admissible actuator inputs with the objective of keeping the robot from falling and simultaneously satisfying* (4.3) - (4.7)*.*

While there are various ways to design the motion policy for bipedal locomotion in literature, our work particularly focuses on reinforcement learning (RL) design approaches [13, 113, 115]. Despite the recent success of RL-based approaches in robust sim-to-real transfer of the policy on robot hardware, existing approaches still suffer from sampling inefficiency and often require prior knowledge of good reference trajectories in training [13, 115]. The proposed trajectory-based RL motion policy design (see Section 4.3) aims to tackle existing limitations of RL-based approaches in bipedal locomotion by incorporating insights from model-based control methods with data-driven reinforcement learning to realize robust bipedal locomotion policies. In addition, an intuitive feedback regulation controller policy (see Section 8.1) is designed to improve the overall robustness of the motion policy.

*Remark* 1. A classic end-to-end RL solution to the bipedal locomotion problem can be considered as a special case of Problem 2. Instead of using the decoupled structure, the end-to-end approaches train a single neural network (NN) policy $\pi(\cdot) : \mathcal{X} \to \mathcal{U}$ that maps the full order states directly to the actuator inputs. However, this approach blindly uses all the data available without insights about the nature or structure of the bipedal locomotion problem, resulting in largely inefficient training with learned policies that are not feasible to be implemented safely in hardware [9].

## 4.3 Motion Policy Design

In this section, we present a sample efficient RL framework for the motion policy design problem described in Section 4.2. The overall structure of the proposed RL-based cascade motion policy is presented in Figure 4.2. We will start with the formal definition of the RL framework for our later discussion. Then we will comprehensively discuss the design of reduced state and action spaces and the specific learning procedure for bipedal locomotion.

### 4.3.1 Reinforcement Learning Framework

As we discussed in Chapter 3, reinforcement learning considers the Markov Decision Process (MDP) as a tuple of components defined by

$$\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathsf{P}, r, \xi, \gamma), \tag{4.9}$$

with the goal is to find an optimal motion policy $\pi^* : \mathcal{S} \to \mathcal{A}$ that maximizes a long-term accumulated reward, defined as

$$J(\pi) = (1-\gamma)\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, \alpha_t, s_{t+1})], \tag{4.10}$$

where $s_t \in \mathcal{S}$ is the observation state of the agent at time $t$, $\alpha_t \in \mathcal{A}$ is the policy action, $r(s_t, \alpha_t, s_{t+1})$ is the reward term, and $\gamma \in (0, 1)$ denotes the discount factor.

To cast bipedal motion policy as a RL problem, one requires (i) adapting the model (4.2) to the MDP form of (4.9), and (ii) configuring the criterion in Problem 1 to align with the RL settings. It is immediate that the probabilistic transition part in (4.9) is equivalent to the described bipedal robot model (4.2). The stochastic transition of the MDP process captures the disturbances and uncertainties $\omega$ such as the random sampling of initial states in the policy training and dynamics uncertainty due to the random interactions with the environment (e.g., early or late ground impacts). Moreover, the desired properties in Problem 1 can be either characterized as rewards or hard constraints in RL. In this work, we formulate criterion (4.3), (4.5), (4.6), (4.7) as rewards, and (4.4) as hard constraints.

### 4.3.2 State space

In our proposed framework, a neural-network motion policy $\pi_y$ maps a feature $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$ via a probability distribution $\pi_y(\cdot|s)$. In particular, the feature can be decomposed into an *endogenous* component, $\zeta$, and an *exogenous* component, $\eta$. The endogenous component $\zeta$ is a reduced dimensional representation of the robot states. The exogenous component $\eta$ corresponds to the external commands, such as desired walking speeds or turning directions, terrain slope, whose transitions will not be affected by the agent through actions [122]. The inclusion of exogenous components enables a single motion policy to capture various locomotion tasks and smooth transitions among these tasks.

**Reduced Dimensional Feature Representation.** Many existing learning-based approaches for bipedal locomotion use the full-order state as the input of the neural network policy, which significantly reduces the sampling efficiency of the training process, resulting in unnecessarily large neural networks and prolonged training time. In this work, we take inspiration from classic model-based approaches in bipedal locomotion to design a lightweight neural network policy structure to improve sampling efficiency and reduce the training time. In particular, we choose as a reduced set of features of the policy the average velocity of the robot's pelvis, the desired velocity of the robot, and the error between the desired and the actual average velocity. This selection is inspired by the Hybrid Zero Dynamics (HZD)-based feedback controllers for bipedal locomotion [16] and the simplicity but effectiveness of the LIP model to provide reference trajectories of the COM and step length [44].

### 4.3.3 Action space

In our motion planning framework, the action determines the parameterized desired joint trajectories. It has been shown that trajectory actions typically provide a

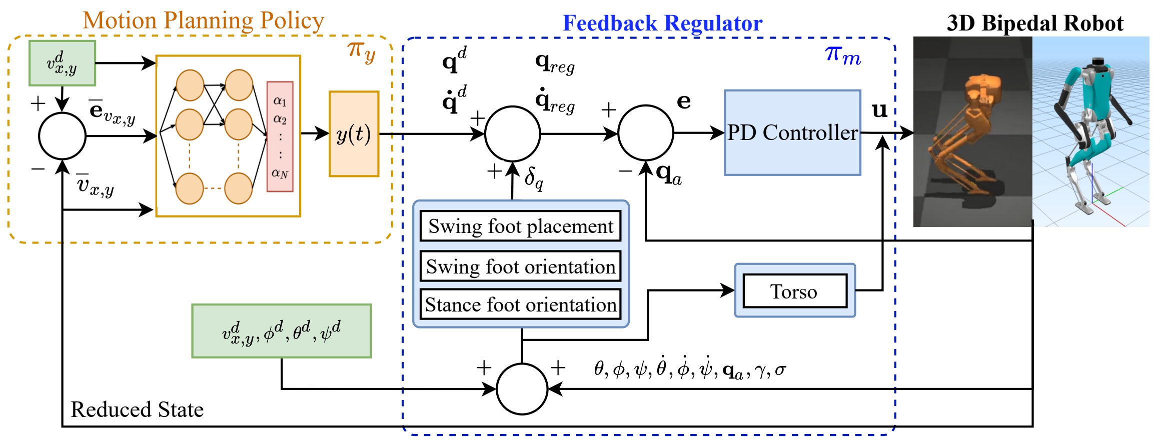

Figure 4.2: Overall structure of the proposed Trajectory-based RL framework. The trajectory planning phase is done by the neural network policy while the feedback regulation block use the robot's sensor information to guarantee the stability of the walking gait and the velocity tracking performance.

better representation of locomotion than the direct actuator inputs [123]. Parameterized trajectories also allow model-free joint references to be tracked by the feedback controller, thereby enabling the seamless sim-to-real transfer of the learned policy on robot hardware.

As discussed later in this section, the motion policy does not need to determine desired trajectories for all actuated joints of the robot. Let $N$ be the number of actuated joints determined by the motion policy $\pi_y$, the desired trajectory of each joint $i \in 0, \ldots, N$ will be parameterized as an $M$-th order Bézier polynomial with coefficients $\alpha_i \in \mathbb{R}^{M+1}$, given as

$$\mathbf{y}^d_i(\tau, \alpha_i) := \sum_{k=0}^{M} \alpha_i[k] \frac{M!}{k!(M-k)!} \tau^k (1-\tau)^{M-k}, \tag{4.11}$$

where $\tau = \frac{t-t^-}{T_{\text{step}}} \in [0,1]$ is the scaled time-based phase variable over one walking step with $t^-$ being the time at the beginning of the step, and $T_{\text{step}}$ is the time duration of one walking step.

**Dimension Reduction of Action Space.** In order to reduce the output size, thereby the overall size, of the neural network policy $\pi_y$, we reduce the action space dimension by incorporating the unique nature of bipedal locomotion.

*Redundant Joints*. The desired trajectory of some actuated joints will be directly commanded by the feedback regulator policy $\pi_m$ described in sec:control. Therefore, the motion policy $\pi_y$ does not need to provide reference trajectories for these joints, significantly reducing the number of outputs required. Specifically, the torso regulation takes care of the stance leg hip roll and pitch joints, the swing foot orientation regulation takes care of the swing ankle roll and pitch joints, and the stance foot regulation takes care of the stance ankle roll and pitch joint. We provided a detailed description of each of these regulations in the following section. Moreover, if arm joints are present (e.g., Digit, see Section 4.5), we can treat the arm as a single pendulum by controlling the motion of the shoulder pitch joint only through the motion policy. Thus, we can lock other arm joints at constant angles, further reducing the policy outputs.

*Gait Symmetry*. For bipedal locomotion, there exists symmetry between the right and left stance gaits. This allows us to learn only the right stance gait parameters and determine the left stance gait parameters using the symmetry condition. Assuming that the set of coefficients for the right stance gait $\alpha^R$ is given, the set of coefficients for the left stance gait $\alpha^L$ can be computed by

$$\alpha^L = \mathbf{T}\alpha^R \tag{4.12}$$

where $\mathbf{T} \in \mathbb{R}^{N \times N}$ is an invertible sparse transformation matrix that captures the symmetry between the robot's joints on the right and left sides.

*Impact Invariance*. To encourage the smoothness of the control actions after the swing foot impacts the ground, we enforce an equality constraint such that at the beginning of every step, the initial point of the Bézier polynomial (determined by $\alpha_i^R[0]$) coincides with the current position of the i-th robot's joint. To determine the switching condition between right and left stances, we detect the impact of the swing foot with the ground by estimating the ground reaction force (GRF) and comparing it with a fixed threshold easily tuned based on experiments performed both in simulation and hardware. Although this threshold is kept fixed during training and evaluation of the policy, early or late contact conditions are indirectly managed by the learned policy through the update of the reference trajectories at the switching conditions. Finally, we enforce the position of the actuated joints to be the same at the end of the right stance and the beginning of the left stance. This encourages continuity in the joint position trajectories after switching the stance foot. When using Bézier polynomials, this condition can be easily enforced through $\alpha_i^R[M] = \alpha_i^L[0]$. Therefore, two Bézier coefficients for each joint can be obtained through the above conditions. This means we only need to find the remaining $M - 1$ coefficients for each of the $N$ reference trajectories, which results in an action space of dimension $N \times M - 1$.

### 4.3.4 Learning Procedure

The proposed framework can use any RL algorithm that handles continuous action spaces, including but not limited to evolution strategies (ES), proximal policy optimization (PPO) [10], and deterministic policy gradient (DDPG) [124]. In this work, we use the ES algorithm because of its simple implementation for parallel processing, and its promising results in environments with a high number of time steps in an episode, actions with long-lasting effects, or with no good estimations available for the value function [112]. All of these conditions are present in the problem of bipedal locomotion.

The reward function adopted in this work is determined by a vector of 9 customized rewards with their respective weights $\mathbf{w}$. Specifically:

$$\mathbf{r} = \mathbf{w}^T [r_{v_x}, r_{v_y}, r_{\mathrm{h}}, r_u, r_{\mathrm{CoM}}, r_{\mathrm{ang}}, r_{\mathrm{angvel}}, r_{\mathrm{fd}}, r_{\mathrm{stf}}]^T. \tag{4.13}$$

These rewards are designed accordingly to the desired properties described in Section 4.2.2 by criteria (4.3) - (4.7). That is, encouraging the policy performance in four sub-tasks: velocity tracking, feasible states (height maintenance), admissible actions (energy efficiency), and naturalistic behavior.

To encourage better **velocity tracking** performance for desired average walking speeds in the longitudinal and lateral direction, rewards $r_{v_x}, r_{v_y}$ are defined as

$$r_{v_x} = \begin{cases} \max\left(\rho_v/(\bar{v}_x - v_x^d + \epsilon), 1\right) & \text{if} \quad |\bar{v}_x - v_x^d| \leq e_{vx} \\ -\rho_v/(\bar{v}_x - v_x^d)^2 & \text{if} \quad |\bar{v}_x - v_x^d| > e_{vx} \end{cases}$$
$$r_{v_y} = \begin{cases} \max\left(\rho_v/(\bar{v}_y - v_y^d + \epsilon), 1\right) & \text{if} \quad |\bar{v}_y - v_y^d| \leq e_{vy} \\ -\rho_v/(\bar{v}_y - v_y^d)^2 & \text{if} \quad |\bar{v}_y - v_y^d| > e_{vy} \end{cases}$$

where $\rho_v$ is a scaling variable that makes the reward function sharp about the desired walking velocity to encourage better velocity tracking, $\epsilon$ is a bias term to prevent singularities when the tracking error is zero, and $e_{vx}, e_{vy}$ are the bounds for the maximum error allowed in the tracking of the desired average velocity.

To encourage the policy to maintain a desired robot's **height**, we define the reward

$$r_{\mathrm{h}} = \begin{cases} \max\{(q_z/q_z^d)^2, 1\} & \text{if} \quad |q_z - q_z^d| \leq e_{qz}, q_z \leq q_z^d \\ \max\{(q_z^d/q_z)^2, 1\} & \text{if} \quad |q_z - q_z^d| \leq e_{qz}, q_z > q_z^d \\ -(q_z - q_z^d)^2 & \text{if} \quad |q_z - q_z^d| > e_{qz} \end{cases}$$

where $q_z^d$ is the desired height and $e_{qz}$ is the maximum error allowed for the height of the robot's base.

The **torque efficiency** reward encourages the learning to reduce the torque applied to the joints.

$$r_u = -\|u\|^2 \tag{4.14}$$

Four rewards are designed to encourage the **naturalistic behavior** of the walking gaits by keeping the center of mass inside the support polygon, keeping the torso upright during the walking motion, and keeping the distance between the feet within a desired nominal range. In particular, (4.15) handles the case when $p_{\text{com}}^{xy}$, the projection of the center of mass on the $xy$ plane, is out of $P$, the area determined by a radius of $0.1m$ about the midpoint between the projection of the two feet on the $xy$ plane, denoted by $Q$.

$$r_{\text{CoM}} = \begin{cases} \rho_d/d & \text{if } \ p_{\text{com}}^{xy} \in P \\ -1/\rho_d(d-0.1)^2 & \text{if } \ p_{\text{com}}^{xy} \notin P \end{cases} \tag{4.15}$$

where $\rho_d$ is a scaling variable, and $d$ is the distance between $p_{\text{com}}^{xy}$ and $Q$.

In (4.16) and (4.17), the torso's angles $(q_\psi, q_\theta, q_\phi)$ and angular velocities $(\dot{q}_\psi, \dot{q}_\theta, \dot{q}_\phi)$ are used to penalize the deviation of the torso from an upright position during the walking motion.

$$r_{\text{ang}} = -(q_\psi^2 + q_\theta^2 + q_\phi^2) \tag{4.16}$$

$$r_{\text{angvel}} = -(\dot{q}_\psi^2 + \dot{q}_\theta^2 + \dot{q}_\phi^2) \tag{4.17}$$

To prevent that the robot's feet spread apart from each other significantly, or the collision of the feet between each other, a penalization to the reward based on the distance between the robot's feet is added in the form of (4.18).

$$r_{\text{fd}} = \begin{cases} -(\Delta_f - \Delta_{f\min})^2 & \text{if} \quad \Delta_f < \Delta_{f\min} \\ -(\Delta_f - \Delta_{f\max})^2 & \text{if} \quad \Delta_f > \Delta_{f\max} \\ 0 & \text{otherwise} \end{cases} \tag{4.18}$$

where $\Delta_{f\min}$ and $\Delta_{f\max}$ are the minimum and maximum desired distance distance between the robot's feet.

Finally, the reward in (4.19) is used to encourage the stance foot to remain static on the ground.

$$r_{\text{stf}} = -\|v_{\text{stf}}\|^2 - \|w_{\text{stf}}\|^2, \tag{4.19}$$

where $v_{\text{stf}} \in \mathbb{R}^3$ and $w_{\text{stf}} \in \mathbb{R}^3$ are the linear and angular velocity of the stance foot.

## 4.4 Feedback Regulator Policy Design

The feedback regulator policy $\pi_m$ modifies some of the trajectories generated by the motion planning policy $\pi_y$ for some of the robot's joints and generates new trajectories for some other joints. This allows the motion planning policy $\pi_y$ to reduce the number of outputs needed to be learned, significantly improving the sample efficiency of the learning framework. The regulations applied are intuitive yet powerful and allow the controller to compensate for uncertainties in the model used for training the high-level planner policy and adapt it to unknown disturbances like external forces or challenging irregular terrains that the learned policy has not experienced during training in simulation. These regulations were originally proposed by Raibert in [125], and they have been applied successfully on the control and balance of legged robots in several works, including [126–129]. As shown in Figure 4.2, the feedback regulations are composed of two submodules: i) trajectory regulations and tracking, and ii) direct torque regulations for torso orientation.

### 4.4.1 Trajectory Regulations and Tracking

Letting $q^d$ be the desired trajectories for the robot's actuated joints provided by the motion policy $\pi_y$, then the regulated trajectories $q^{\text{reg}}$ are determined by

$$q^{\text{reg}} = q^d + \mathbf{A}\delta_q, \tag{4.20}$$

where $\delta_q$ is the vector of compensations applied on top of the trajectories for some of the robot's joints directly related with the swing foot placement, swing foot orientation and stance foot orientation. The matrix $\mathbf{A}$ is an assignation matrix that assigns the compensation term with its corresponding joint. Thus, we will use simple PD controllers to track the regulated reference trajectories at the joint level to compute the torque inputs for the actuated joints of the robot. In this work, the PD controllers are defined as

$$u = -\mathbf{K_p}(q - q^{\text{reg}}) - \mathbf{K_d}(\dot{q} - \dot{q}^{\text{reg}}), \tag{4.21}$$

where $\mathbf{K_p}$ and $\mathbf{K_d}$ are the matrices of PD gains associated with the actuated joints of the robot.

The following joint regulations are applied in this work:

$$\delta_q = \left[\delta_{hr}^{sw}, \delta_{hp}^{sw}, \delta_{hy}^{sw}, \delta_{tp}^{sw}, \delta_{tr}^{sw}, \delta_{tp}^{st}, \delta_{tr}^{st}\right]^T, \tag{4.22}$$

which is determined by:

$$\delta_q = \mathbf{P} \times \mathbf{E} + \mathbf{B}, \tag{4.23}$$

where, $\mathbf{P}$ is a gain matrix, $\mathbf{E}$ is a vector of velocity errors, and $\mathbf{B}$ is a vector of feed-forward correction terms, respectively defined as follows:

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & S_y K^{sw}_{phr} & S_y K^{sw}_{dhr} & 0 \\ K^{sw}_{php} & K^{sw}_{dhr} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ K^{st}_{ptp} & 0 & 0 & 0 & 0 \\ 0 & 0 & K^{st}_{ptr} & 0 & 0 \end{bmatrix}, \tag{4.24}$$

$$\mathbf{E} = \begin{bmatrix} \bar{v}_x - v^d_x \\ \bar{v}_x - v^{ls}_x \\ \bar{v}_y - v^d_y \\ \bar{v}_y - v^{ls}_y \\ q_\phi - q^d_\phi \end{bmatrix}, \tag{4.25}$$

$$\mathbf{B} = \begin{bmatrix} \beta_y & \beta_x & 0 & \xi_{tp} & \xi_{tr} & 0 & 0 \end{bmatrix}^T. \tag{4.26}$$

The underlying motivation of the joint regulators is described as follows. The **swing foot regulations**, i.e., $\delta^{sw}_{hp}$ $\delta^{sw}_{hr}$, and $\delta^{sw}_{hy}$, are originally inspired by the LIP model and has been applied to improve the stability and robustness of model-based feedback controller for 3D bipedal based on the tracking of the average walking speed [125–129]. The compensation for lateral speed regulation $\delta^{sw}_{hr}$ gives a trajectory compensation for the swing leg's hip roll angle. Analogously, $\delta^{sw}_{hp}$, the compensation for the longitudinal speed regulation, outputs a trajectory compensation for the swing leg's hip pitch joint. Moreover, $\delta^{sw}_{hy}$, the compensation for the heading angle of the robot's torso, adds a trajectory compensation to the swing leg's hip yaw angle to keep the torso's yaw orientation at the desired angle. $S_y \in \{1, -1\}$ depends on the swing foot being left or right, $\bar{v}_x$, $\bar{v}_y$ are the longitudinal and lateral average velocities of the robot, $v^{ls}_x$, $v^{ls}_y$ are the velocities at the end of the previous step, $v^d_x$, $v^d_y$ are the reference velocities, and $K^{sw}_{php}, K^{sw}_{dhp}, K^{sw}_{phr}, K^{sw}_{dhr}$ are the proportional and derivative gains of hip pitch and roll joints, respectively. The phase variable $\tau$ is used to smooth the regulation at the beginning of each walking step and reduce torque overshoots. The terms $\beta_x$ and $\beta_y$ are outputs of an additional PI controller used to compensate for the accumulated error in the velocity and prevent the robot from drifting towards a non-desired direction.

The **swing foot orientation regulations**, i.e., $\delta_{tp}^{sw}$ and $\delta_{tr}^{sw}$, are applied to keep the swing foot parallel to the walking surface to ensure a proper landing orientation of the swing foot. These compensations are decoupled for the roll ($\delta_{tr}^{sw}$) and pitch ($\delta_{tp}^{sw}$) joints of the robot's ankle of the swing foot. These regulations are obtained by applying decoupled inverse kinematics (IK) to the robot's leg. Therefore, we represent them as $\xi_{tp}$ and $\xi_{tr}$ in (4.26) as they are dependant on the kinematic tree of the robot and the slope estimation of the walking surface. To estimate the terrain's slope, we assume the stance foot of the robot is aligned with the terrain's surface, and we use the measurements of the robot's IMU and joint angles to compute the orientation of the stance foot through forward kinematics. In Section 4.5, we provide detailed expressions for these regulations.

Finally, the **stance foot orientation regulations**, i.e., $\delta_{tp}^{st}$ and $\delta_{tr}^{st}$, are added to improve the tracking performance of the desired average walking speed. The compensations $\delta_{tp}^{st}$ and $\delta_{tr}^{st}$ are applied to the stance ankle's pitch and roll joints, respectively, to add a trajectory that modifies the current position of these joints.

### 4.4.2 Torque regulations

The torque regulation module applies torque compensations directly to stance hip joints to maintain the desired torso orientation. The **torso regulation** is used to keep the robot's torso in an upright position, which is desired for a natural motion of the walking gait. Assuming that the stance foot is fixed to the ground during the single support phase and that we have a discrete instantaneous impact during the double support phase, the orientation of the torso is directly controlled by the hip roll and hip pitch joints of the robot's stance leg. Therefore, the PD torque regulation denoted by $(u^{st}_{hr})$ and $(u^{st}_{hp})$ can be applied respectively to the hip roll and hip pitch joint of the stance leg to keep the torso upright.

$$\begin{bmatrix} u^{st}_{hr} \\ u^{st}_{hp} \end{bmatrix} = -\begin{bmatrix} K_{p\psi} & K_{d\psi} & 0 & 0 \\ 0 & 0 & S_\theta K_{p\theta} & S_\theta K_{d\theta} \end{bmatrix} \begin{bmatrix} q_\psi - q^d_\psi \\ \dot{q}_\psi - \dot{q}^d_\psi \\ q_\theta - q^d_\theta \\ \dot{q}_\theta - \dot{q}^d_\theta \end{bmatrix}$$

in which, $S_\theta \in \{1, -1\}$ depends on the stance foot being left or right, $q^d_\phi$, $q^d_\theta$, $\dot{q}^d_\phi$, and $\dot{q}^d_\theta$ are desired torso roll and pitch angles and angular velocities, and $K_{p\psi}, K_{d\psi}, K_{p\theta}, K_{d\theta}$ are manually tuned PD gains.

## 4.5 Illustration Examples

In this section, we present the details of the implementation of the proposed framework on an underactuated bipedal robot Cassie and a humanoid robot Digit, both built by Agility Robotics.

**Cassie** has 20 degrees of freedom (DoF) and 10 actuated joints. Each leg has five actuated joints corresponding to the motors located on the robot's hip, knee and ankle, and two passive joints corresponding to the robot's shin and tarsus joints. During the single support phase (only one foot on the ground), the robot is underactuated because of its narrow feet.

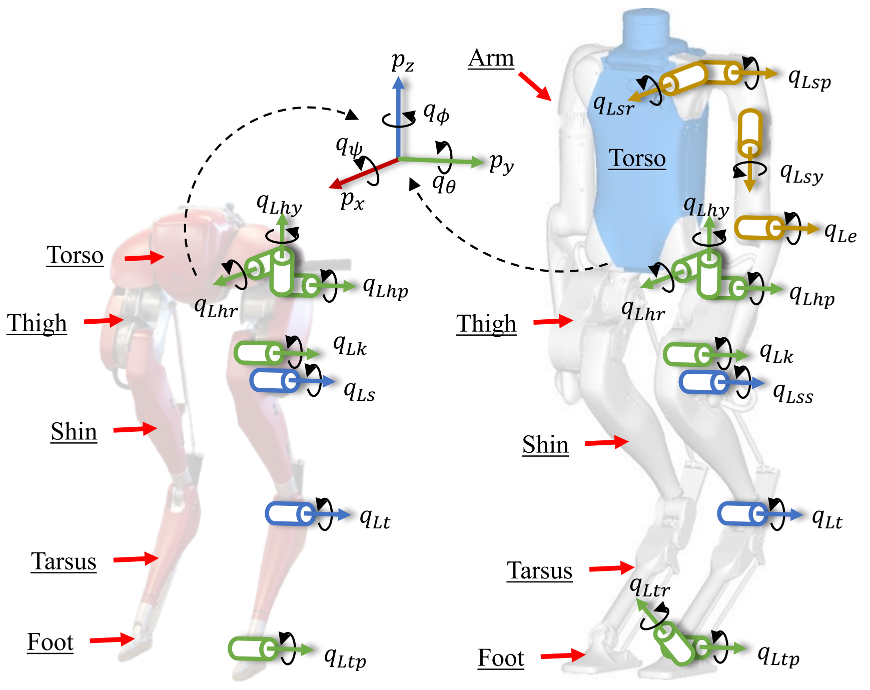

Figure 4.3: Robot models used to test the proposed RL locomotion framework

**Digit** has the same leg morphology as Cassie, with additional joints for the ankle roll, shoulder, and elbow. This makes Digit a more complex system with 30 DoF and 20 actuated joints. Moreover, Digit is equipped with a full stack of vision sensors, including an RGB camera, four depth cameras, and a LiDAR. Figure 4.3 shows the kinematic structure of Cassie and Digit with a description of the notation used for the robot's floating base and joints.

### 4.5.1 State and Action Space

Following the motion policy design presented in Section 4.3.2, the feature state space $\mathcal{S}$ is determined by $s_t = (\eta_t, \zeta_t)$ with $\eta_t = (v_x^d, v_y^d)$, and $\zeta_t = (\bar{v}_x, \bar{v}_y)$, where $(\bar{v}_x, \bar{v}_y)$ are the average longitudinal and lateral velocity, and $(v_x^d, v_y^d)$ correspond to the desired average walking speed. We consider the average speed during one walking

| Description | Cassie | Digit |
|---|---|---|
| dim($\mathcal{S}$) | 6 | 6 |
| Outputs of motion policy (N) | 6 | 7 |
| dim($\mathcal{A}$) | 24 | 28 |
| Hidden layers in NN | 4 | 4 |
| Number of units per layer | 32 | 32 |
| Total number of parameters NN | 4184 | 4316 |

Table 4.1: Details of the state and action space and NN implemented in Cassie and Digit.

step of the robot, which lasts about 400 $ms$ for Cassie and 500 $ms$ for Digit. Similarly, following the considerations discussed in Section 4.3.3, the number of outputs determined for the motion planning policy for Digit is N=7, whereas for Cassie N=6 because we do not have arms motion. More details about the dimension of the state and action spaces are provided in Table 4.1.

### 4.5.2 Neural Network Structure

The structure of the lightweight neural network used in our framework is shown in Figure 4.4, and the details about its parameters are shown in Table 4.1. ReLU activation functions are used between hidden layers, whereas the final layer employs a sigmoid function to limit the range of the outputs. Moreover, Table 4.2 shows a detailed comparison of the NN structure of our method with state-of-the-art RL frameworks for bipedal locomotion. For a fair comparison, we only considered studies implemented on the robot Cassie. Table 4.2 shows the NN is considerably smaller in size, making the proposed RL framework more lightweighted, faster to train, and feasible to implement on real-time controllers even on budget-limited processors. This is the smallest NN implemented in simulation and hardware to realize robust and

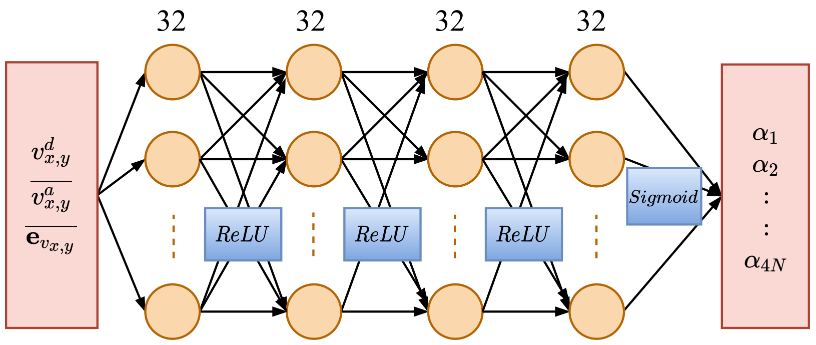

Figure 4.4: Detailed structure of the neural network implemented for the robots Cassie and Digit. By incorporating insights from the symmetry and dynamics of the walking motion, plus simple but effective feedback regulations, we reduce significantly the dimension of the state and action spaces, which results in the smallest NN used for locomotion of real 3D bipedal robots.

| **Method** | **State** | **Action** | **Layers** | **Units** | **Total parameters** |
|---|---|---|---|---|---|
| Ours | 6 | 24 | 4 | 32 | 4184 |
| [70] | 80 | 10 | 2 | 256 | 89098 |
| [13] | 49 | 10 | 2 | 128 | 225034 |
| [113] | 277 | 10 | 2 | 512 | 410122 |

Table 4.2: Comparison of the total number of parameters of the neural network with other learning frameworks for bipedal locomotion with the robot Cassie. The neural network implemented in our method has about 20x fewer parameters when compared with the other methods.

stable locomotion on the 3D bipedal robots Cassie and Digit to the best of our knowledge.

### 4.5.3 Training setup

To train the NN presented in Section 4.5.2 we used the evolution strategies (ES) algorithm [112], using the tuning parameters shown in Table 4.3. Our learning pipeline uses a model-based balancing controller to obtain a pool of initial states that are feasible to be implemented in both simulation and the real robot. We use a customized

| **Parameter** | **Value** |
|---|---|
| Population | 24 |
| Standard deviation | 0.1 |
| Decay standard deviation | 0.9999 |
| Limit standard deviation | 1e-4 |
| Learning rate | 0.01 |
| Decay learning rate | 0.9999 |
| Limit learning rate | 1e-4 |

Table 4.3: Tuning parameters used for training of the policy using Evolution Strategies (ES).

environment using MuJoCo [130], with each episode starting from a robot's state chosen randomly from the pool of balanced initial states and uniformly sampled desired walking velocities. We denote that the trained policy learns to walk from scratch without using previously known reference trajectories or policies pre-trained with expert demonstrations. In Table 4.4, we detail the values of the gains and bounds used for the rewards introduced in Section 4.3.4. We denote that the weight corresponding to $r_{\text{stf}}$, the reward associated with keeping the stance foot static during the step, is equal to zero for Cassie. This reward was added particularly for the Digit because the robot's torso is significantly heavier than Cassie's, which caused Digit's stance foot to slip on the ground. In addition, to encourage policies that realize sustained walking, we increased the episode length from 10000 simulation steps (Cassie) to 15000 (Digit), which are equivalent to 5 and 7.5 seconds, respectively. The episode has an early termination if any of the following conditions are violated:

$$\begin{aligned} &|q_\psi| < 0.5, \quad |q_\theta| < 0.5, \quad |q_\phi| < 0.5, \\ &|\dot{q}_\psi| < 2, \quad |\dot{q}_\theta| < 2, \quad |\dot{q}_\phi| < 2, \\ &0.8 < q_z < 1.2, \quad \Delta_{f\min} < \Delta_f < \Delta_{f\max}, \end{aligned} \tag{4.27}$$

| **Coefficient** | **Cassie** | **Digit** |
|---|---|---|
| $\rho_v$ | 1e$^{-3}$ | 1e$^{-3}$ |
| $\epsilon$ | 1e$^{-5}$ | 1e$^{-5}$ |
| $e_{vx}$ | 0.1 | 0.1 |
| $e_{vy}$ | 0.2 | 0.2 |
| $q_z^d$ | 0.91 | 1.00 |
| $e_{qz}$ | 0.05 | 0.04 |
| $\rho_d$ | 0.01 | 0.01 |
| $d$ | 0.1 | 0.1 |
| $\Delta_{f\min}$ | 0.2 | 0.2 |
| $\Delta_{f\max}$ | 0.4 | 0.4 |
| $\mathbf{w}$ | [0.8, 0.2, 0.1, 0.01, 0.1, 0.5, 0.5, 5, 0]$^T$ | [0.3, 0.3, 0.2, 0.1, 0.5, 0.5, 0.5, 5, 0.5]$^T$ |

Table 4.4: Coefficients and weights used for the rewards during the training of each environment.

| **Parameter** | **Range** | **Unit** |
|---|---|---|
| Link Mass | $[0.85, 1.15]$ | kg |
| Link Center of Mass | $[0.95, 1.05]$ | m |

Table 4.5: Dynamic properties and sample range used for dynamic randomization during training.

where $q_z$ is the height of the robot's pelvis and $\Delta_f$ is the distance between the feet. In addition, we use dynamic randomization in our training process to improve the robustness of the policy and the sim-to-real transfer success. These parameters are shown in Table 4.5.

Figure 4.5 shows the evolution of the normalized mean reward during training for both Cassie and Digit. The number of training episodes needed by the policy to achieve a stable reward is significantly higher in the Digit's environment. This result is expected given the higher level of complexity imposed by the model of the Digit robot.

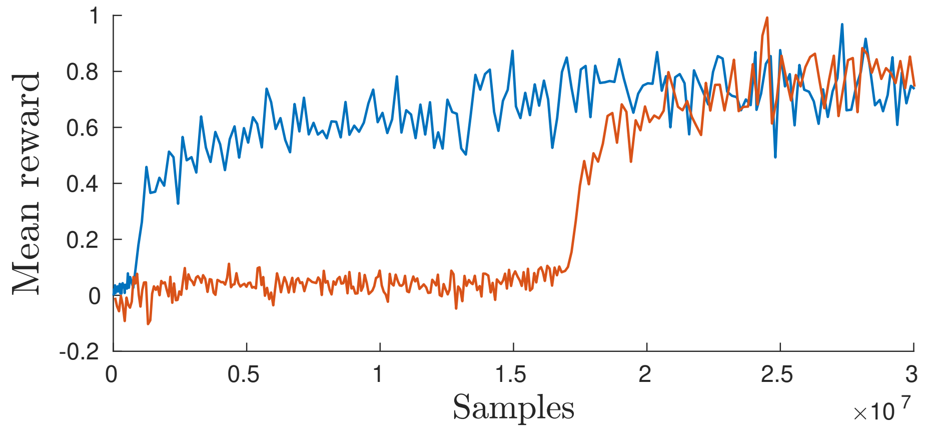

Figure 4.5: Learning process of the trained policy for Cassie (blue) and Digit (red).

Comparing the sample efficiency between different RL frameworks for bipedal locomotion is difficult because of the particular settings used for each training setup (e.g., learning task, episode length, policy update frequency, learning algorithm, prior knowledge of the walking gait, performance of the trained policy). In addition, not all the methods present information about the number of samples required to learn a stable walking gait. However, Table 4.6 shows a comparison between state-of-the-art learning-based frameworks for bipedal locomotion. To promote a fair comparison, we only considered methods that use the bipedal robot Cassie. The results show that our method needs fewer samples than other approaches for the reward to converge to a stable value. In addition, Table 4.6 shows that the proposed framework requires less wall time than other approaches, except for [70], which learns policies that walk at a single desired walking speed using known reference trajectories. On the other hand, our method learns a single policy that tracks various speeds without using known reference trajectories. The policy is trained using a single 12-core CPU machine.

|  | **Task** | **Training time [h]** | **Samples** |
|---|---|---|---|
| Ours | Various speeds | 3 | 0.6e7 |
| [70] | One speed | 2.5 | - |
| [13] | Various speeds | 8 | 1e7 |
| [91] | Various speeds | 16 | - |
| [113] | Various speeds | - | 1e7 |

Table 4.6: Comparison between different RL frameworks for bipedal locomotion with the robot Cassie.

### 4.5.4 Feedback regulations

The gains of the compensations described in Section 4.4.1 and Section 4.4.2 for Cassie and Digit are detailed in Table 4.7. We denote that the regulation for the stance foot orientation is applied only to Digit to enhance the speed tracking performance of the controller in hardware experiments. In addition, given the kinematic tree for Cassie and Digit, the IK functions used in the swing foot orientation regulation are defined in Table 4.7 as $\xi_{tr}$ and $\xi_{tp}$, where $\lambda_r$ and $\lambda_p$ are offsets that depend on the geometric design of the swing leg, and $\gamma, \sigma$ are the inclination of the terrain with respect to the robot's floating base.

## 4.6 Simulation and Experimental Results

Once the trained policy has been exhaustively tested in simulation, we deploy the learned controller on the hardware and evaluate its performance under challenging conditions and terrains. This section shows the performance of the proposed controller structure when evaluated in terms of speed tracking, stability of the walking gait, and robustness against external disturbances and challenging terrains. A sequence of the learning process of the policy and the sim-to-real transfer can be seen in this video: https://www.youtube.com/watch?v=ocAZtHr07Fw.

| Gain | Cassie | Digit |
|---|---|---|
| $K_{phr}^{sw}$ | 1 | 0.5 |
| $K_{dhr}^{sw}$ | 0.05 | 0.03 |
| $K_{php}^{sw}$ | 0.6 | 0.5 |
| $K_{dhp}^{sw}$ | 0.01 | 0.1 |
| $K_{ptp}^{st}$ | - | 0.1 |
| $K_{ptr}^{st}$ | - | 0.02 |
| $K_{p\psi}$ | 100 | 3500 |
| $K_{d\psi}$ | 20 | 500 |
| $K_{p\theta}$ | 100 | 2000 |
| $K_{d\theta}$ | 20 | 500 |
| $\xi_{tr}$ | $q_\psi + q_{hr}^{sw} + \lambda_r + \gamma$ | $q_\psi + q_{hr}^{sw} + \lambda_r + \gamma$ |
| $\xi_{tp}$ | $q_\theta + q_{hp}^{sw} + \lambda_p + \sigma$ | $q_\theta + q_{hp}^{sw} + \lambda_p + \sigma$ |

Table 4.7: Gains and IK functions used in the feedback regulator policy for Cassie and Digit.

### 4.6.1 Simulation results on Cassie

#### Speed Tracking

For evaluation of speed tracking, we assigned a desired velocity profile with fast changes in both longitudinal ($v_x$) and lateral ($v_y$) directions with respect to the robot's body frame. The results presented in Figure 4.6 show that the controller keeps good tracking of the desired velocities in both directions, and it can effectively handle the changes in the speed profile even for large speed changes without significant overshoot. We denote that depending on the combination of the velocity profiles in both directions, the robot can perform different behaviors such as walking in place, walking to the right, left, forward, backward, and walking in a diagonal direction.

#### Stability and feasibility of the walking gait

To evaluate the stability of the generated walking gait, we analyzed the periodicity described by joint limit cycles. Figure 4.7 shows that the phase portrait for

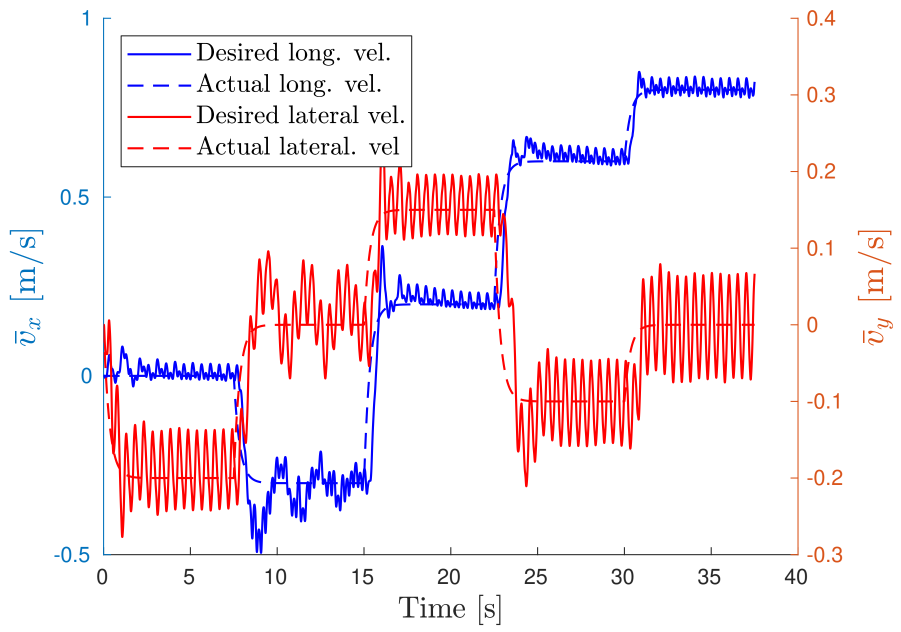

Figure 4.6: Speed tracking performance of the proposed controller in simulation with the robot Cassie. The controller tracks the desired speed for different walking directions: walking forward ($v_x > 0$), backward ($v_x < 0$), to the right ($v_y > 0$), to the left ($v_y < 0$), diagonal (any combination of the previous cases).

the actuated joints while the robot is walking at a constant desired velocity. The plot shows the convergence of the orbits to a periodic limit cycle, demonstrating the stability of the walking gait. Furthermore, the corresponding orbits for the left and right are approximately symmetrical, which was expected by the conditions enforced in the formulation of the RL framework. The minor discrepancies, mostly noticed in hip roll joints, are due to the swing leg regulator's efforts to maintain the lateral stability of the robot.

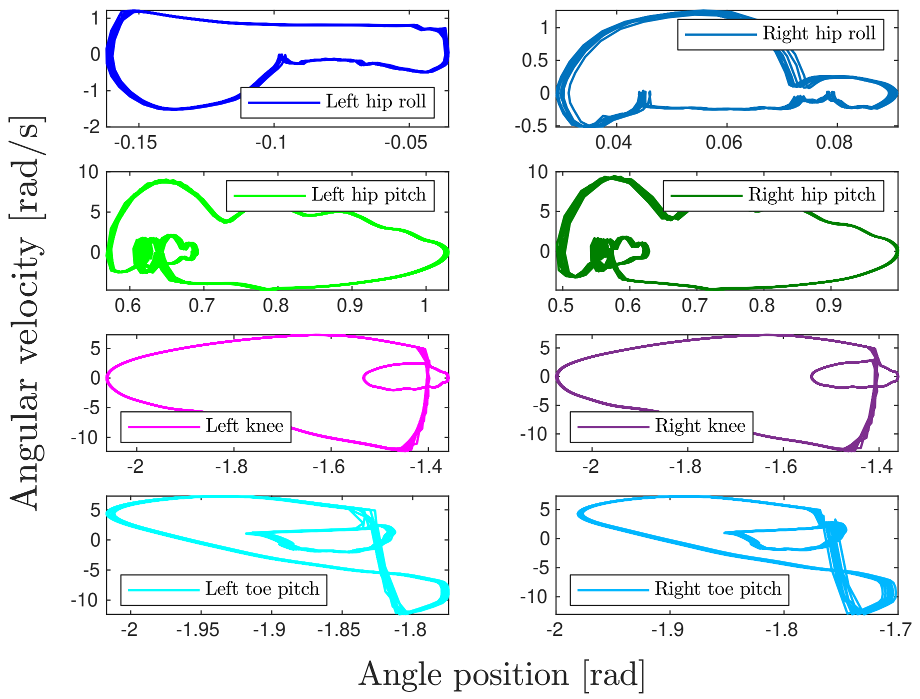

Figure 4.7: Walking limit cycle of the learned policy when tracking a longitudinal velocity $v_x^d = 0.4\ m/s$, and lateral velocity $v_y^d = 0\ m/s$.

### 4.6.2 Experimental results on Digit

#### Speed Tracking

We evaluate the speed tracking performance of the controller in hardware by assigning a velocity profile with variations in the desired velocities in both directions. The results presented in Figure 6.5.a show that the controller keeps good tracking of the desired velocities, especially for the velocity in the longitudinal direction ($\bar{v}_x$). We observe that the tracking error is higher for the lateral velocity ($\bar{v}_y$), which could be caused by the continuous motion of the robot from left to right and vice-versa, asymmetries in the hardware joints associated with the lateral movement, and drifting in the IMU measurements used to estimate the linear velocity. In addition, Figure 6.5.b

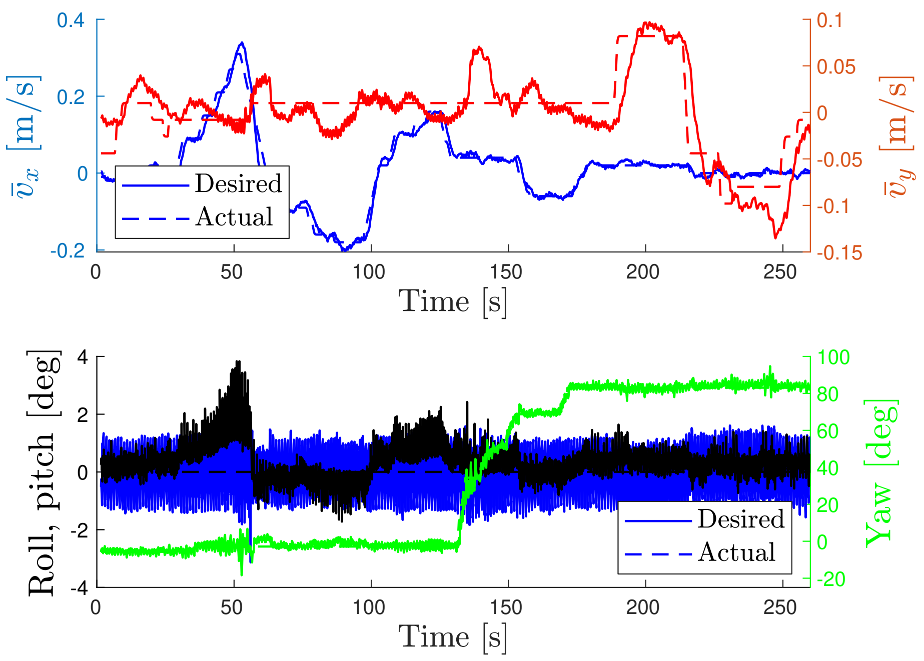

Figure 4.8: Speed tracking performance of the proposed controller. The controller tracks the desired speed for different cases: walking in place ($v_x = 0, v_y = 0$ ), forward ($v_x > 0$), backward ($v_x < 0$), to the right ($v_y > 0$), to the left ($v_y < 0$), and diagonal (any combination of the previous cases).

shows that the controller keeps the torso upright during the walking gait and accurately tracks the desired heading angle. This tracking performance enables the application of the proposed RL-based cascade motion policy for navigation indoors and outdoors.

## 4.7 Conclusion

This work presents a novel RL framework for the design of a cascade motion policy that simultaneously addresses two important problems in bipedal locomotion: trajectory planning and feedback regulation. By incorporating the physical insights of dynamic walking such as symmetry motion, invariance through impact condition, and heuristic regulations into the learning process, we provide a complete and effective solution for the design of feedback controllers that realize stable and robust walking gaits without any prior knowledge of reference trajectories. The method relies on a small-size network with reduced state and action spaces, resulting in improved sample efficiency and reduced training time. The proposed method is tested in simulation with two bipedal robots Cassie and Digit, and successful sim-to-real transfer of the learned policy is demonstrated on Digit with minimal tuning. Extensive hardware experiments show the learned policy can track desired walking speeds in any direction while maintaining stable walking gaits. Moreover, the policy is robust to external disturbances and challenging terrains, including rubber ground, pavement, grass, and slopes.

# Chapter 5: Template Model Inspired Task Space Learning

## 5.1 Introduction and Motivation

In Chapter 5, we presented an RL framework for bipedal locomotion based on a cascade structure to compensate the learned trajectories with feedback regulators to increase the robustness of the walking gait [15, 23]. Although the method was successfully tested in hardware, the interpretability of the learned policy was limited by the complex structure imposed over the input-output mapping of the RL policy and the addition of compensation terms on top of the learned joint trajectories. On the one hand, the integration of the feedback regulators improves the robustness and sim-to-real transfer of the learned policy. On the other hand, it makes it difficult to identify the actual contribution of the learned policy to the robustness of the walking gait. Moreover, the selection of the observation state of the policy is chosen heuristically based on empirical observations with insight from the HZD framework. This results in a policy limited to naturally exploiting the state and action spaces and a restricted walking speed range, e.g., $v_x \in [-0.5, 0.5]$ m/s on the robot Digit.

In this chapter, we explore a more efficient and clean framework that completely decouples the HL learning policy from the LL controller with better insights into the selection of the state and action spaces, which results in improved sample efficiency and interpretability of the policy.

We demonstrate the proposed framework is general for 2D and 3D bipedal robots and can be applied even in the case of underactuated robots. Moreover, we show that the learned policy achieves enhanced performance and robustness compared with the framework presented in Chapter 4.

Common methods applied in bipedal locomotion rely on solving optimization problems using the robot's full-order or reduced-order model to find feasible trajectories that realize stable walking gaits. In general, using full-order models results in computationally expensive problems that cannot be solved in real-time [17, 131]. To reduce the computation time, reduced-order models are used to capture the dynamics of the full-order system and plan trajectories for the robot's center of mass (CoM) and end-effectors. However, the assumptions made on reduced-order models such as a constant CoM height limit their performance on dynamic locomotion behaviors and their accuracy in predicting the behavior of the real robot under certain conditions. Recently, the angular momentum-based linear inverted pendulum (ALIP) has been presented as an improved alternative to the Linear Inverted Pendulum (LIP) to predict the evolution of the model's state, demonstrating in simulation and hardware experiments that the angular momentum about the contact point can be more accurately predicted than the CoM velocity [132, 133].

With the recent success of deep learning in tackling challenging control problems, machine learning-based approaches have exploited advances in physics simulators and computing power to learn locomotion policies through more structured learning frameworks.

Some end-to-end RL frameworks rely on using optimization to obtain a single feasible reference trajectory [13, 70], or libraries of reference trajectories [113, 134] to guide the learning. However, these approaches require large amounts of data, and the learned policy often lacks interpretability and control over the parameters of the walking gait. This makes it difficult to adjust the policy during the sim-to-real transition. As an alternative, more complex frameworks have been proposed to combine learning algorithms with model-based controllers. The authors in [14] take insights from the Hybrid Zero Dynamics (HZD) to learn joint trajectories for planar robots. In [135], an HZD-based approach is used to learn a policy that satisfies Control Barrier Functions (CBF) defined on the reduced-order dynamics.

In this chapter, we propose a hierarchical RL-based approach to address bipedal locomotion in underactuated and fully actuated robots. At the HL stage, RL is used to train policy that learns task space commands for different walking speeds. At the LL, a model-based nonlinear controller is implemented to track the trajectories generated by the HL planning. Several RL-based approaches have been already proposed to exploit hierarchical structures. In [91, 134], a task space policy is trained to walk at different speeds. However, the method relies on solving a series of optimization problems using the Spring Linear Inverted Pendulum to create a gait library that is used as a reference for the reward and the target end-effector positions. The policy learns residual terms that are added to the task space references [91] or joint space references [134]. Different from these approaches, our method directly learns a set of task space actions that completely characterize the dynamic walking gait without the need for previously computed reference trajectories. Moreover, we use different state and action spaces that significantly simplify the complexity of the learning problem and can be generalized to both unactuated and underactuated robots.

The main contributions of this chapter are as follows: 1) **A simple, efficient, and general** hierarchical learning framework that fully decouples the HL planner from the LL feedback controller. Different from other task space learning approaches, our method **(i)** uses a reduced-order state for the RL, **(ii)** learns to walk from scratch, and **(iii)** computes a set of task space actions that fully characterize dynamic walking gaits. The selection of inputs and outputs is general to bipedal robots of different morphology and degrees of freedom. We show results for actuated and underactuated 2D (Rabbit, Walker2D) and 3D robots (Digit).

2) **Insightful design of the RL state space**. We use the ALIP state and speed-tracking information to design a reduced-order state space for the RL that captures the complex dynamics of bipedal locomotion while simplifying the learning process. We expect this to set the path towards learning-based methods for legged locomotion that are more simple and interpretable.

3) **Enhanced flexibility of the policy** to naturally exploit the nonlinear dynamics of bipedal locomotion. By including the desired step length, torso orientation, and CoM's height in the action space of the RL, the policy is not restricted to particular behaviors. This allows the policy to learn natural behaviors seen in dynamic locomotion without enforcing them during training.

4) **A robust locomotion controller** that accurately tracks a wide range of walking speeds, even under external disturbances and challenging terrains, with inclinations up to 20 degrees for both underactuated and fully-actuated robots.

## 5.2 Preliminaries and Problem Formulation

### 5.2.1 Bipedal locomotion as a hierarchical problem

As introduced in section 4.2.1, the bipedal locomotion problem can be characterized as a hybrid system determined by a collection of phases of continuous dynamics with discrete events between the transitions of the continuous phases. Formally, the hybrid system model for biped locomotion from equation 4.2 can be further defined as

$$\Sigma : \begin{cases} \dot{x} = f(x) + g(x)u + \omega(x,u) & x \in \mathcal{X} \setminus \mathcal{H} \\ x^+ = \Delta(x^-) & x^- \in \mathcal{H}, \end{cases} \tag{5.1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$denotes the robot states, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is a vector of actuator inputs. and $\omega \in \Omega \subseteq \mathbb{R}^w$ a vector of disturbances and uncertainties. The switching surface $\mathcal{H}$ is typically the hyper-surface of points corresponding to the height of the swing leg above the ground being zero, and the reset map $\Delta : \mathcal{H} \to \mathcal{X}$ denotes the post-impact state values $x^+$ immediately after switching as a function of the pre-impact state values $x^-$ right before switching.

The control of the bipedal locomotion system described by equation (5.1) can be formulated as a hierarchical control problem composed by a HL planner and a LL tracking controller. This cascade structure is presented in Fig. 5.1. The high level policy $\pi_y$ generates trajectories to realize walking gaits according to design parameters and HL commands, e.g., average walking speed, robustness, terrain slope, etc. The LL policy $\pi_m$ computes the actuator inputs to track the desired trajectories commanded by the HL planner.

The general structure presented in Fig. 5.1 can characterize most controller formulations used in both model-based and model-free state-of-the-art methods for bipedal

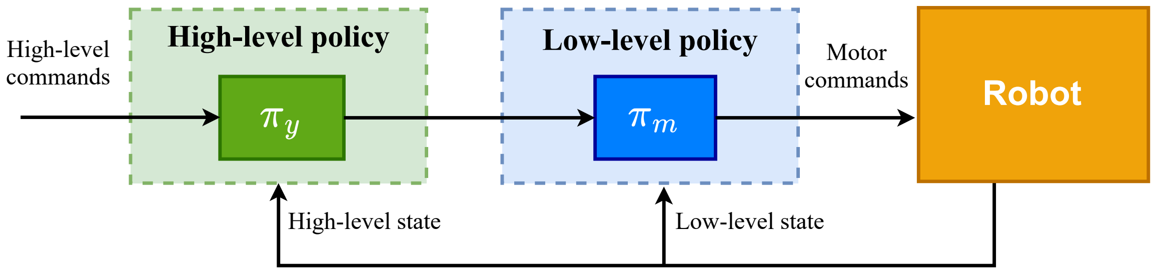

Figure 5.1: Hierarchical structure for bipedal locomotion

locomotion. Once the HL trajectories have been generated by the policy, classic model-based control approaches such as feedback linearization, inverse dynamics QP, operational task space controllers, or simple PD controllers can be used to track the desired HL commands. The choice of the LL control policy $\pi_m$ will mostly depend on the action space of the HL policy, e.g., joint space versus task space.

### 5.2.2 Reduced order models for HL planning

Reduced order models have become a powerful tool for the design of HL planners for bipedal locomotion since they allow using simple dynamical models to characterize biped walking behaviors. Recently, the ALIP model has gained attention because of its advantages over LIP to predict the evolution of the state space. The learning-based approach proposed in this work is heavily inspired by recent results using ALIP as a step planner [132, 133].

**Angular Momentum-based Linear Inverted Pendulum (ALIP):** Considering the states $\{q_x, L^y\}$, where $q_x$ is the CoM position in the $x$ direction, and $L^y$ is the pitch component of the angular momentum about the contact point, the ALIP dynamics is given by

$$\begin{bmatrix} \dot{q}_x \\ \dot{L}^y \end{bmatrix} = \begin{bmatrix} 0 & 1/(mH) \\ mg & 0 \end{bmatrix} \begin{bmatrix} q_x \\ L^y \end{bmatrix}, \tag{5.2}$$

where $m$ is the total mass and $H$ is the constant CoM height. The main advantage of using the ALIP model over the LIP model is that the evolution of the angular momentum about the contact point is closer to its behavior on the full-order robot's model and the actual hardware [132].

**Limitations of reduced-order models**

Although ALIP does better work describing the actual behavior of the system than LIP, both are linear models subject to assumptions such as point mass body, constant CoM height, and the angular momentum about the CoM being zero during the walking gait.

In addition, the prediction of the state at the end of the step depends on the step duration $T$. This implies that an accurate prediction would depend on the perfect timing of the touchdown event, which could only happen in ideal conditions, e.g., perfect tracking of the LL controller, point-contact foot, and non-irregular walking surfaces.

To analyze this effect, we compare the predicted value of $L^y$ scaled by $mH$ at the end of the step with the actual $L^y$ for the five-link bipedal robot Rabbit in Fig. 5.2. We show the evolution of $L^y$ in simulation using the MuJoCo physics engine [130]. To simulate ideal conditions on the model as closely as possible, we set the geometry of the robot's links to be very thin (to emulate point contact with the ground) and use a LL feedback linearization controller with high gains to encourage better tracking and accurate touchdown timing. For the "non-ideal" conditions, we use the real

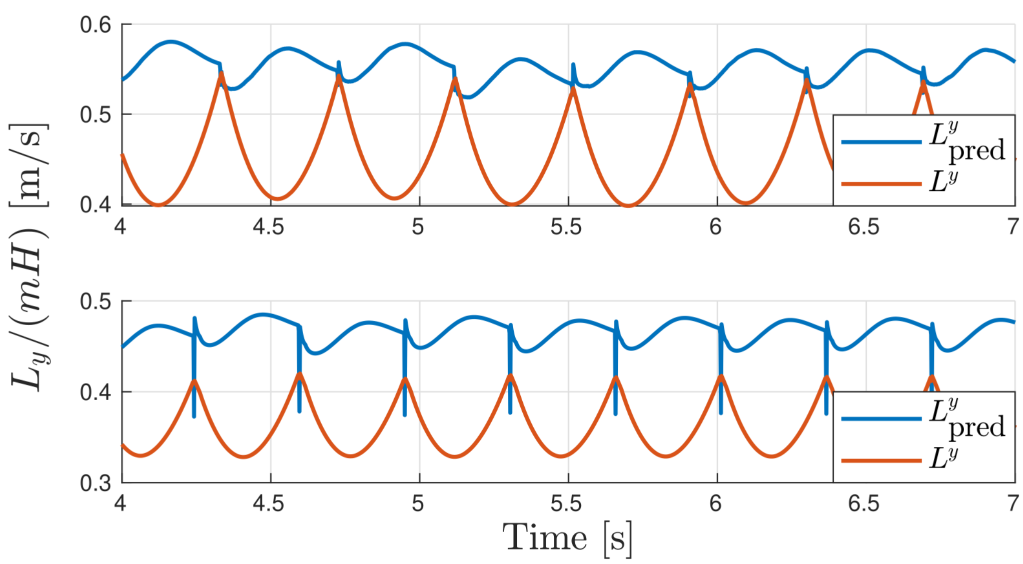

Figure 5.2: Prediction of $L^y$ at the end of each step under ideal (top) and non-ideal (bottom) conditions.

geometry and dynamic properties of the robot's links (as described in [136]), and we use an inverse dynamics QP controller [137]. The results show that under non-ideal conditions, the prediction of $L^y$ at the end of the step differs significantly from its actual value.

### 5.2.3 Task-space LL controller

Several approaches have been proposed in the literature to design task space controllers that consider the full-order model of the legged robot. Considering a mechanical system with configuration space $\mathcal{Q}$ and generalized coordinates $q \in \mathcal{Q}$, the equations of motion formulated using the method of Lagrange are given by:

$$D(q)\ddot{q} + H(q, \dot{q}) = Bu + J^T(q)\lambda \tag{5.3}$$

$$J(q)\ddot{q} + \dot{J}(q, \dot{q})\dot{q} = 0, \tag{5.4}$$

where $D(q)$ is the inertia matrix, $H(q, \dot{q}) = C(q, \dot{q})\dot{q} + G(q) + F$ is the vector sum of the Coriolis, centripetal, gravitational, and additional non-conservative forces, B is the actuation matrix, and $J(q)$ the Jacobian of the holonomic constraints.

We denote that the system (5.3) can be expressed in the general form (5.1). Let $x = \left(q^T, \dot{q}^T\right)^T \in T\mathcal{Q} = \mathcal{X}$, then

$$f(x) = \begin{bmatrix} \dot{q} \\ -D^{-1}(q)\left(J^T(q)\lambda - H(q, \dot{q})\right) \end{bmatrix} \tag{5.5}$$

$$g(x) = \begin{bmatrix} 0 \\ D(q)^{-1}B \end{bmatrix}. \tag{5.6}$$

The task space feedback controller tracks a set of desired trajectories of the form:

$$y(x) = y^a(x) - y^d(\tau(x)), \tag{5.7}$$

where $y^a$ and $y^d$ are smooth functions, and $y^d$ characterizes the desired behavior of the system. Upon the assumption that $y(x)$ has relative degree 1 or 2, nonlinear control methods can be applied to find a control law that drives $y(x)$ to zero, which implies the outputs converge to their target values.

## 5.3 Method

This section presents the methodology for the design of the proposed learning-based hierarchical controller for bipedal locomotion. First, we introduce the overall structure of the framework. Then, we describe the learning-based HL and model-based LL components of the framework.

### 5.3.1 Hierarchical structure for bipedal locomotion

The proposed learning-based framework combines the capabilities of model-based and model-free methods into a hierarchical structure to realize robust locomotion controllers for underactuated and fully-actuated bipedal robots. Inspired by the success

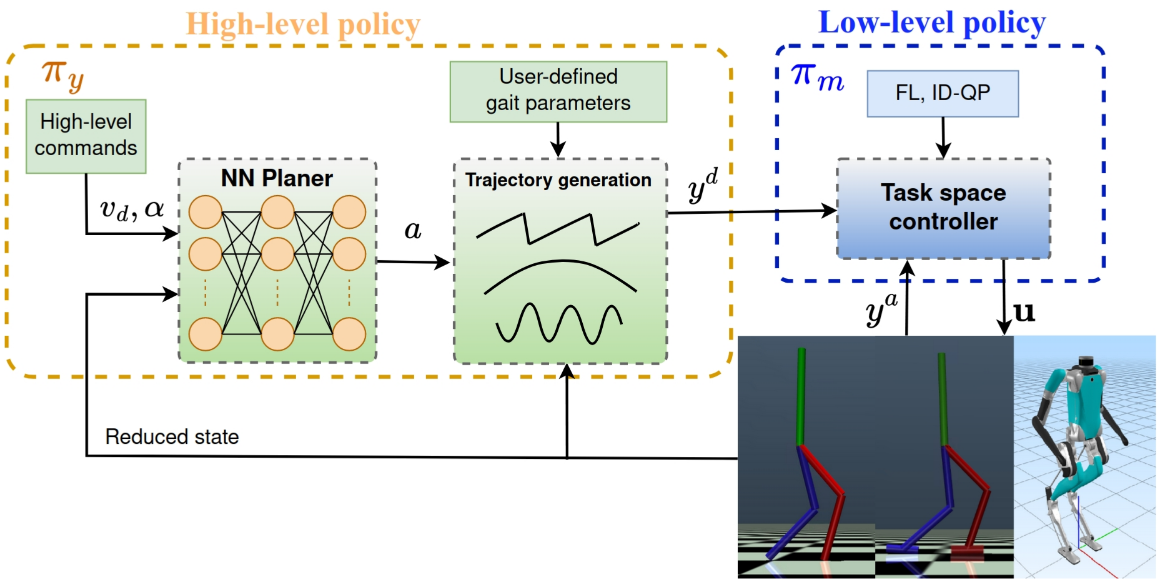

Figure 5.3: Overall structure of the proposed learning-based framework. The HL policy maps a reduced-order state to task space trajectories that are tracked by the LL policy.

of reduced-order models for the online generation of HL trajectories, we use reinforcement learning to train an HL policy that maps a reduced state space inspired by the ALIP model to a set of task space commands to generate online task space trajectories for the robot's base and end-effectors. For the LL task space controller, we use well-known model-based inverse dynamics controllers to guarantee the tracking performance of the system's outputs.

The proposed hierarchical structure is presented in Fig. 7.2. By combining the learning-based HL planner with the model-based LL controller, we obtain a robust controller capable of accurately tracking a wide range of walking speeds while preserving a good tracking performance for the task space trajectories. This significantly increases the flexibility and safety of the policy when compared with pure learning-based controllers.

### 5.3.2 Reinforcement Learning for High-Level Planning

The problem of determining a motion policy for bipedal robots can be modeled as a Markov Decision Process (MDP), which consists of a tuple of components defined as

$$\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathsf{P}, r, \xi, \gamma). \tag{5.8}$$

Here $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the set of feasible actions referred to as the action space. Specifically, at time $t$, an agent (i.e., the motion planner) takes an action $a_t \in \mathcal{A}$ at state $s_t \in \mathcal{S}$, transits into the next state $s_{t+1} \in \mathcal{S}$ according to the transition probability $\mathsf{P}(s_{t+1}|s_t, a_t)$ and receives a reward $r(s_t, a_t, s_{t+1})$. Moreover, $\xi$ denotes the distribution of the initial state $s_0 \in \mathcal{S}$, and $\gamma \in (0, 1)$ denotes the discount factor. The stochastic transition of the MDP process captures the random sampling of initial states in the policy training and dynamics uncertainty due to model mismatch and random interactions with the environment (e.g., early ground impacts).

#### Reduced-Order State Space

Several works have already proposed using a reduced state of the robot as the observation space of the learning algorithm. However, the choice of the reduced state is made based on trial and error or empirical observations of the policy performance. In this work, we leverage recent results on the effectiveness of using angular momentum about the contact point to regulate the walking speed of biped robots [132]. Inspired by the ALIP model, we select the state

$$s = (q_x, q_y, L^x, L^y, e_{\bar{v}^x}, e_{\bar{v}^y}, v_x^d, v_y^d, \alpha). \tag{5.9}$$

where $(q_x, q_y, L^x, L^y)$ is the ALIP state composed by the robot's base position in the $x, y$ axes and the angular momentum about the contact point along the $x$ and $y$ axes, $(e_{\bar{v}_x}, e_{\bar{v}_y})$ is the error between the average velocity the robot's base $(\bar{v_x}, \bar{v_y})$ and the desired robot's velocity $(v_x^d, v_y^d)$, and $\alpha$ is the terrain slope measured in radians. We denote that we use the robot's base position instead of the CoM because of practical convenience for future hardware experiments. The CoM estimation on complex robots may result in noisy measurements, while the base position with respect to the contact point can be easily computed using forward kinematics.

The choice of $(e_{\bar{v}_x}, v_x^d, \alpha)$ as part of the learning state is justified by the overall objective of the HL planner, which is to control the average walking speed of the robot under challenging terrains. We believe speed tracking and robustness are key parameters for the deployment of bipedal robots in the real world. In addition, the output for most global/local motion planning algorithms for navigation is given in terms of longitudinal, lateral and angular speeds.

We assume the slope of the terrain is known by the learning agent. This assumption is reasonable since most of the bipedal robots available for research and commercial applications are equipped with perception systems to map the surrounding environment. Even in the absence of perception systems, proprioceptive approaches could be used to accurately estimate the terrain slope based on the orientation of the robot's base and feet, as we have shown in simulation and hardware in our previous work [18].

**Task Action Space**

The action $a \in \mathcal{A}$ is chosen to be

$$a = (p^x_{\text{sw},T}, p^y_{\text{sw},T}, q^d_\phi, h^d) \tag{5.10}$$

where $p^x_{\text{sw},T}, p^y_{\text{sw},T}$ correspond to the position of the swing foot w.r.t. the robot's base at the end of the swing phase $T$, i.e., the landing position of the swing foot, $q^d_\phi$ is the desired absolute torso pitch angle, and $h^d$ is an offset added to the nominal height of the robot's base w.r.t. the stance foot. This selection of the action space encourages the flexibility of the policy to exploit the natural nonlinear dynamics of the biped robot and enhance the robustness of the policy under big disturbances, sudden speed changes, and walking at high speeds, as it will be shown in section 6.4.

The HL actions $a$ are used to generate smooth task space trajectories for the robot's floating base and end-effectors. The trajectory for $p^{x,y}_{\text{sw}}$ is generated using a minimum jerk trajectory of a straight line segment connecting $p^{x,y}_{\text{sw},0}$ with $p^{x,y}_{\text{sw},T}$. The swing foot position at the beginning of the walking step, $p^{x,y}_{\text{sw},0}$ is computed using Forward Kinematics (FK) and updated at every touch-down event. The desired position for the swing foot at landing, $p^{x,y}_{\text{sw},T}$ is updated by the HL policy at the frequency of $33Hz$. The vertical trajectory of the swing foot position w.r.t. the robot's base is generated using a 5th order Bézier Polynomial parameterized by the vertical position of the foot at the beginning ($p^z_{\text{sw},0}$) and the end ($p^z_{\text{sw},T}$) of the step, and the high foot clearance $p^z_{sw,T/2}$. For flat ground terrain, we have

$$p^z_{\text{sw},T} = -h^d \tag{5.11}$$

We update $p^z_{\text{sw},T}$ by

$$p^z_{\text{sw},T} = -h^d + p^x_{\text{sw},T} * \tan(\alpha) - p^z_{\text{off}}, \tag{5.12}$$

where $\alpha$ is the terrain slope and $p^z_{\text{off}} = 0.005$m is a small offset added to guarantee the swing foot makes contact with the ground.

The Neural Network chosen to parameterize the HL policy is a Recurrent Neural Network with 2 hidden layers, each layer with 128 units for the case of 2D robots and 256 units for 3D robots. The hidden layers use the ReLU activation function, and the output layer is bounded by the sigmoid activation function and a scaling factor to constrain the maximum value of the HL commands.

### 5.3.3 Low-level task space controller

The LL task space controller is designed using standard techniques of the nonlinear systems control literature. In particular, we implement two types of model-based controllers i) Feedback Linearization (FL), and ii) Inverse Dynamics with QP formulation (ID-QP). We evaluate the performance of the HL policy with different LL controllers and show that the learned policy is robust to any choice of the LL controller. The purpose of this evaluation is to demonstrate the versatility of the task space-based HL planner to adapt to different LL control structures without affecting the performance of the learned policy. This also provides more flexibility for the designer to use any LL control approach at their convenience. For instance, FL is easy to implement and requires less computation time, but it is known to be hard to implement on real hardware. Therefore, FL could be used during the training process of the HL policy, while any suitable ID-QP formulation could be used for hardware experiments.

For more details, we refer the reader to [137, 138], where several QP formulations for bipedal locomotion are proposed with successful applications to real hardware. In this work, we use the most basic case of the ID-QP formulation in [137] for the 2D robots and the Task Space Inverse Dynamics (TSID) formulation in [137] for the 3D robot Digit.

### 5.3.4 Learning procedure

The reinforcement learning algorithm we use in this work is an implementation of the Proximal Policy Optimization [10] algorithm with parallel experience collection, input normalization, and fixed covariance. The algorithm shares the same code base as the implementations in [134] and [13].

For each episode, the initial state of the robot is set randomly from a normal distribution about an initial pose corresponding to the robot standing in the double support phase. One iteration of the HL policy corresponds to the interaction of the learning agent with the environment. The HL policy takes the reduced-order state $s \in \mathcal{S}$ and computes an action $a \in \mathcal{A}$ that is converted in desired task space trajectories $y^d$ at the time $t$. The reference trajectories are then sent to the LL task space controller. The LL control loop runs at a frequency of 1 KHz, while the HL planner runs at 33 Hz. The maximum length of each episode is 300 steps, which corresponds to 9 seconds of simulated time.

The episode has an early termination if any of the following conditions are violated:

$$|q_\phi| < 1\text{rad}, \quad q_z < 0.5\text{m}. \tag{5.13}$$

The simple reward function (7.3) adopted in this work is designed to keep track of the target walking speed while realizing a stable walking gait. In particular, the terms $r_{v_x}, r_{v_y}$ encourage the tracking of the longitudinal and lateral target speeds. Since the torso pitch angle is part of the learning action space, the term $r_{L_{\text{CoM}}}$ encourages the policy to avoid excessive changes in the torso orientation without explicitly restricting the torso pitch angle. Finally, the term $r_a$ encourages the policy to avoid excessive variations between the last action and the current action. This avoids unnecessary overshooting in the commanded actions that may produce risky behaviors during the walking gait. The weighted reward function is given as:

$$\mathbf{r} = \mathbf{w}^T [r_{v_x}, r_{v_y}, r_{L_{\text{CoM}}}, r_a]^T, \tag{5.14}$$

where

$$r_{v_x} = \exp\left(-\|\bar{v}_x - v_x^d\|^2\right) \tag{5.15}$$

$$r_{v_y} = \exp\left(-\|\bar{v}_y - v_y^d\|^2\right) \tag{5.16}$$

$$r_{L_{\text{CoM}}} = \exp\left(-\|L_{\text{CoM}}\|^2\right) \tag{5.17}$$

$$r_a = \exp\left(-\|a_k - a_{k-1}\|^2\right). \tag{5.18}$$

and $\mathbf{w}^T$ is a vector of weights corresponding to each reward term. For 2D robots we use $\mathbf{w}^T = [0.6, 0, 0.2, 0.2]$ while for 3D robots we use $[0.3, 0.3, 0.2, 0.2]$.

## 5.4 Illustration example

In this section, we show the proposed method can be generalized to both underactuated and fully actuated robots without any changes to the structure of the HL planner policy. Moreover, we demonstrate the framework can be applied in 2D and 3D bipedal robots. We use 3 different robots.

**Rabbit** is a five-link, planar underactuated bipedal robot with point feet and four actuated joints, two in the hips and two in the knees. Despite its simple mechanical structure, Rabbit still provides a suitable representation of biped locomotion, which is the reason it has been considered as a test bed for advanced control theory in the field of legged robots [136].

**Walker2D** is a seven-link, planar, fully actuated bipedal robot with 6 actuated joints, two in the hips, two in the knees, and two in the ankles. The additional degrees of freedom at the ankles enable the robot to realize human-like walking gaits and balancing.

Schematics of the Walker2D and Digit are shown in Fig. 5.4. Rabbit shares the same design and structure as Walker2D without the feet and ankle joints.

**Digit** is a 3D fully actuated bipedal robot with 30 DoF and 20 actuated joints. Each leg has six actuated joints corresponding to the motors located on the robot's hip, knee, and ankle and two passive joints corresponding to the robot's shin and tarsus joints. In addition, it has four actuated joints per arm corresponding to the shoulder and elbow joints. Fig. 5.4 shows the kinematic structure of Digit.

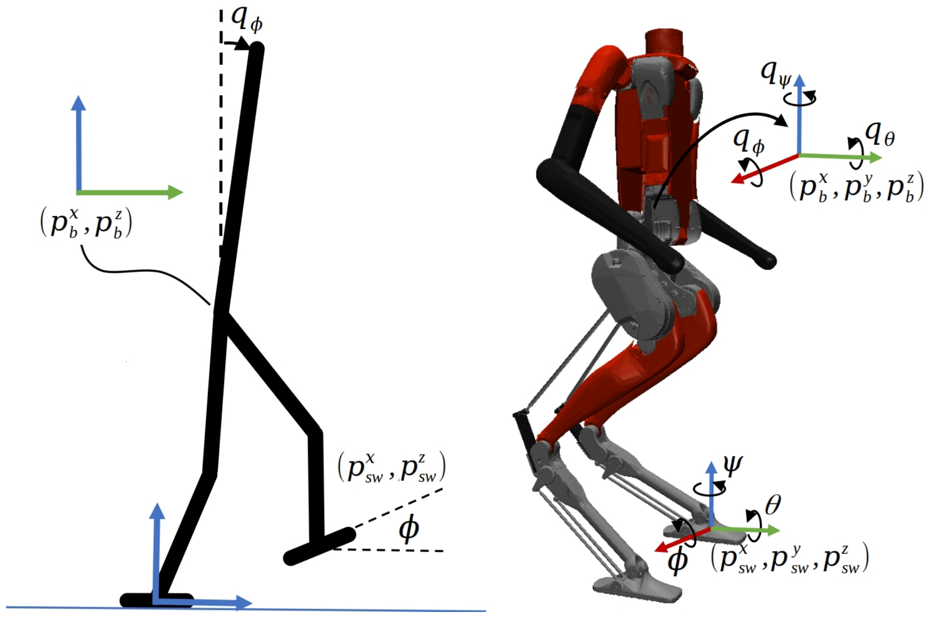

Figure 5.4: Schematics of the robots Walker2D and Digit.

### 5.4.1 Task-space outputs for the LL controller.

The set of task space outputs of relative degree 2 described by equation (5.7) to characterize the walking gait of the biped robot are defined as follows:

$$y_2^a(q) := \begin{bmatrix} q_\phi \\ q_z \\ p_{sw}^x \\ p_{sw}^y \\ p_{sw}^z \\ \phi_{sw} \end{bmatrix} \rightarrow \begin{pmatrix} \text{torso pitch angle} \\ \text{base height} \\ \text{swing foot } x \\ \text{swing foot } y \\ \text{swing foot } z \\ \text{swing foot pitch} \end{pmatrix} \tag{5.19}$$

This selection of outputs is common in the field of bipedal locomotion. The first five outputs $(q_\phi, q_z, p_{sw}^x, p_{sw}^y, p_{sw}^z)$ are valid for both underactuated and fully actuated robots. However, for an underactuated robot, it is not possible to control the horizontal position of the robot's base or CoM. Therefore, the evolution of the base velocity is indirectly controlled by the HL learned policy through the planning of touchdown position. In the case of fully actuated robots, we also consider the sixth output to control the swing foot pitch angle to be parallel to the walking surface. This contributes to reducing disturbances at the touchdown event. Although we could add an additional output to control the horizontal position and velocity of the robot's base, we prefer to rely on the HL planning to control the robot's speed indirectly. The objectives of this choice of design are twofold: i) devise a general framework for both underactuated and fully-actuated robots that share the same structure for the HL policy independently of the particular design of the bipedal robot. ii) simplify the design of the controller and avoid limitations of the torque ankle to control the robot's base position. Although this effect could not be significant for quasi-static locomotion gaits, it does matter when realizing agile and dynamic locomotion.

## 5.5 Simulation results

In this section, we show the performance of the learned HL policy under different testing scenarios with three different robot models, including Rabbit, Walker2D, and Digit. Moreover, we analyze the contribution of the HL policy to the robustness of the walking gait, and we compare our method with similar model-based and model-free approaches.

### 5.5.1 Speed tracking for different velocity profiles.

We test the learned policy for tracking a velocity profile in different directions. Fig. 6.5 shows the velocity tracking performance of the learned HL policy for the robots Rabbit and Walker2D. To evaluate the robustness of the policy under different LL controllers, we test the same policy with the ID-QP controller and the FL controller with different tracking gains. The results show the policy effectively tracks the walking speeds in the range $[-1, 1]$ m/s, even with aggressive changes in the velocity profile.

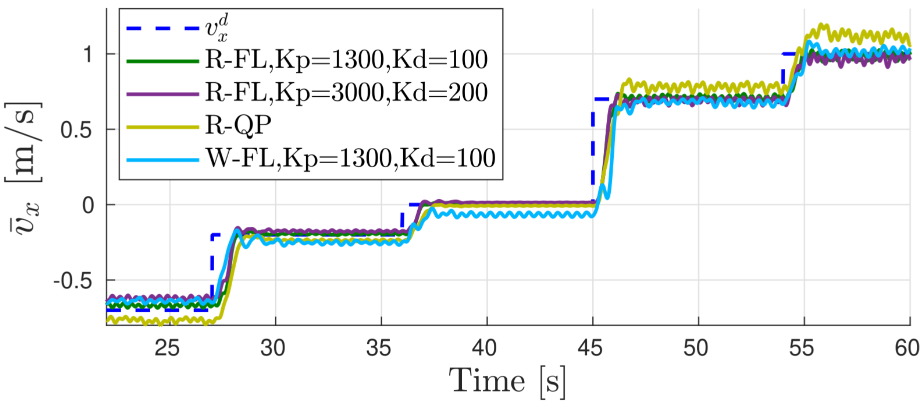

Figure 5.5: Velocity tracking performance of the learned policy with different LL controllers. The prefix R/W is used to differentiate a policy for Rabbit or Walker2D.

To highlight the contribution of the choice of action space, we present in Fig. 6.6 the actions computed by the HL policy for different commanded velocities. When a steep change in the desired velocity is commanded, the policy uses the torso orientation to compensate for variations in the robot's speed and angular momentum. We denote that these behaviors are not enforced during the training process but arise naturally from the insightful design of the proposed framework. Interestingly, some of these strategies are also observed in human locomotion [139].

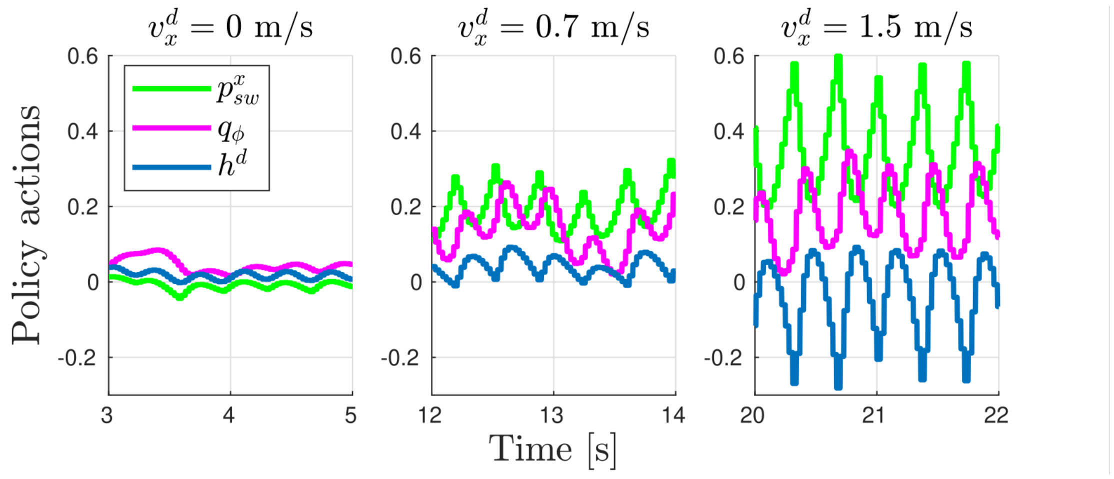

Figure 5.6: Contribution of the policy actions for different speeds.

## 5.5.2 Comparison with ALIP model-based approach

To assess the advantages of the proposed HL planner with respect to pure model-based approaches, we compare the performance between our learning-based controller and the ALIP-based controller. Fig. 5.7 (top) shows the HL-RL policy (ours) outperforms the model-based controller in tracking a velocity profile, especially for high speeds. We also show the variation of $L_{\text{CoM}}$ (bottom) to denote the trade-off the policy learns between minimizing $L_{\text{CoM}}$ and tracking $v_x^d$. For small speeds, $L_{\text{CoM}}$ looks quite similar for both controllers. For high speeds, the policy learns to prioritize speed tracking over minimizing $L_{\text{CoM}}$. This behavior is encouraged by the selection of weights in the reward function (7.3).

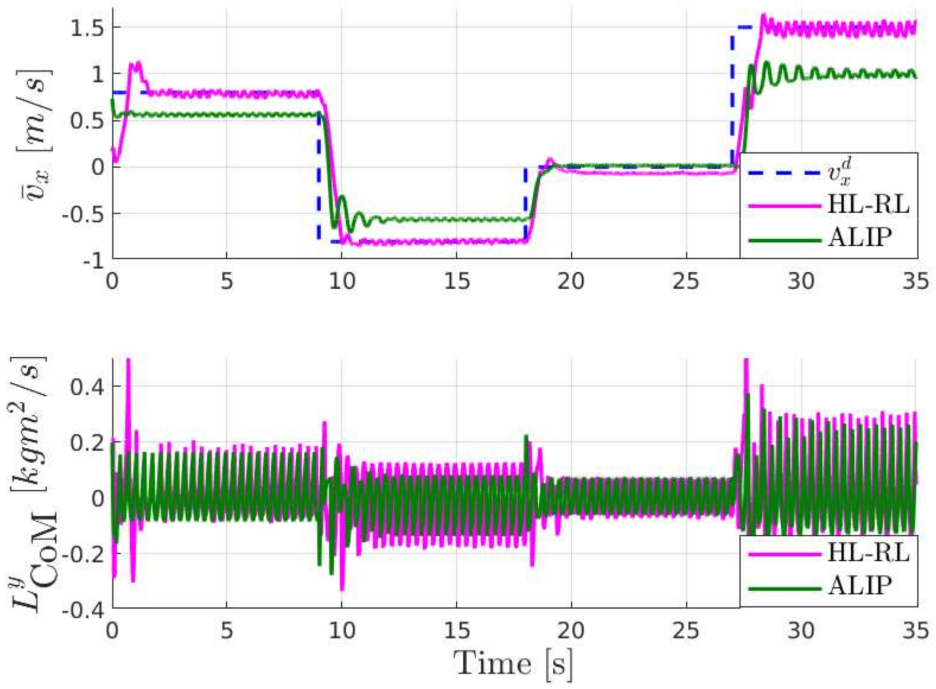

Figure 5.7: Comparison of HL-RL with ALIP-based controller.

### 5.5.3 Comparison with other RL-based approaches

To highlight the contribution of the proposed framework in terms of speed tracking for 3D bipedal robots and sample efficiency, we compare our method with our previous work in [18] and the end-to-end learning approach presented in [13]. Although there are no end-to-end learning approaches implemented on the Digit robot in the literature, there are several works that have done so with the robot Cassie, which shares the same leg morphology as Digit. Therefore, we choose to implement the end-to-end learning approach in [13] with Digit as it focuses on speed tracking, which makes it more comparable to our method. Moreover, the framework in [13] is the base for several SOTA end-to-end learning approaches for bipedal locomotion [83, 91, 140]. Finally, the code implementation for [13] is publicly available online, which makes the comparison as fair as possible in terms of the reproducibility of their work.

In Fig. 5.8, we present the comparison results for speed tracking on flat ground with the robot Digit using our learned HL policy, the HZDRL controller in [18], and the end-to-end RL policy in [13]. We observe that the HZDRL controller fails, i.e., the robot falls, for speeds higher than 0.5 m/s, while the end-to-end RL controller fails to track low speeds accurately and realizes a non-smooth walking motion that causes higher variance in the speed profile. This effect may be caused by the reference trajectory used to guide the learning. In our implementation of [13], we use a reference trajectory corresponding to Digit walking forward at a speed of 0.8 m/s. For more details, the reader can refer to [13]. Our learned policy can successfully track the desired walking speed in a significantly wider range compared with the other methods. Additional testing with Digit demonstrates our controller can handle desired walking speeds in the range $v_x \in [-1.0, 1.5]$m/s and $v_y \in [-0.5, 0.5]$m/s, including combinations of both, i.e., diagonal walking, as can be seen in the following video: https://www.youtube.com/watch?v=YTjMgGka4Ig.

In terms of sample efficiency, Fig. 5.9 shows our approach uses fewer data samples to successfully train a robust policy when compared with the end-to-end learning approach. Although this is not a surprising result as several authors have studied

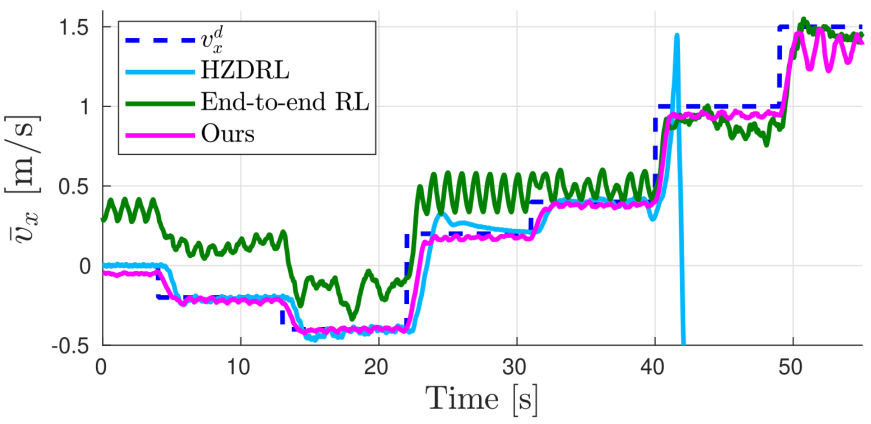

Figure 5.8: Comparison of speed tracking performance with SOTA RL-based approaches.

the effects of task space learning in the data sample efficiency [91, 141], to the best of our knowledge, our method is the only task space approach that learns to walk from scratch and has been successfully implemented in 3D bipedal robots without the need of previously computed reference trajectories, e.g., the gait library used in [91].

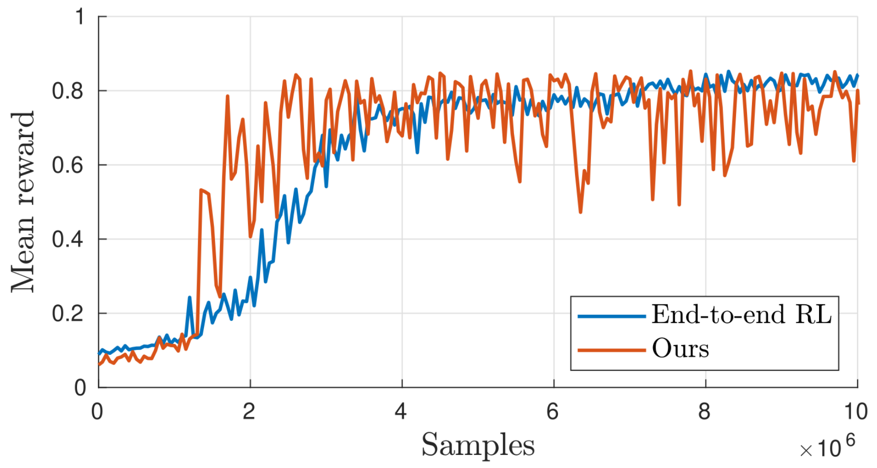

Figure 5.9: Comparison of sample efficiency between traditional end-to-end RL and our proposed method. The higher variance in the reward is caused by the effect of the exploration noise in the task-space actions, which allows the policy to explore a more diverse set of behaviors during training.

### 5.5.4 Robustness on challenging terrain

We test our controller in terrains with slopes up to 20 degrees for Rabbit and Walker2D and 10 degrees for Digit. Fig. 5.10 shows a grid plot with the Root Mean Squared Error between $\bar{v}_x$ and $v_x^d$ for Digit with $v_x^d \in [-1.0, 1.0]$m/s, $v_y^d \in [-0.5, 0.5]$m/s, and $\alpha \in [-10, 10]$ degrees. We denote that during training, we only use $\alpha \in [0, 10]$. Yet, we test the policy in an extended range of slopes to highlight the robustness and interpolation capability of the policy to scenarios not seen during training. In this scenario, we define the robustness of the policy by its capability to keep track of $v_x^d, v_y^d$ under challenging conditions. The results show that the learned policy keeps good tracking of $v_x^d$ in almost all conditions, except for combinations where the robot is walking backward at high speed on very steep terrain.

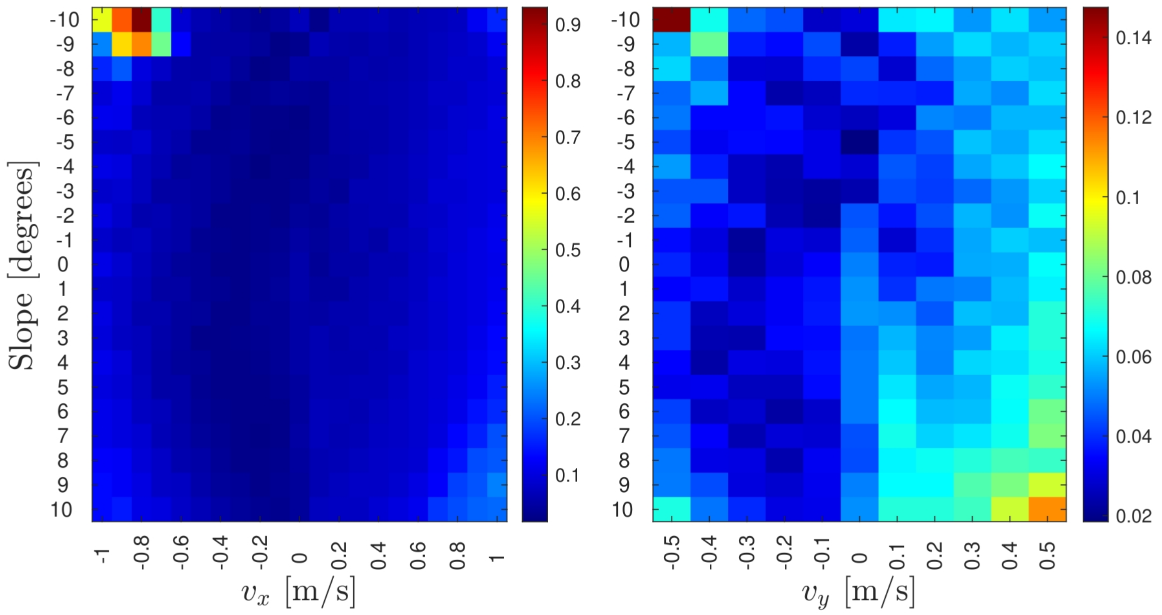

Figure 5.10: Robustness of the learned policy under different conditions of terrain slope and target speed.

A higher error in the tracking of $v_y^d$ is caused by the higher variance of the instantaneous velocity along the $y$ axis, which is expected in bipedal locomotion as the robot's torso tends to oscillate more about the sagittal plane. We also notice there is a higher error when tracking positive speeds along the $y$ axis. This effect may be caused by a biased created during training by the exposure of the policy to more samples with negative commands of $v_y$.

### 5.5.5 Robustness against disturbances.

The trained policy is also subjected to extensive tests against external disturbances $F_x \in [-80, 60]$N applied in both forward and backward directions with duration $t \in [0.15, 3]$s. Fig. 6.8 shows the policy reacts effectively to all the applied disturbances without falling and maintaining the tracking of the desired walking speeds. In some scenarios, the policy uses interesting combinations of the task space outputs to realize effective strategies to reject the disturbances, e.g., bending forward/backward to absorb the impact. The results are similar for Walker2D and Digit. More details can be seen in the accompanying video: https://www.youtube.com/watch?v=YTjMgGka4Ig.

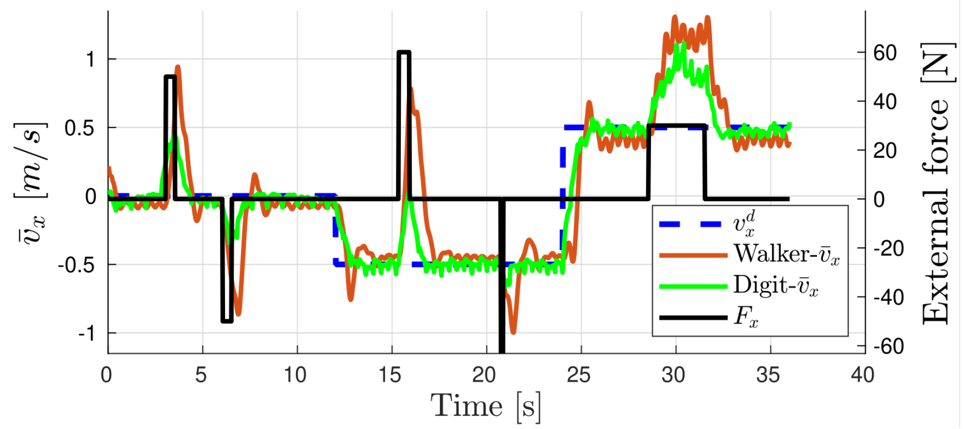

Figure 5.11: Robustness test to external disturbances.

## 5.6 Conclusion

In this chapter, we present a simple and effective learning-based hierarchical approach to realize robust locomotion controllers for bipedal robots. The design of the HL policy is inspired by the reduced-order state of the ALIP model and a set of task space commands that include the step length, torso orientation, and height. This insightful choice of state and action spaces results in a compact policy that learns effective strategies for robust and dynamic locomotion. We show the HL planner is agnostic to the choice of LL controller, and its application is general to underactuated and fully actuated 2D and 3D robots. Finally, we show the learned policy tracks a wide range of speeds even under challenging conditions, such as external forces and slopes up to 20 degrees. Future work will focus on hardware experiments with the robot Digit and the extension of the proposed hierarchical framework to integrate different behaviors such as balancing, climbing stairs, and walking over stepping stones.

# Chapter 6: Data-Driven Latent Space Representation for Robust Bipedal Locomotion Learning

## 6.1 Introduction and Motivation

Conventional methods for bipedal walking often involve solving optimization problems using the robot's full-order [142] or reduced-order model [41, 143] to find feasible trajectories that enable stable walking gaits. While the full-order model captures all the complexities and details of the robot's dynamics, it can be computationally demanding and not suitable for real-time control [17].

On the other hand, reduced-order template models (such as linear inverted pendulum (LIP) and its variants [4, 41, 144]) simplify the dynamics of the system, making it easier to plan trajectories for the robot's center of mass and end-effector. However, as we discussed in Chapter 5, these reduced-order models often require strict constraints to account for the mismatch between the reduced and full-order states of the robot. As an alternative to the use of template models for high-level stepping planning, we proposed in Chapter 5 a hierarchical structure that combines a template-based RL policy with a model-based low-level controller. Even though this framework shows promising results for high-speed and robust walking, the selection of the observation state is still biased by the expertise of the designer and the theoretical and empirical results of reduced-order locomotion models.

An important question arises here. What is the best choice for the selection of the observation state for the RL policy? This question becomes the motivation of the study in this chapter. One critical aspect that we intend to address is the extraction of effective state representations that capture the complex dynamics of bipedal robots, enabling efficient learning and control. With the significant increase in data availability and the potential of machine learning to analyze and discover patterns in unstructured data, data-driven techniques have become powerful tools in the control of complex systems.

These data-driven approaches offer novel ways to address challenges in control design, often sidestepping traditional, more rigid, and computationally expensive methods. There is a growing interest in using RL-based approaches that allow exploiting data from simulation to train controllers in a model-free fashion [13, 70, 113, 134]. However, similar to model-based techniques, they are highly dependent on the quality of the state representation provided to the learning algorithm. As an alternative, more complex frameworks have been proposed to combine learning algorithms with model-based controllers. In [135], an HZD-based approach is used to learn a policy that satisfies Control Barrier Functions (CBF) defined on the reduced-order dynamics. In our previous work [18, 23], a cascade structure is implemented to compensate the learned trajectories with feedback regulators to increase the robustness of the walking gait. A hierarchical structure that combines a template-based RL policy with a model-based low-level controller is proposed in [95].

An effective state representation that can accurately capture the complex dynamics of the whole system can significantly enhance the learning process, enabling more efficient learning and better transferability of control strategies. Unsupervised learning, dimensionality reduction, and representation learning methods can be employed to extract relevant features from high-dimensional sensory data, enabling the development of more efficient and interpretable state representations [145,146]. Dai et al. [60] learn the step-to-step residual dynamics via an adaptive control approach, which is then used to design a foot-stepping controller for a bipedal robot in simulation. More complex learned residual dynamic models are also combined with Model Predictive Control (MPC) for agile systems [147]. These techniques require previous knowledge of the dynamics of the system's model. Several works also have exploited latent representations through reinforcement learning for locomotion. Peng et al. [75] combine techniques from adversarial imitation learning and unsupervised reinforcement learning to develop skill embeddings that produce locomotion behaviors. Starke et al. [52] extract a multi-dimensional phase space from the full-order state motion data, which effectively clusters animations and produces a manifold with better temporal and spatial alignment. However, these are end-to-end frameworks, which makes it difficult to establish a relationship between the latent space and the control actions of the policy. Moreover, these approaches have been mostly used to control animated characters in simulation, and there are no studies focused on implementation for actual bipedal robots.

In this chapter, we propose a novel data-driven framework for bipedal walking that combines a learned low dimensional state representation of bipedal locomotion with a robust gait planner using RL, as depicted in Fig. 6.1. Our framework uses an

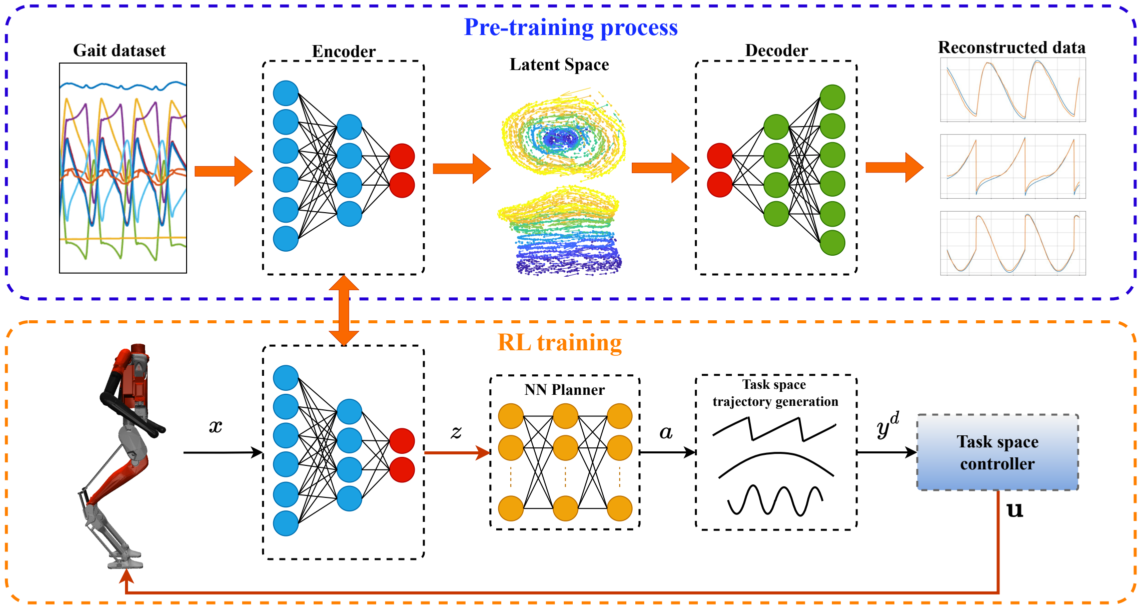

Figure 6.1: An overview of the overall structure and flow of the proposed learning-based framework. In the pre-training phase, the autoencoder learns a latent space that captures the dynamics of the full-order system. During the RL training, the policy maps the latent representation to a set of task space actions that are translated into task space trajectories. Finally, a whole-body task space feedback controller computes the motor torque to track the desired task-space trajectories.

autoencoder to extract an effective reduced-dimensional state representation of the full-order system dynamics. We then integrate this reduced-order latent space with reinforcement learning and a task space feedback controller to train robust locomotion policies.

This work introduces two key contributions. First, we demonstrate that the complex dynamics of bipedal robots can be effectively captured using a low-dimensional learned latent space. This allows for a more compact representation of the system's behavior. Second, we show that the learned latent variables can be leveraged to design a robust locomotion policy using RL. By bridging the gap between state representation learning and learning-based control policies, this work enables leveraging existing locomotion data to develop more effective and adaptable frameworks for versatile and robust bipedal locomotion.

## 6.2 Preliminaries and Problem Formulation

### 6.2.1 Low-Dimensional Models for Locomotion Planning

As we discussed in Chapter 5, the bipedal locomotion problem can be characterized as the hybrid system determined by equation (5.1). For a typical humanoid robot with high degrees of freedom (DoF), the dimension of the robot states $x$ is too large to be effectively used for online feedback for locomotion planning. Low-dimensional models have become a powerful tool for motion planning of bipedal locomotion, given their potential to characterize the dynamics of bipedal walking into simple linear or nonlinear models.

However, existing reduced-order template models, such as the LIP and ALIP, make assumptions that limit the full capabilities of walking robots. These assumptions include a constant CoM height and zero angular momentum about the CoM during the walking gait. While previous research has explored alternative template models for bipedal locomotion, this study aims to investigate whether available locomotion data can be utilized to identify an effective reduced-dimensional state representation of bipedal locomotion for motion planning purposes.

### 6.2.2 Data-driven Low Dimensional Latent Space

Autoencoders are a great tool for harnessing high-dimensional gait data to extract a reduced dimensional latent representation of the system that captures the essence of the full-order robot dynamics. An autoencoder works by compressing the input data into a low-dimensional latent space through a feature-extracting function in a specific parameterized closed form, such as neural networks [146]. This function, called *encoder* is determined by

$$z = h(x, \theta_e), \tag{6.1}$$

where $z$ is the latent variable or representation encoded from the input $x$, and $\theta_e$ is the vector of parameters for the encoder neural network. Another closed-form parameterized function called the *decoder* maps from the latent space back to the full-order state. This function is defined by

$$\hat{x} = d(z, \theta_d), \tag{6.2}$$

where $z$ is the encoded latent variable, $\hat{x}$ is the reconstruction of the original input data $x$, and $\theta_d$ is the vector of parameters for the decoder neural network.

The strength of autoencoders lies in their ability to preserve most of the crucial information from the original data in the latent representation, even though it is of much lower dimensionality. This is ensured by training the autoencoder to minimize the reconstruction loss $\mathcal{L}$, which measures the error between the original data and its reconstruction from the latent space. In summary, autoencoder training consists of finding a value of the parameter vectors $\theta_e, \theta_d$ that minimize the reconstruction error:

$$\mathcal{J}_{\mathrm{AE}}(\theta_e, \theta_d) = \sum_{x^{(i)} \in \mathbf{X}} \mathcal{L}\left(x^{(i)}, d\left(h\left(x^{(i)}, \theta_e\right), \theta_d\right)\right), \tag{6.3}$$

where $x^{(i)}$ is a training sample containing the vectors of position and velocity of the generalized coordinates, and $\mathbf{X}$ is the training set containing all the data samples. $h$ and $d$ are the encoding and decoding functions introduced in (6.1) and (7.8). Once the autoencoder has been trained, the resulting latent representation can then be utilized as part of the state for the high-level planner policy, offering a compact yet expressive state space for learning and control.

## 6.3 Method

The proposed data-driven framework is shown in Fig. 6.1. The gait policy will be learned via RL by utilizing an effective latent representation of the full-order robot's dynamics to a set of task space commands that generate desired trajectories for the robot. Then, a whole body task space controller (TSC) from [95] is employed to accurately track the desired trajectories to realize stable locomotion.

The proposed framework is tested with the robot Digit, which is a 3D fully actuated bipedal robot with 30 DoF and 20 actuated joints built by the company Agility Robotics. Each leg has six actuated joints corresponding to the motors located on the robot's hip, knee, and ankle and three passive joints corresponding to the robot's tarsus, shin spring, and heel spring joints. In addition, it has four actuated joints per arm corresponding to the shoulder and elbow joints. Since the spring joints are very stiff, we considered them as fixed joints in this work. Therefore, the vector of generalized coordinates for Digit is defined by

$$q = (p_b, q_\phi, q_j), \tag{6.4}$$

where $p_b = (q_x, q_y, q_z) \in \mathbb{R}^3 \in$ is the position of the robot's base, $q_\phi \in \mathbb{R}^4$ is the quaternion representation of the robot's orientation, and $q_j = (q_1, \dots, q_{n_j})$ is the vector of the robot's joints with $n_j = 24$. Therefore, $q \in SE(3) \times \mathbb{R}^{24} \subset \mathbb{R}^{31}$, $\dot{q} \in \mathbb{R}^{30}$, and $x \in \mathbb{R}^{61}$.

### 6.3.1 Reduced Dimensional Latent Space Representation

To collect the locomotion dataset to train the autoencoder, we use the hierarchical controller proposed in [95]. The dataset is collected by performing walking gaits at various velocities. Specifically, the velocities $v_x$ and $v_y$ are varied within the ranges of $[-0.5, 1.0]$ m/s and $[-0.2, 0.2]$ m/s, respectively, with a step size of 0.1 m/s, resulting in a total of 16 different walking gaits. Each walking gait has a duration of 10 seconds, with the data being collected at a frequency of 50 Hz. Therefore, the complete locomotion dataset $\mathbf{X}$ consists of $40,000$ samples of the robot's full-order states, i.e., $\mathbf{X} = \{x^{(i)} | i \in [1, 40000]\}$.

***Remark:*** Note that any locomotion controller could be used to collect the gait data. For instance, many commercial robots are equipped with proprietary controllers that could be used to collect this data even though they are a black box from the user's perspective. Moreover, we do not make any assumptions about the distribution of the gait data.

In this work, we use an encoder parameterized by a fully connected neural network with three hidden layers of 128, 64, and 32 units, respectively, and ReLU activation functions. The input of the encoder is the robot's full order states. However, since the inputs of the encoder need to be bounded, the absolute base position of the robot cannot be directly used. This is because the absolute position can grow unbounded as the robot moves in the sagittal or frontal plane. To address this, the base position and orientation of the robot are transformed from the world frame to the stance foot frame, which ensures that the inputs to the encoder remain bounded and allows for effective learning of the latent representation of the robot's dynamics.

The autoencoder is trained using Adam optimizer [148] with a learning rate of 0.001 and a batch size $B$ of 128. The reconstruction loss is computed with the mean squared error (MSE) between the original values of the gait dataset $x^{(i)} \in \mathbf{X}$ and their reconstructed values $\hat{x}^{(i)} \in \hat{\mathbf{X}}$, given as

$$\mathcal{L} = \frac{1}{B} \left( x^{(i)} - \hat{x}^{(i)} \right)^2 . \tag{6.5}$$

The autoencoder is trained for 400 episodes in a 12-core CPU machine with an NVIDIA RTX 2080 GPU. The training takes about 10 minutes using the described locomotion dataset $\mathbf{X}$ and PyTorch.

The selection of the dimension of the latent variable $z$ involves a trade-off. On one hand, a smaller dimension is desirable as it reduces the number of inputs for the RL policy. However, the dimension should also be large enough for the autoencoder to accurately reconstruct the full-order state of the system. In our study, we investigated this trade-off and discovered that even for a complex system like a humanoid robot, a latent variable dimension of $N = 2$ is sufficient to capture the dynamics of the full-order systems effectively. This finding is supported by the results shown in Fig. 6.2, where increasing the dimension of the latent variable does not lead to significant improvements in reconstruction quality. Even in cases where performance is slightly degraded, the reconstructed variable still follows the same pattern as the original state. Interestingly, our findings align with those reported in [52], where the authors demonstrated that periodic movements in bipedal locomotion can be represented using fewer than five phase variables. This similarity may be attributed to the symmetric and periodic nature of bipedal walking.

### 6.3.2 RL-based Gait Policy using Learned Latent Variables

Given the reduced dimensional latent space, we train an RL policy for robust locomotion based on our previous work in [95]. The states of the policy are defined

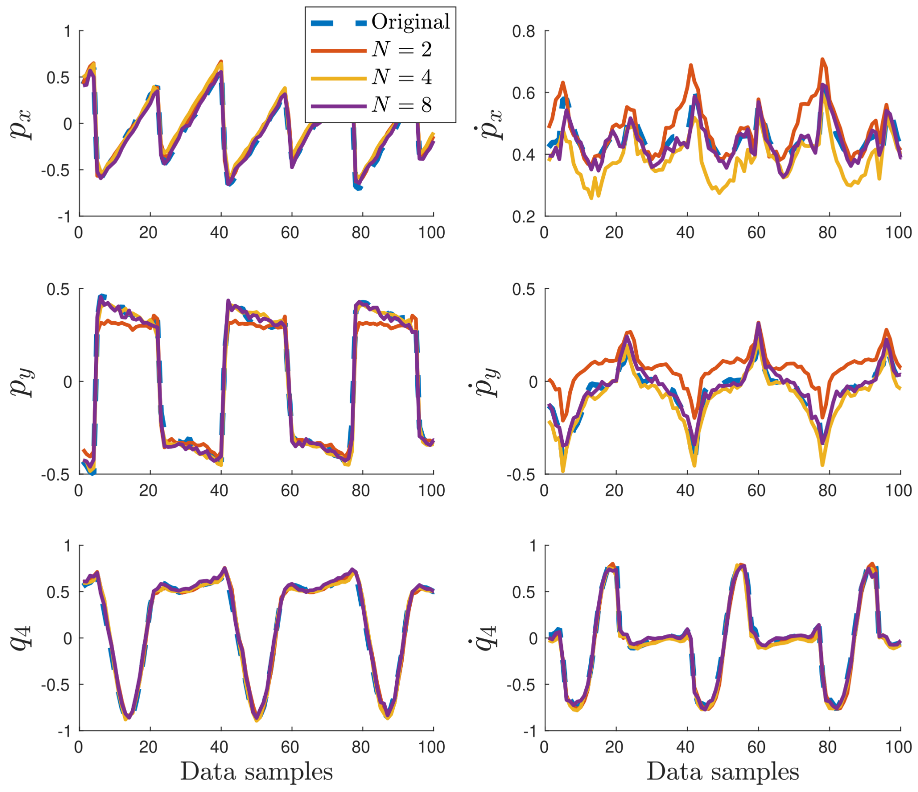

Figure 6.2: Reconstruction of the robot's state with different dimensions $N$ for the latent variable $z$. The plots show the position (left column) and velocity (right column) for the robot's base $x$ coordinate (top), robot's base $y$ coordinate (middle), and left knee joint (bottom), respectively.

as

$$s = (z, e_{\bar{v}}, v^d, a_{k-1}), \tag{6.6}$$

where $z = (z_1, \ldots, z_N) \in \mathbb{R}^N$ is the encoded latent state, $e_{\bar{v}} = (e_{\bar{v}_x}, e_{\bar{v}_y})$ is the error between the average velocity, $\bar{v} = (\bar{v}_x, \bar{v}_y)$, and the desired velocity, $v^d = (v_x^d, v_x^d)$, of the robot, and $a_{k-1}$ is the last action of the planner policy.

The selection of the action space in this work is designed to exploit the natural nonlinear dynamics of the biped robot and enhance the robustness of the policy under various challenging scenarios. Since Digit is a fully actuated system during the single support phase, we include the instantaneous base velocity as part of the action $a$. We choose to control instantaneous velocity over the base position because controlling the position during the stance phase through the TSC is more challenging than velocities in practical hardware implementation. This is caused by the noisy and inaccurate estimation of the position and the limited amount of torque in the robot's ankles. Moreover, sudden motions of the foot position due to terrain irregularities could produce big errors in the base position tracking that could result in aggressive control maneuvers. Therefore, controlling the instantaneous velocity is a less aggressive strategy for the TSC and provides some degree of damping to the ankle motion that helps with the stability of the walking gait under irregular terrains. Thus the action $a \in \mathcal{A}$ of the policy for Digit is chosen to be:

$$a = (p^x_{\text{sw},T}, p^y_{\text{sw},T}, v^d_x, v^d_y), \tag{6.7}$$

which corresponds to the landing position of the swing foot in $x$ and $y$ coordinates with respect to the robot's base and an offset to the instantaneous velocity of the robot's base in $x$ and $y$ coordinates, as illustrated in Fig. 6.3. The trajectory generation module transforms the policy action $a$ into smooth task-space trajectories for the robot's base and end-effectors. Specifically, the trajectory for the relative swing foot positions, $p^x_{\text{sw}}$ and $p^y_{\text{sw}}$, are generated using a minimum jerk trajectory connecting initial foot positions with target foot positions from the policy action. In particular, the initial foot positions will be computed at every touch-down event and kept constant throughout the current step.

The neural network (NN) chosen to parameterize the gait policy is a feed-forward network with two hidden layers, each layer with 128 units. The hidden layers use the ReLU activation function, and the output layer is bounded by the Tanh activation

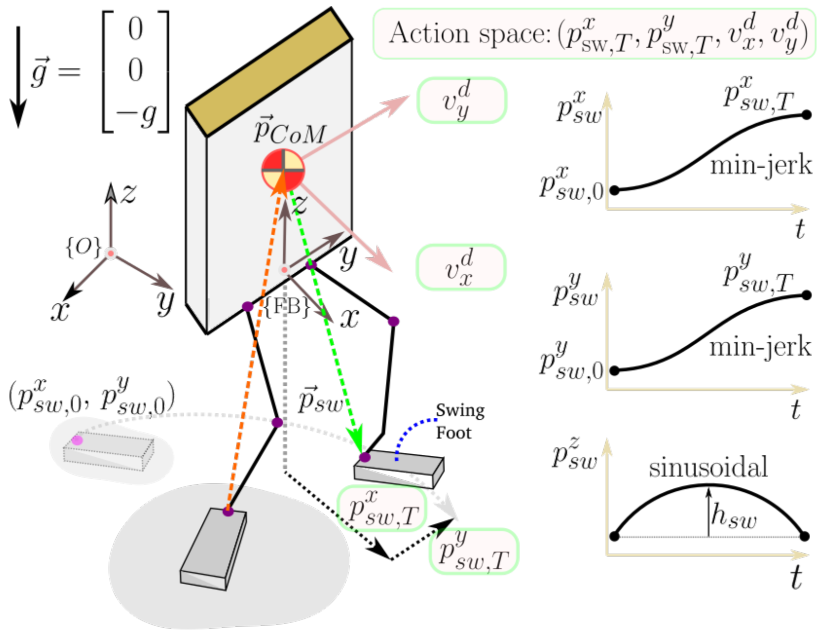

Figure 6.3: Visualization of the policy actions and the trajectories generated for the task space controller. Additionally, we keep the torso straight up and the base at a constant height.

function and a scaling factor to constrain the maximum value of the policy commands within the feasible physical limits of the robot hardware. Finally, we use the TSC model-based controller introduced in Chapter 5 to track the task space trajectories, as described in our previous work [95]. However, we denote that the TSC can be implemented using any control method, including feedback linearization, inverse dynamics, and inverse kinematics.

## 6.3.3 Learning Procedure of the RL Gait Policy

To train the RL policy, we use the Proximal Policy Optimization algorithm [10] with input normalization, fixed covariance, and parallel experience collection. We follow the algorithm implementation described in [13].

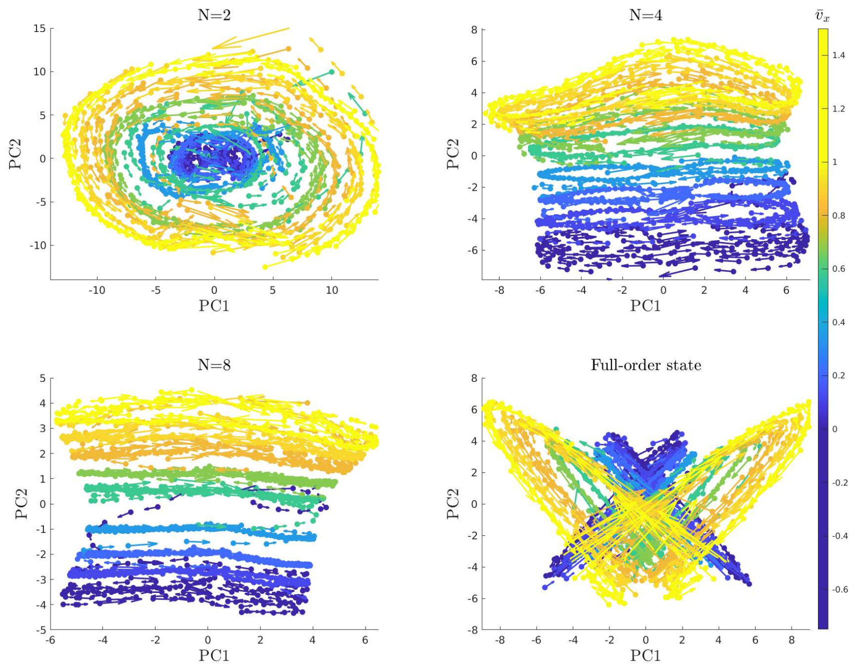

Figure 6.4: Two-dimensional principal component analysis (PCA) of the learned latent manifolds at different walking speeds.

For each episode during the RL training, the initial state of the robot is set randomly from a normal distribution about an initial pose corresponding to the robot standing in the double support phase. An episode will be terminated early if the torso pitch and roll angles exceed 1 rad or the base height falls below 0.8 m. The reward function adopted in this work is designed to (1) keep track of desired velocities, (2) minimize the angular momentum around the center of mass, denoted as $L_{\text{CoM}}$, and (3) reduce the variation of the policy actions between each iteration. More specifically,

$$r = \mathbf{w}^T [r_v, r_{L_{\text{CoM}}}, r_a]^T, \tag{6.8}$$

with

$$r_{v_x} = \exp\left(-\|\bar{v} - v^d\|^2\right), \tag{6.9}$$

$$r_{L_{\text{CoM}}} = \exp\left(-\|L_{\text{CoM}}\|^2\right), \tag{6.10}$$

$$r_a = \exp\left(-\|a_k - a_{k-1}\|^2\right), \tag{6.11}$$

and the weights are chosen as $\mathbf{w}^T = [0.6, 0.3, 0.1]$.

One iteration step of the policy corresponds to the interaction of the learning agent with the environment. The RL policy takes the reduced order state $s$ and computes an action $a$ that is converted in desired task-space trajectories $y^d$ at the time $t_k$. The reference trajectories are then sent to the task-space controller, which sends torque commands to the robot. This workflow is depicted in Fig. 6.1. The feedback control loop runs at a frequency of 1 kHz, while the high-level planner policy runs at 50 Hz. The maximum length of each episode is 600 iteration steps, which corresponds to 12 seconds of simulated time.

## 6.4 Simulation results

In this section, we show the performance of the learned planner policy under different testing scenarios with the bipedal robot Digit. Moreover, we analyze the latent representation generated during the testing of the RL policy. More details about these experiments can be found here: https://www.youtube.com/watch?v=SUIkrigsrao.

### 6.4.1 Latent State Representation

To visualize the learned latent manifolds, we applied principal component analysis (PCA) to reduce the dimensionality of the latent space to 2D. Fig. 6.4 shows the 2D representation of the latent space for dimensions $N = 2$, $N = 4$, and $N = 8$. The data used for visualization correspond to the robot Digit walking with the learned RL policy within the range of $v_x^d \in [-0.75, 1.4]$ m/s. In all three cases, the latent space exhibits a well-distributed and disentangled representation of the data, with

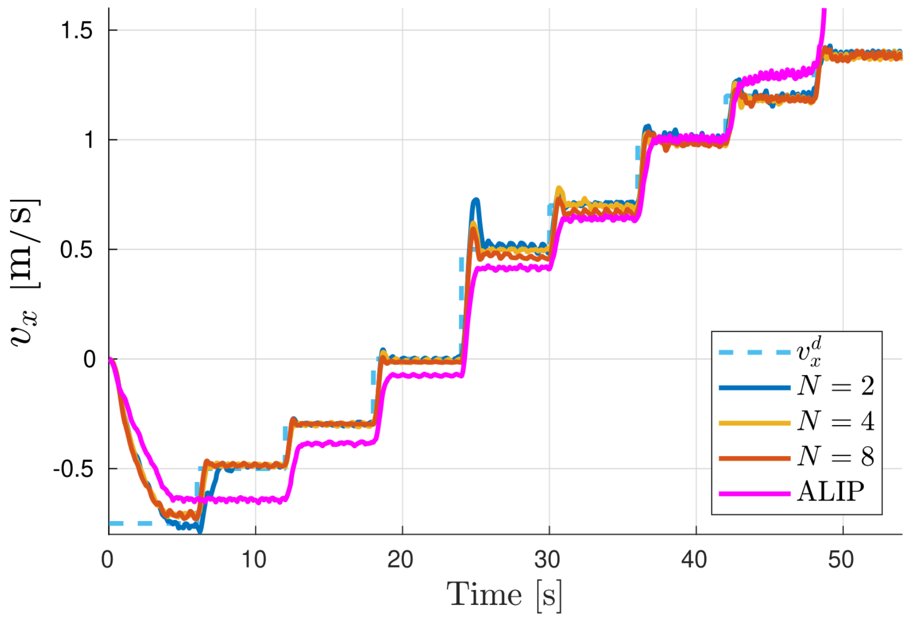

Figure 6.5: Comparison of velocity tracking performance of the learned policy with different dimensions of the latent space $N$ with a model-based ALIP planner described in [4].

each walking speed corresponding to a specific area in the 2D plane. For comparison, Fig. 6.4 also includes the 2D PCA visualization of the full-order state of the robot. It is evident that data points for different speeds overlap with no clear structure. This highlights that the latent space, even with a very low dimensionality $N$, effectively captures the distribution of the data in the full-order system. This demonstrates the potential of exploiting the latent representation for control purposes.

### 6.4.2 Tracking Performance of Different Velocity Profiles

We evaluated the performance of the learned gait policies with latent space dimensions with $N = 2$, $N = 4$, and $N = 8$ in tracking a velocity profile in different directions using the Digit robot. As shown in Fig. 6.5, the policy successfully tracks walking speeds in the range $v_x^d \in [-0.75, 1.4]$ m/s, even with aggressive changes in the velocity profile. Importantly, the data collected to train the autoencoder was within the range of $v_x^d \in [-0.5, 1.0]$. These results show the effectivity of the learned latent states to control the walking velocity even with a low dimension $N$. Furthermore, the results demonstrate that the latent representation captures the dynamic nature of the walking gait and can be generalized to scenarios outside the training distribution. Additional tests about this aspect are presented in Sec. 6.4.3.

For comparison, Fig. 6.5 also shows the speed-tracking performance of a model-based ALIP planner based on the design in [4]. Our policy outperforms the ALIP-based planner in terms of velocity tracking and offers a wider range of admissible walking speeds. The ALIP planner fails to maintain a stable walking gait for speeds higher than 1.2 m/s. In addition, Fig. 6.6 illustrates the correspondence between the walking speeds and control actions of the RL policies with different $N$s and the ALIP planner. Specifically, we focus on the swing foot landing position with respect to the robot's base $p_{x,T}^{sw}$ when the robot is walking at 0.7 and 1.0 m/s. Interestingly, the actions of the RL policy exhibit similar patterns across different latent space dimensions $N$. This finding aligns with the results presented in Sec. 6.4.1, where we showed that even a small $N$ is sufficient to characterize the dynamics of the full-order system fully. When provided with a larger latent space dimension, the policy learns to disregard less important states in the latent space, resulting in similar policy actions for different $N$ values.

Furthermore, the actions of the latent space-based policies exhibit similar patterns to those of the ALIP planner. This is intriguing because the latent space used by the RL policy does not have a direct physical interpretation. Nevertheless, the RL policy

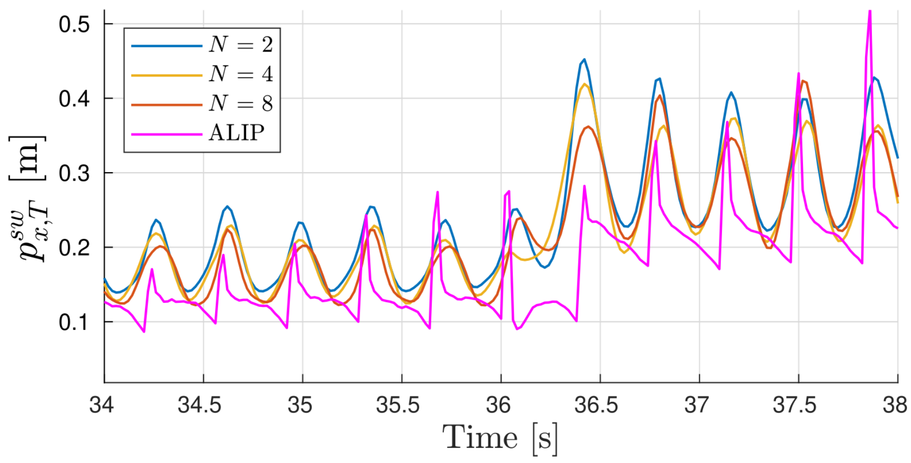

Figure 6.6: Comparison of the desired swing foot landing positions (the policy action) from different latent space dimensions $N$ and the ALIP planner.

learns to behave in a manner similar to the template model-based planner. These behaviors are not enforced during training, suggesting that the latent space naturally captures the dynamics of both the template model and the full-order system.

### 6.4.3 Policy Generalization to Out-of-Distribution Scenarios

In addition to evaluating the policy's performance on data it was trained on, we also assess its ability to handle out-of-distribution scenarios. To do this, we conduct tests where the policy is instructed to maintain a constant speed while the height of the base is varied. It is important to note that the training of the autoencoder and the RL policy did not include any data with varying base heights, as the locomotion data used for training latent spaces was collected with a fixed base height of 1 m. However, as depicted in Fig. 6.7, the policy demonstrates successful tracking of the desired walking speed regardless of the different base heights commanded. An interesting future work direction may be the exploration of the robust generalization of the latent space

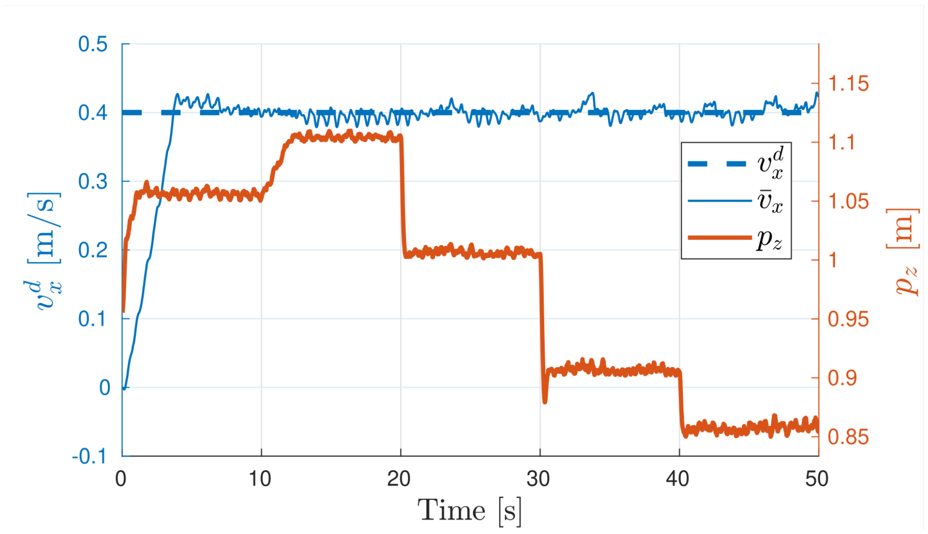

Figure 6.7: Generalization of the latent space and trained policy to out-of-distribution data.

to generate transitions between different locomotion tasks such as walking, jogging, sitting, and jumping.

Furthermore, we conducted tests to evaluate the policy's robustness against external disturbances in the forward and backward directions. The disturbances ranged from $-100$ N to 60 N, with durations ranging from 0.1 s to 1.5 s. As illustrated in Fig. 6.8, the policy demonstrated effective reactions to the different disturbances, successfully maintaining stability and tracking the desired walking speeds without falling.

## 6.5 Conclusion

In this work, we present a novel data-driven learning framework to realize robust bipedal locomotion. The design of the high-level RL gait policy takes data-driven reduced dimensional latent variables as input states and generates a set of task space

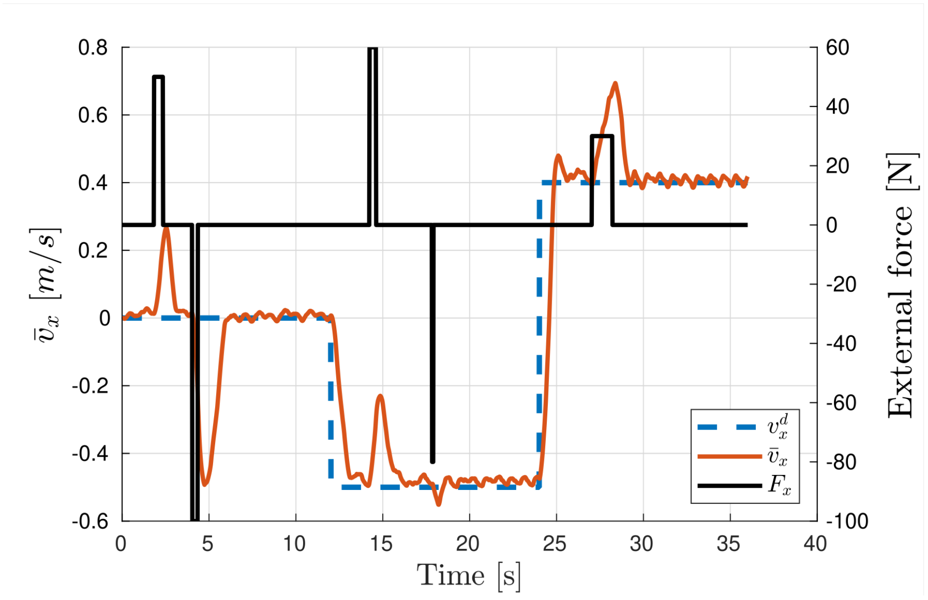

Figure 6.8: Robustness test to external disturbances.

commands, including the robot's step length with respect to the base and instantaneous velocity offset of the robot's base. The latent representation of the full-order state is obtained using an autoencoder trained with supervised learning from locomotion data collected with existing locomotion controllers. Our work shows that the learned latent representation manifold has a disentangled structure that is directly correlated with the speed of the walking robot. The insightful choice of the RL state and action spaces results in a compact policy that learns effective strategies for robust and dynamic locomotion in simulation. Future work will focus on implementing and validating the proposed framework on Digit, further demonstrating its effectiveness and adaptability for real-world bipedal locomotion.

# Chapter 7: Terrain-aware locomotion with hierarchical task space learning

## 7.1 Introduction and motivation

One of the main advantages of legged robots over their wheeled counterparts is their potential to navigate challenging terrains and unstructured environments. In the early stage of legged locomotion research, the focus was on ***blind locomotion*** where legged robots were designed to move without any real-time exteroceptive feedback from their environment. These systems relied heavily on pre-programmed movements and robust locomotion controllers to navigate their surroundings. However, as anyone who has observed the effortless grace of animals and humans to traverse rugged terrains can attest, there is an important difference between simply moving and moving with an awareness of one's environment, known as ***perceptive locomotion***.

Conventional methods employed in blind locomotion often involve solving optimization problems to find feasible trajectories that enable stable walking gaits either using the robot's reduced-order model [44,132,149], the full-order model [17,34], or a hierarchical combination of both [150,151]. Although most of the existing RL-based controllers for bipedal locomotion have focused on learning gaits for blind locomotion,

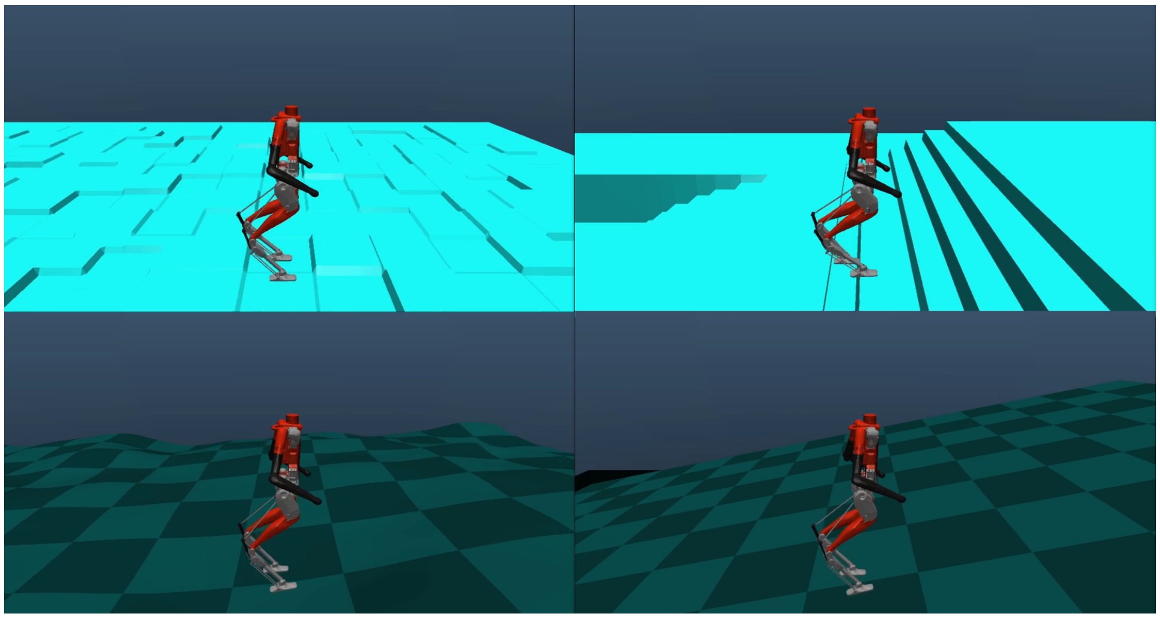
Figure 7.1: Digit walking on different terrains with a single policy.

the impressive robustness of the learned walking policies allows the robot to walk in challenging terrains such as slopes [18], hills [70], and even flights of stairs [152].

The evolution from blind locomotion to perceptive locomotion marked a significant paradigm shift in the field. Exteroceptive information plays a crucial role in influencing robot locomotion. By integrating visual sensors and processing capabilities, robots could now "see" and respond to their environment in real time. This is important for legged robots to adapt to and traverse various environments, such as rough terrains, uneven surfaces, and obstacles. Exteroceptive information not only expanded the range of terrains they could navigate but also improved their efficiency and safety by allowing robots to adjust their gait, speed, and balance in response to the ground beneath them.

In recent years, there has been a growing interest in using deep reinforcement learning to achieve terrain-aware locomotion. By exposing the robot to various terrains and scenarios, these systems can learn optimal movement strategies through extensive interaction with the environment and careful reward design, often resulting in highly adaptive and robust locomotion policies. Similar to the blind locomotion case, reference trajectories could be used to enhance the performance of the policy and the sample efficiency either by imitation learning [153] or as base trajectories that are compensated with the trained policy [154]. Gangapurwala *et al.* present a unified model-based and data-driven approach for quadrupedal planning and control over uneven terrain, utilizing proprioceptive and exteroceptive feedback [155]. Takahiro *et al.* present a robust controller for quadrupedal locomotion in challenging natural environments, trained by reinforcement learning and based on proprioceptive and exteroceptive feedback [156]. Kareer *et al.* propose a framework for visual navigation and locomotion over obstacles, which enables a quadrupedal robot to navigate unseen indoor environments while stepping over small obstacles [157]. Other works have proposed end-to-end RL frameworks with a teacher-student architecture that uses scandots from the terrain as privileged information during the training of the teacher policy with an additional training phase to replace the scandots with an ego-centric depth camera input. This results in agile policies to navigate in challenging terrains [158], stepping stones and balance beans [159], and even in extreme parkour settings with custom-designed rewards [160, 161].

Despite the efforts to integrate legged robots with terrain information through learning-based frameworks, most of the research in the field has focused on quadruped robots. Some of the first attempts to integrate terrain information in an RL policy for bipedal locomotion were implemented in simulations of physic-based animations. In [162], a reduced character state and reduced terrain state were used to train a policy to navigate terrains with steps and gaps in a simulation of a 2D environment. This approach was extended [163] by using the full height field map and character state in an end-to-end RL framework with a mixture of actor-critic experts updated through temporal difference learning. These approaches were extended to the 3D case in simulation in [164], where Hierarchical Deep Reinforcement Learning was used to train a high-level policy that makes step target decisions based on high-dimensional inputs, including terrain maps or other suitable representations of the surroundings and a low-level policy that learn to achieve robust walking gaits. The work in [88,114] addresses the irregular terrain challenge by using pre-planned footsteps obtained from the environment height field. The RL policy then uses the foothold sequence along with the robot's state to compute the joint target positions. While these results were important milestones in terrain-aware locomotion, they were constrained to physics-based simulations with no applications to real-world hardware experiments.

Recently, two important contributions to learning-based perceptive bipedal locomotion have been implemented with Agility Robotics' Cassie, showcasing robust locomotion over challenging terrains. Marum et al. [165] build upon the work in attention-based belief encoding for quadruped locomotion [156] to train an end-to-end policy to navigate a wide variety of terrains using noisy exteroception. Duan et al. [90] use a heightmap expressed in the robot's local frame to train an end-to-end locomotion policy that can navigate in environments with stairs and steps of different heights. The height map is then replaced by a height map predictor obtained from depth camera images. The work in [90] is the first successful implementation of sim-to-real learning for vision-based bipedal locomotion over challenging terrains.

In this chapter, we propose a perceptive bipedal locomotion framework that combines the versatility of learning-based policies for high-level (HL) commands with the robustness of a low-level model-based task space controller to track the desired task-space trajectories. To perceive the terrain around the robot, we use privileged information during training by extracting directly from the simulation the egocentric terrain height map, which is encoded into a latent space that captures the features of different terrain types. This latent terrain representation is used along with a reduced-order representation of the robot dynamics as the input of the learned policy. During training, the policy learns to navigate through challenging scenarios while exploiting the latent encoding of the terrain and the reduced order dynamics of the robot. This results in a highly efficient yet robust locomotion policy for bipedal robots. While there have been several efforts to achieve humanoid locomotion on irregular terrains by using pre-planned footsteps [88, 114], this is, to the best of our knowledge, the first RL-based policy to successfully achieve terrain-aware bipedal locomotion on a human-size robot, e.g. Digit.

The remainder of the chapter is organized as follows. Section 7.2 addresses terrain-aware locomotion as a hierarchical problem, introduces the proposed framework components, and lays out the RL formulation of our approach. Section 7.3 explains the supervised learning approach to encode a latent representation of the terrain by means of variational autoencoders, and it presents a detailed analysis of the importance of the dimension of the learned latent space. Section 7.4 shows simulation results of the proposed framework with the robot Digit, with ablation studies and baseline comparisons to determine the effect of our method on the learning efficiency and policy robustness. Finally, Section 7.5 presents conclusions and future directions of our work.

## 7.2 Method

In this section, we outline the methodology employed in designing the learning-based hierarchical controller for perceptive bipedal locomotion, i.e. using proprioceptive and exteroceptive feedback.

### 7.2.1 Hierarchical terrain-aware bipedal locomotion

Building on the success of reduced-order models for online generation of HL trajectories [44,95,132,149], we employ reinforcement learning to train a high-level planner policy that harnesses an effective representation of the full-order robot's dynamics and the terrain information. As shown in Fig. 7.2, the proposed high-level RL policy takes as input a latent space encoding learned from the local height map of the terrain together with a reduced-order representation of the robot's dynamics based on the LIP model and the state of the swing foot of the robot. The output of the RL policy is a set of task space commands used to generate online task space trajectories for the robot's base and end-effectors. The low-level controller is a model-based whole-body controller used to guarantee the tracking performance of the task space desired trajectories.

### 7.2.2 Reinforcement Learning for High-Level Planning

The problem of determining a motion policy for bipedal robots can be modeled as a Markov Decision Process (MDP). The stochastic transition of the MDP process

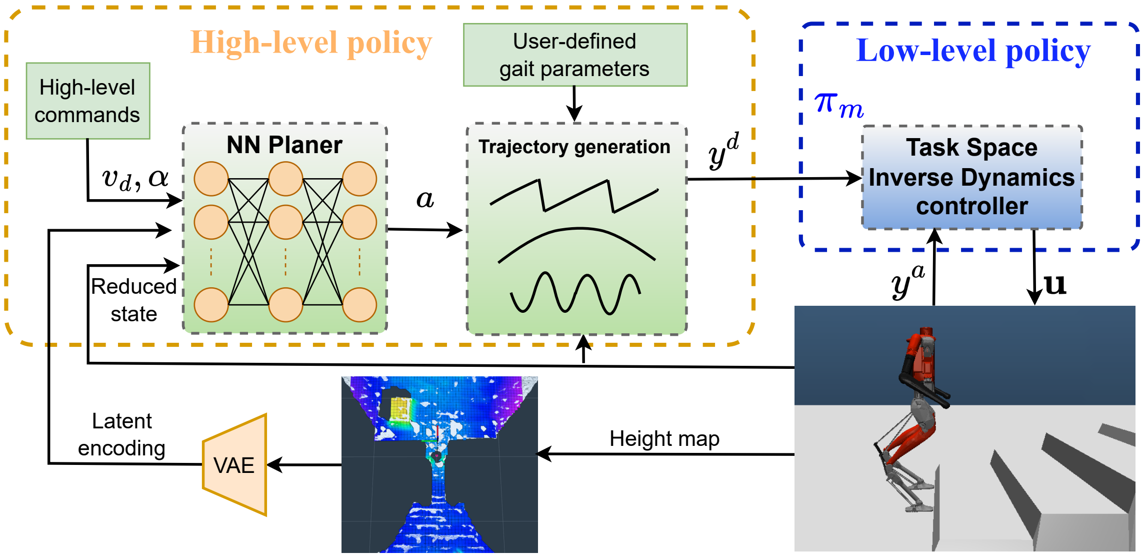

Figure 7.2: The hierarchical structure of the proposed framework: a high-level (HL) policy for gait planning and a low-level (LL) controller for trajectory tracking. A convolutional variational autoencoder (CVAE) is used to encode the local height map to a reduced dimensional latent variable for training terrain-aware perception locomotion policy.

captures the random sampling of initial states in the policy training and dynamics uncertainty due to model mismatch and random interactions with the environment (e.g., early ground impacts).

### Reduced-Order State Space

In this work, we leverage the insights provided by template models to regulate the walking speed of biped robots [132]. Inspired by the success of [18, 95] to use template-based models, we select the state

$$s = (x_{\text{b}}, q_{\text{z}}, e_{\bar{v}}, v_x^d, v_y^d, p_{sw}, v_{sw}, z, a_{k-1}), \tag{7.1}$$

where $x_{\mathrm{b}} = (q_x, q_y, v_x, v_y)$ is the LIP state composed by the robot's base $x$ and $y$ position and velocity, $h_{\mathrm{b}}$ is the robot's base height, $e_{\bar{v}} = (e_{\bar{v}_x}, e_{\bar{v}_y})$ is the error between the average velocity of the robot's base $(\bar{v_x}, \bar{v_y})$ and the commanded robot's velocity $(v_x^d, v_y^d)$, $p_{sw}, v_{sw} \in \mathbb{R}^3$ are the cartesian position and velocity of the robot's swing foot, $z_{hm}$ is the latent variable encoded from the local height map centered at the robot's base, and $a_{k-1}$ is the last policy action. The details about the encoded representation $z_{hm}$ of the terrain map are presented in Section 7.3.

**Task Space Actions**

The action $a \in \mathcal{A}$ is chosen to be

$$a = (p^x_{\mathrm{sw},T}, p^y_{\mathrm{sw},T}, p^z_{\mathrm{sw},T}, h_{\mathrm{sw}}, v^x_{\mathrm{off}}, v^y_{\mathrm{off}}) \tag{7.2}$$

where $p^{x,y,z}_{\mathrm{sw},T}$ correspond to the landing position of the swing foot w.r.t. the robot's base at the end of the swing phase $T$, $h_{\mathrm{sw}}$ is swing foot clearance, and $v^x_{\mathrm{off}}, v^y_{\mathrm{off}}$ are an offset added to the commanded speed of the robot. This selection of the action space encourages the flexibility of the policy to exploit the natural nonlinear dynamics of the biped robot and enhance the robustness of the policy under challenging terrains, disturbances, and sudden speed changes caused by irregularities in the terrain, i.e. tripping over.

**Low-level task space controller**

Given the desired set of policy actions, the trajectory generation module transforms the policy action $a$ into smooth task-space trajectories for the robot's base and end-effectors. Specifically, the trajectory for the relative swing foot positions, $p^x_{\mathrm{sw}}$ and $p^y_{\mathrm{sw}}$, are generated using a minimum jerk trajectory connecting initial foot positions with target foot positions from the policy action. In particular, the initial foot positions will be computed at every touch-down event and kept constant throughout the current step. To track the task space trajectories, we implemented a model-based whole-body controller with Quadratic Programming formulation based on work presented in [95, 150].

### Rewards

The reward function adopted in this work is designed to exploit the privileged information from the terrain's height map to shape the motion of the robot's swing foot while keeping track of the desired walking speed and reducing the variation of the policy actions between each iteration. More specifically, we define the reward function

$$\mathbf{r} = \mathbf{w}^T [r_{v_x}, r_{v_y}, r_{sw_x}, r_{sw_y}, r_{sw_z}, r_{sw_h}, r_{sw_f}, r_a]^T, \tag{7.3}$$

with

$$r_\square = \exp\left(-\|\square - \square^d\|^2\right) \tag{7.4}$$

$$r_a = \exp\left(-\|a_k - a_{k-1}\|^2\right), \tag{7.5}$$

where $\square$ represents the measured value and $\square^d$ is the target value. For the velocity rewards $r_{v_x}, r_{v_y}$, the target value is the desired walking speed sampled at the beginning of each episode. For the swing-foot rewards $r_{sw_x}, r_{sw_y}, r_{sw_z}$ the target values are the 3D foothold position, where the $(x, y)$ target coordinates are ideally estimated from the desired walking speed and the $z$ target coordinate is the terrain height corresponding to the $(x, y)$ coordinates. Finally, the rewards $r_{sw_h}, r_{sw_f}$ correspond to the swing-foot clearance and force, which encourages the policy to adjust the max height of the swing foot during the swing phase according to the terrain and to avoid early contact of the swing foot with the edges of the terrain during the swing phase. The weights are chosen as $\mathbf{w}^T = [0.2, 0.1, 0.075, 0.075, 0.15, 0.2, 0.15, 0.1]$.

### RL training setup

To train the RL policy, we use the Proximal Policy Optimization algorithm [10] with input normalization, fixed covariance, and parallel experience collection. We follow the algorithm implementation described in [13]. The neural network selected for the RL policy is an MLP with two hidden layers, each one with 128 units and *tanh* activation function. For the PPO algorithm, we use a batch size of 64, a discount factor of 0.95, and 56000 samples with 6 epochs of policy update per algorithm iteration. For each training episode, a terrain type is randomly selected from a set of 5 different terrains: hills, slopes, random stairs, squared steps, and stairs up. These terrains are randomly generated from a diverse set of parameters, such as slope degree, number of stairs, stairs dimensions (width, height, depth), and size of squares, among others. Moreover, the initial state of the robot is drawn from a normal distribution about an initial pose corresponding to the robot standing in the double support phase.

One iteration step of the policy corresponds to the interaction of the learning agent with the environment. The RL policy takes the reduced order state $s$ and computes an action $a$ that is converted in desired task-space trajectories $y^d$ at the time $t_k$. The reference trajectories are then sent to the task-space controller, which sends torque commands to the robot. This workflow is depicted in Fig. 7.2. The feedback control loop runs at a frequency of 1 kHz, while the high-level planner policy runs at 33 Hz. The maximum length of each episode is 300 iteration steps, which corresponds to 9 seconds of simulated time. An episode will be terminated early if the torso pitch and

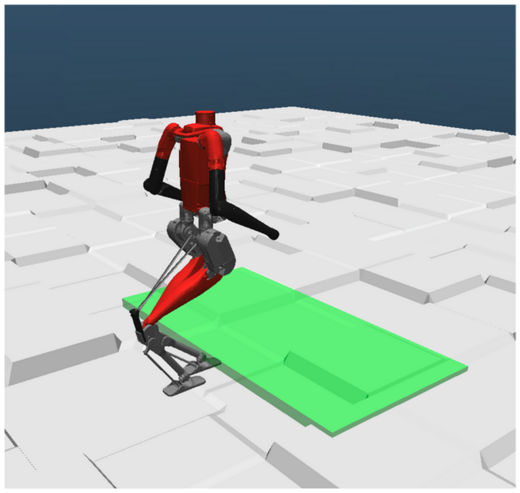
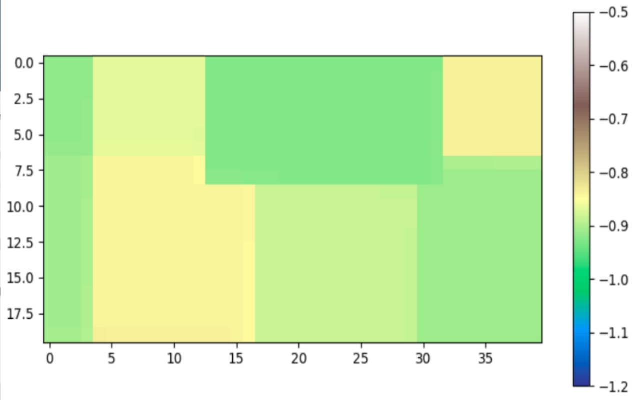

Figure 7.3: Local height map used to detect the terrain around the robot.

roll angles exceed 1 rad or if the height of the robot's base relative to the stance foot is less than 0.4 m.

## 7.3 Terrain representation in latent space

To perceive the terrain around the robot, we use a local height map of the terrain corresponding to the area of 2 square meters in front of the robot's base. When using a resolution of 5 square cm, the local terrain height map is represented by a matrix of size $20 \times 40$, which results in a total of 800 elements. Using all the elements of the local terrain height map matrix as input of the RL policy would result in an increased number of network parameters and, consequently, a larger inference time. Moreover, many of these elements may not have useful information for the policy to produce effective control actions. For illustration, we show in Fig. 7.3 a sample of this local height map obtained from simulation.

Therefore, we use a convolutional variational autoencoder (CVAE) to encode the whole terrain height matrix into a single vector $z_{hm}$ of dimension $m$. This encoding process significantly reduces the neural network (NN) input dimension while harnessing the advantages of VAEs, such as probabilistic modeling, generative capabilities, and the ability to capture meaningful latent representations [145].

## 7.3.1 Terrain data collection

We use a customized simulation environment in Mujoco [130] to create different terrain profiles, including sloped planes, hills, squared steps, and stairs with different configurations (up, down) and dimensions (width, depth, and height). For each type of terrain, we collect 60000 samples of local height maps with respect to the robot's base as described in Section 7.3.In this work, we do not assume the existence of a locomotion controller we can use to collect the data of the height samples. To collect a diverse dataset of terrain height maps, we simulate the kinematic motion of the robot around the terrain by updating the position of the robot's base in the simulation environment according to randomly sampled velocity while keeping the base height at a height that follows a normal distribution with a mean 0.92 m and standard deviation 0.1 m. Thus, the complete dataset $\mathbf{X_{hm}}$ consists of 360000 samples of local terrain height maps $\mathbf{x}$ , i.e., $\mathbf{X_{hm}} = \{x_{hm}^{(i)} | i \in [1, 360000]\}$.

## 7.3.2 Convolutional Variational Autoencoder

A CVAE is a type of neural network model that is particularly adept at handling complex data like images. It combines the principles of convolutional neural networks (CNNs), which are excellent for image processing, with those of variational autoencoders (VAEs), which are a type of generative model that excel in learning the underlying probability distribution of input data. This is different from the traditional autoencoder, where the input is mapped to a deterministic unique value. Therefore, one of the primary advantages of CVAEs is their proficiency in handling high-dimensional data, such as large maps, and effectively compressing them into lower-dimensional representations that capture the essential features and structures of the original data through a conditioned probability distribution.

In this work, the *encoder* part of the CVAE consists of three convolutional layers: the first layer comprises 32 filters with a kernel size of 4, stride 2, and padding 1, effectively reducing the input's spatial dimensions by half while increasing the depth of the feature map. The subsequent layers follow a similar structure but with an increased number of filters (64 and 128, respectively), further compressing and encoding the spatial information present in the height map.

These convolutional layers are followed by two fully connected layers, each mapping the flattened output of the last convolutional layer to the latent space. Specifically, these layers output the mean $\mu$ and variance $\sigma$ of the prior distribution, both sized according to the predefined latent variable dimension $m$. Then, the latent random variable $z_{hm}$ can be expressed as a deterministic variable

$$z_{hm} = g_\theta(\epsilon, x_{hm}), \tag{7.6}$$

where $x_{hm}$ is the sample vector corresponding to the local height map, $g_\theta(\cdot)$ is the encoder function parameterized by $\theta$, and $\epsilon$ is an auxiliary random variable with an independent marginal probability distribution. Therefore, if we choose $\epsilon$ to be the univariate Gaussian distribution $\mathcal{N}(0, 1)$, the random latent variable $z_{hm}$ is determined by

$$z_{hm} = \mu + \sigma\epsilon. \tag{7.7}$$

This method, known as the reparameterization trick, allows backpropagation through random sampling processes, which is essential to train VAEs through standard stochastic gradient descent methods.

The encoder part of the CVAE not only efficiently reduces the dimensionality of the input data but also captures its essential features in a form conducive to generative tasks. By learning this compact latent representation, the CVAE can effectively generate a probability distribution from which one can reconstruct the local height map samples and even generate new, unseen maps that share statistical properties with the training data.

### 7.3.3 Reconstruction of the height map

The decoder is the closed-form parameterized function that maps from the latent space back to the full-order state. This function is defined by

$$\hat{x}_{hm} = d(z_{hm}, \theta_d), \tag{7.8}$$

where $z_{hm}$ is the encoded latent variable, $\hat{x_{hm}}$ is the reconstruction of the original input data $x_{hm}$, and $\theta_d$ is the vector of parameters for the decoder neural network.

The CVAE is trained by minimizing the standard $\beta$-VAE loss [145] $\mathcal{L}$, which consists of reconstruction loss and the Kullback-Leibler (KL) divergence [166] as the latent loss. Then, the VAE loss is formulated as

$$\mathcal{L} = \mathrm{MSE}(x_{hm}^{(i)}, \hat{x}_{hm}^{(i)}) + \beta D_{\mathrm{KL}}(q(z^{(i)}|x_{hm}^{(i)})||p(z_{hm}^{(i)})) \tag{7.9}$$

where $\hat{x}_{hm}^{(i)}$ is the reconstructed height map, $p(z_{hm}^{(i)})$ is the prior distribution parameterized by the Gaussian distribution $\epsilon$, and $q(z_{hm}^{(i)}|x_{hm}^{(i)})$ is the posterior distribution of the latent variable $z_{hm}^{(i)}$ given $x_{hm}^{(i)}$. The autoencoder is trained using Adam optimizer [148] with a learning rate of 0.001 and a batch size $B$ of 256. The autoencoder is trained for 40 epochs in a 12-core CPU machine with an NVIDIA RTX 2080 GPU. The training takes about 20 minutes using the described dataset $\mathbf{X_{hm}}$ and PyTorch.

There is no rule of thumb for the proper size for the encoded state used to capture the features of the encoder input. On one side, a latent variable of large dimension allows a better reconstruction of the local height map. On the other side, a smaller dimension of the latent variable enables more efficient encoding and, consequently, results in more compact networks both for the VAE and the RL policy. To analyze the trade-off between these two properties, we conduct an ablation study with different values of the latent variable dimension $m$. In Fig. 7.4, we show the loss $\mathcal{L}$ during training of the CVAE for different values of $m$. Latent dimensions 256, 128, and 64 show the most rapid decrease in loss, indicating efficient learning and improved ability to capture data features. Latent dimensions 64 and 32 show a good balance between model complexity and learning efficiency. The smaller dimensions 8 and 4 exhibit higher average loss, which means the model is not complex enough to capture all the necessary features of the data.

In addition, we apply principal component analysis (PCA) to the encoded representation of the different latent dimensions. These results, presented in Fig. 7.5, show that for $m \geq 16$, the latent space shows the same structure for the encoded data, while for values of $m \leq 8$, the structure of the data changes significantly. This means that even a latent dimension as low as $m = 16$ can still capture the same features from

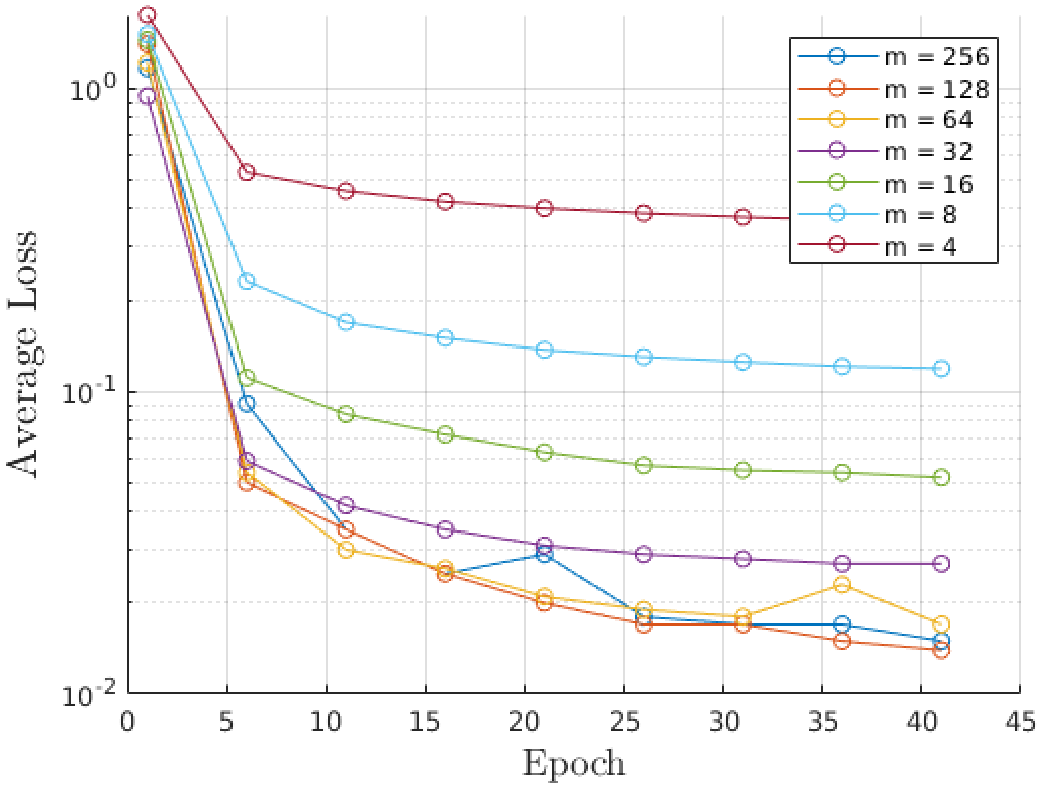

Figure 7.4: Latent representation learned by the CVAE for different dimensions of the latent variable $m$.

the terrain height map as more complex models with $m \in \{32, 64, 128\}$. Therefore, we choose $m = 16$ as the latent variable used for the CVAE during the training of the RL policy. This choice is further supported by the results in Fig. 7.6, where we present the reconstruction of a terrain height map sample with different values of $m$. As expected, the CVAE with $m = 16$ is the model with the lowest latent dimension that can still produce an accurate reconstruction of the terrain height map.

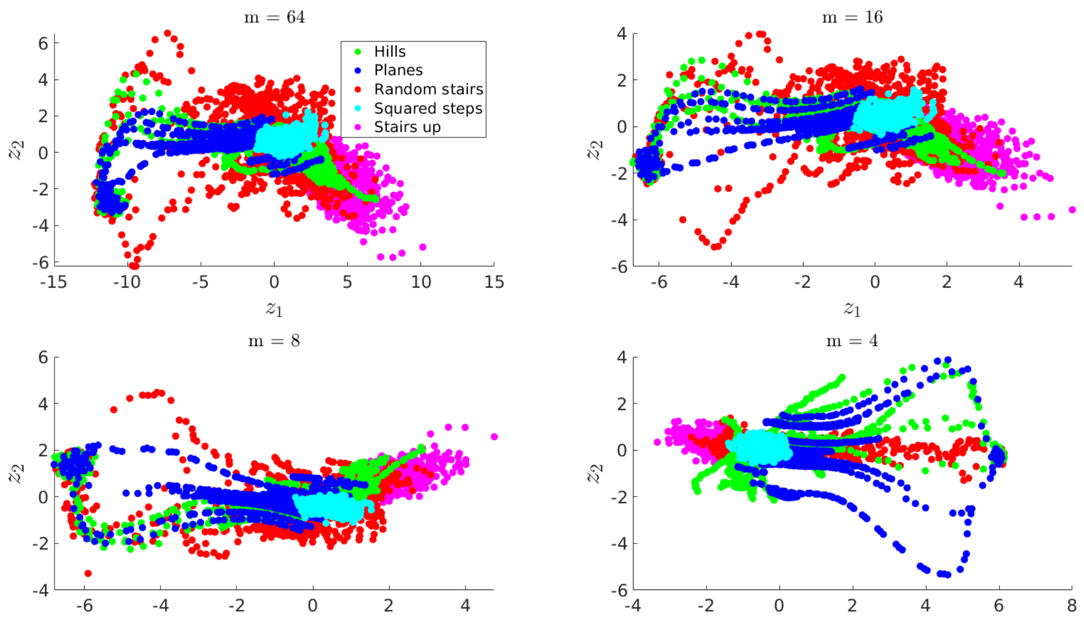

Figure 7.5: Latent representation learned by the CVAE for different terrain types.

## 7.4 Simulation Results

In this section, we show the performance of the proposed hierarchical controller in simulation with a 3D bipedal robot Digit. The learned HL policy is tested in different terrains, including narrow and wide stairs, slopes, wavy terrain, and irregular steps with random heights.

### 7.4.1 Learning convergence

We demonstrate the effectiveness of the latent representation of the terrain by conducting an ablation study of the effect of this latent terrain representation on the efficiency and effectiveness of the learning process. In Fig. 7.7, we present the evolution of the reward during the RL training for different values of the latent dimension

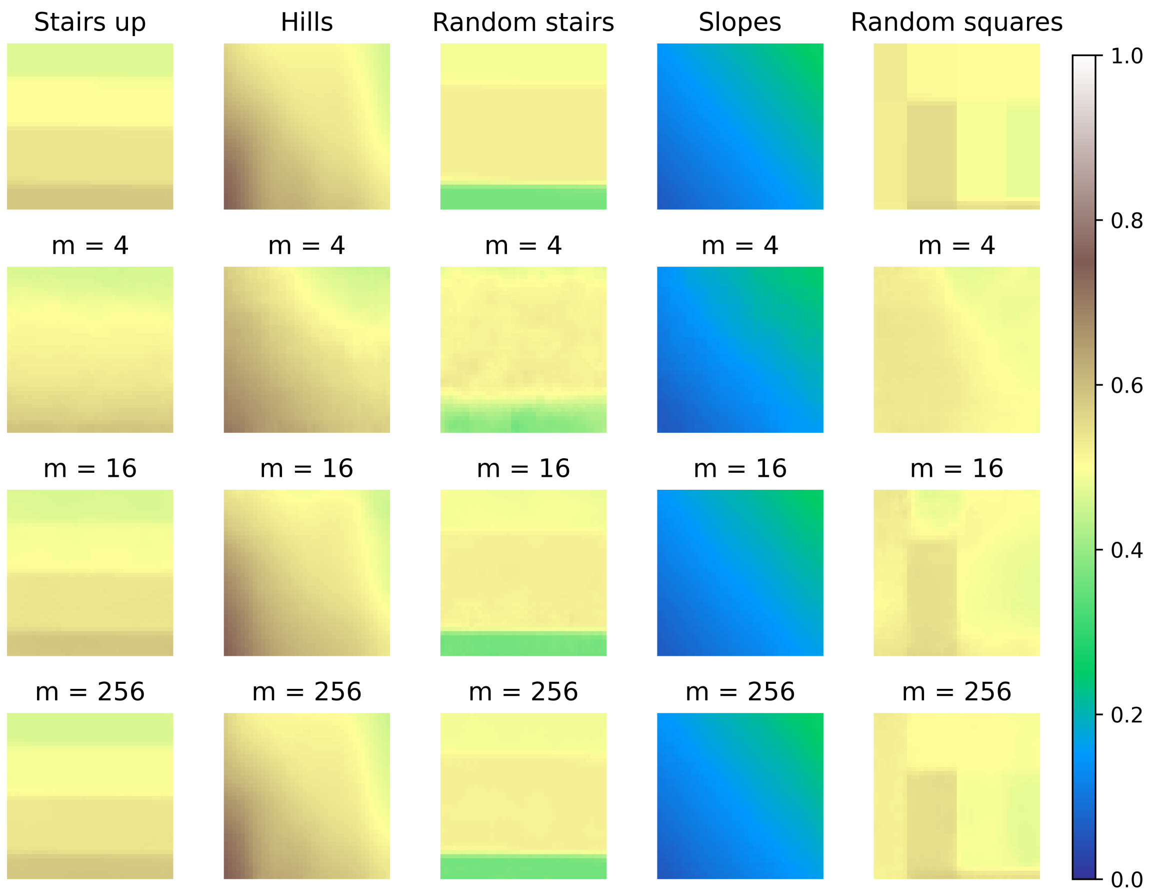

Figure 7.6: Reconstruction of the local height map for different terrains and latent dimensions.

$m$. Notably, the reward curves with better learning efficiency (fewer number epochs to converge) are the ones corresponding to $m \leq 32$, while the reward curves corresponding with $m \geq 64$ show slower convergence to a lower value. In addition, we replace the latent representation of the terrain with the complete local height map matrix, which results in significantly decreased sample efficiency and policy performance, as shown in Fig. 7.7.

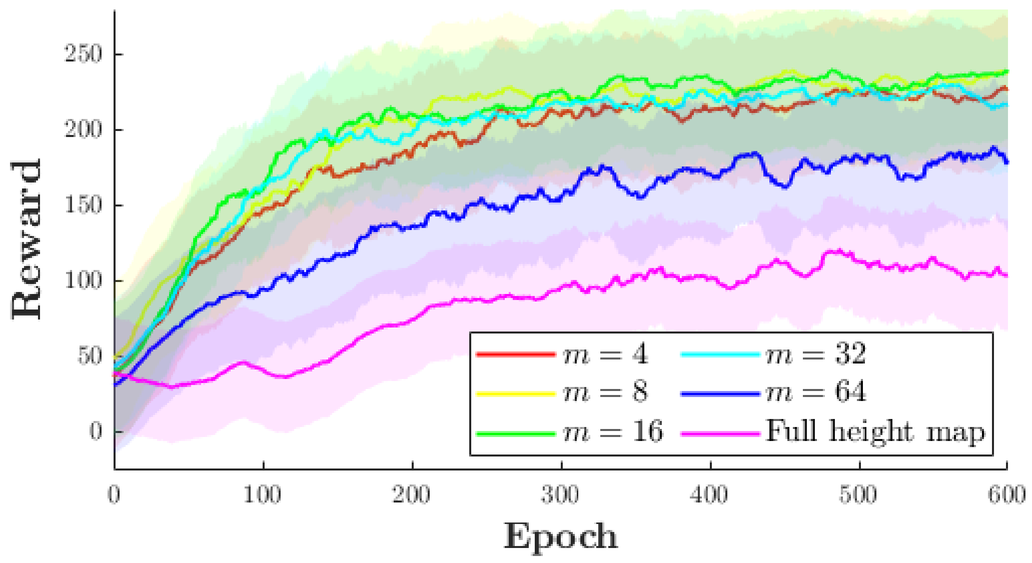

Figure 7.7: Reward convergence for different values of the latent variable dimension.

### 7.4.2 Policy performance

We denote the RL policy learns to walk from scratch and that one single trained policy can successfully navigate on a variety of terrains as shown in Fig. 7.1. When walking over irregular terrains like stairs or square steps, as shown in Fig. 7.8, the policy adapts the feet' landing location by taking shorter or longer steps when needed to avoid collisions with the edges of the terrain. Similarly, the policy adapts the foot location to compensate for the heavy robot's inertia to avoid falling, i.e., when taking a step up or down the stairs. These adaptive strategies naturally emerged during the training from the effective integration of the terrain features into the RL policy and the combination of rewards.

While the policy successfully navigates on all the terrains shown in Fig. 7.6 without falling, there is a tradeoff between robustness and tracking of the commanded walking speed. The complexity of the terrain causes a higher tracking error in some

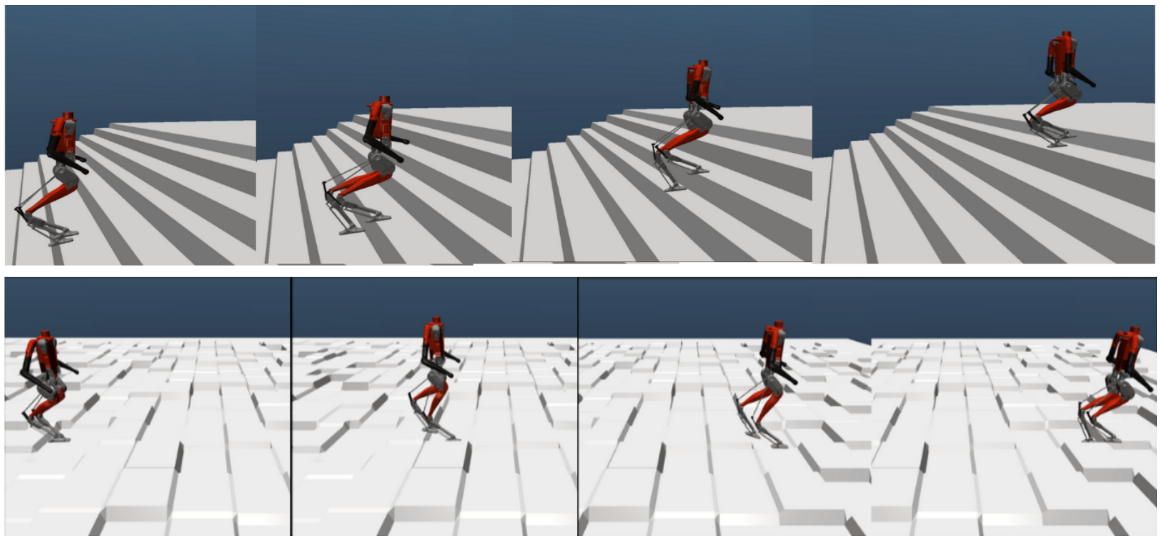

Figure 7.8: Learned policy performance on irregular terrains.

of the terrains since the policy sacrifices the velocity tracking performance to guarantee the robustness of the walking gait. This is particularly evident in Fig. 7.9, where the desired and average speed of the robot are shown for all the terrains. Despite the irregularities in the terrain, the policy adapts its behavior to keep close track of the target speed, except in cases where the terrain conditions are too challenging, e.g., the hill terrain, where the terrain inclination is too high, forcing the policy to take a step back to avoid falling.

Furthermore, we analyze the contribution of the latent representation of the height map by keeping the input of the CVAE fixed during the whole episode. This effect is equivalent to making the policy "see" flat ground even when the robot is walking on nonflat terrains. Under these conditions, the policy fails to account for terrain height variations, and the robot immediately falls after tripping over the edges of the stairs. In Fig. 7.9, this case is labeled as "Stairs fail."

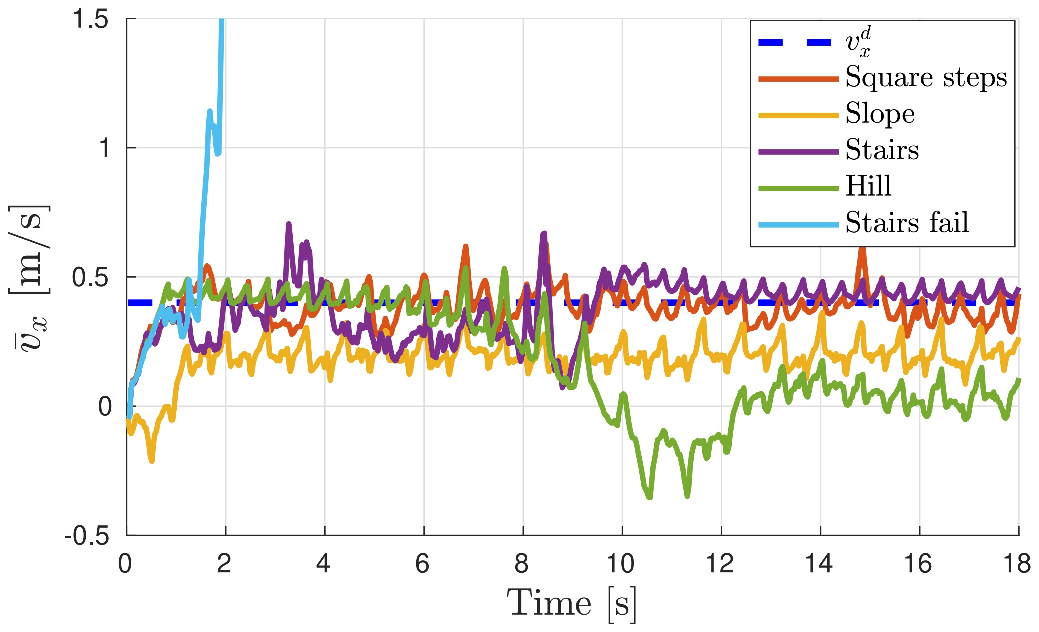

Figure 7.9: Learned policy performance on different terrains.

### 7.4.3 Robustness and comparison

We test the policy's robustness across diverse terrains for at least two hundred experiments per terrain type. The resulting success rate for each type of terrain is shown in Fig. 7.10 with a confidence interval $\geq 95\%$. This consistency in success across a spectrum of terrains highlights the policy's capability to effectively navigate and adapt without the need for terrain-specific tuning. These results also demonstrate the policy's repeatability and reliability, which are essential qualities for humanoid robots to enable their deployment in environments made for humans.

In addition, we compare the performance of our approach against two baselines (policy A, and policy B) that share the same policy architecture as our proposed method with two modifications. Policy A corresponds to the case of blind locomotion, where the policy does not have a meaningful representation of the upcoming terrain.

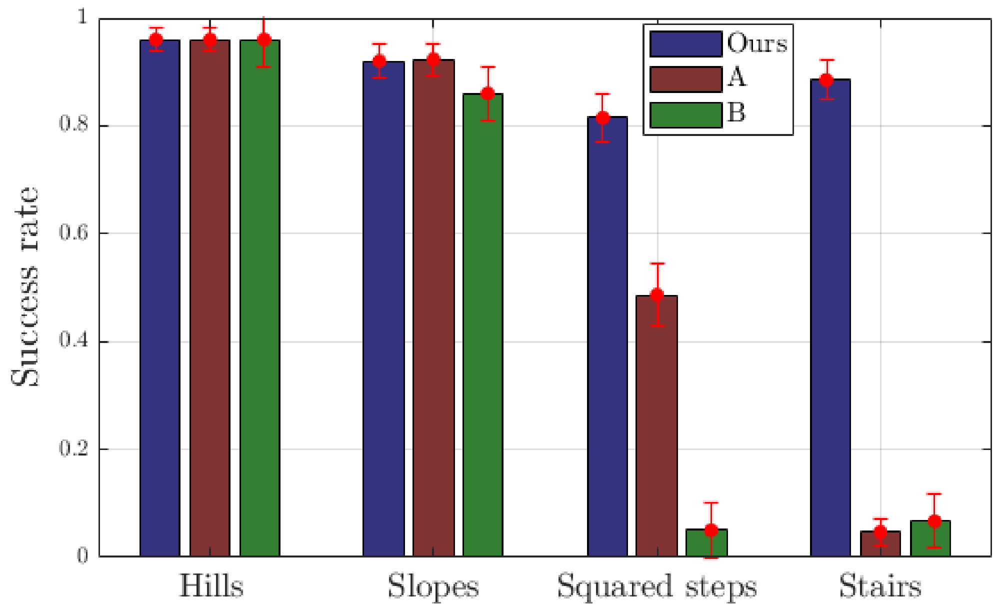

Figure 7.10: Robustness of the policy to different terrains.

By keeping the input of the local height map to a fixed value corresponding to flat terrain, the policy ”thinks” it is walking on flat ground. The policy is robust enough to handle terrains with hills and slopes, which is consistent with several works on blind bipedal locomotion, where it has been shown the potential of RL policies to navigate on these types of terrains without the use of exteroceptive feedback. However, its success rate drops significantly for terrains with random square steps and stairs.

On the other hand, policy B corresponds to the case where the full terrain height map $\mathbf{x} \in \mathbb{R}^{20\times 40}$ is included in the policy state. While this approach has been successfully applied in other end-to-end RL frameworks for bipedal locomotion [90, 165], we find that it is not compatible with our proposed compact and sample-efficient framework. We hypothesize that the lack of structure in the data of the raw terrain height map results in a bottleneck for learning effective actions. This effect is observed in Fig. 7.7, where the reward curve for the policy with the full height map converges significantly slower to a smaller value compared with the policies that use the latent representation of the terrain height map. To alleviate this effect, the work in [90] uses a basis RL policy trained according to [83] and uses the complete height map along with the full-order robot's state to learn compensations added to the basis RL policy.

## 7.5 Conclusion

We propose a framework for learning terrain-aware perceptive locomotion for bipedal locomotion. Our proposed approach integrates a latent representation of the local height map with a reduced-order representation of the robot's dynamics for bipedal locomotion to form an efficient state representation for terrain-aware locomotion and uses task space trajectories to formulate effective actions. By combining a learning-based HL terrain-aware planner with an LL model-based controller, we obtain a robust controller capable of traversing challenging terrains while preserving a good speed-tracking performance. We provide a detailed analysis of the selection of the latent space dimension with empirical demonstrations that a bigger dimension of the state is not necessarily better but the capability of the latent representation to capture the important features of the terrain. Future work will focus on the hardware implementation with the robot Digit.

# Chapter 8: Sim to real

Reinforcement Learning techniques require significant amounts of data to learn policies that can effectively solve the required tasks. The complexity of the task depends on several factors, including dimensionality of the observation and action space, sparsity of the rewards, dynamics of the underlying system, exploration difficulty, safety, and risk factors. The more complex the task, the more experience is required. Using real robotics systems to acquire such amounts of experience is not feasible for most robotics applications, especially for applications where data collection is expensive and dangerous, such as bipedal robots. While there has been significant research in the last years to develop offline RL frameworks that can train policies purely from data collected offline [167], and sample efficient algorithms that can accelerate the exploration process [168] or learn directly on real-world systems [169,170], online RL trained on simulation is still one the most preferred methods for training RL robotics systems.

Using simulations to train RL policies allows us to easily generate significant amounts of experience data by harnessing the faster-than-real-time capabilities of modern physics engines [130] and the parallelization of multiple simulated environments, resulting in substantial improvements in training time [171, 172]. Moreover, the integration of simulators with accelerating computing hardware has allowed the massive parallelization of environments. to reduce the training time from days to hours and even minutes [173].

One of the main challenges in applying RL to real-world robotics is the sim-to-real gap. This gap is characterized by the differences between the simulated environment, where the RL model is initially trained, and the real world, where the model is ultimately deployed. In bipedal locomotion, this gap is significant due to the complex interaction of robotic components with varied terrains and environmental conditions. The research done to bridge this gap, known in the literature as sim-to-real transfer, is crucial for the successful application of policies trained on simulation to physical robots. In particular, sim-to-real transfer in RL refers to the process of training a robotic control system in a simulated environment and then transferring the learned policy to a real-world robot without -or very little- additional training or tuning on the hardware.

In this chapter, we discuss some important details in the implementation of the frameworks presented in Chapters 4-7 to reduce the sim-to-real gap. In particular, we provide details about the implementation of our controllers and the software-hardware integration used in our frameworks. Moreover, we discuss sim-to-real transfer techniques, such as dynamic randomization and curriculum training, and the hardware results with the robot Digit.

## 8.1 Controller implementation on hardware

We provide details about the integration of the high-level and low-level controllers of the control-learning framework introduced in our works [18, 23, 95]. Moreover, we

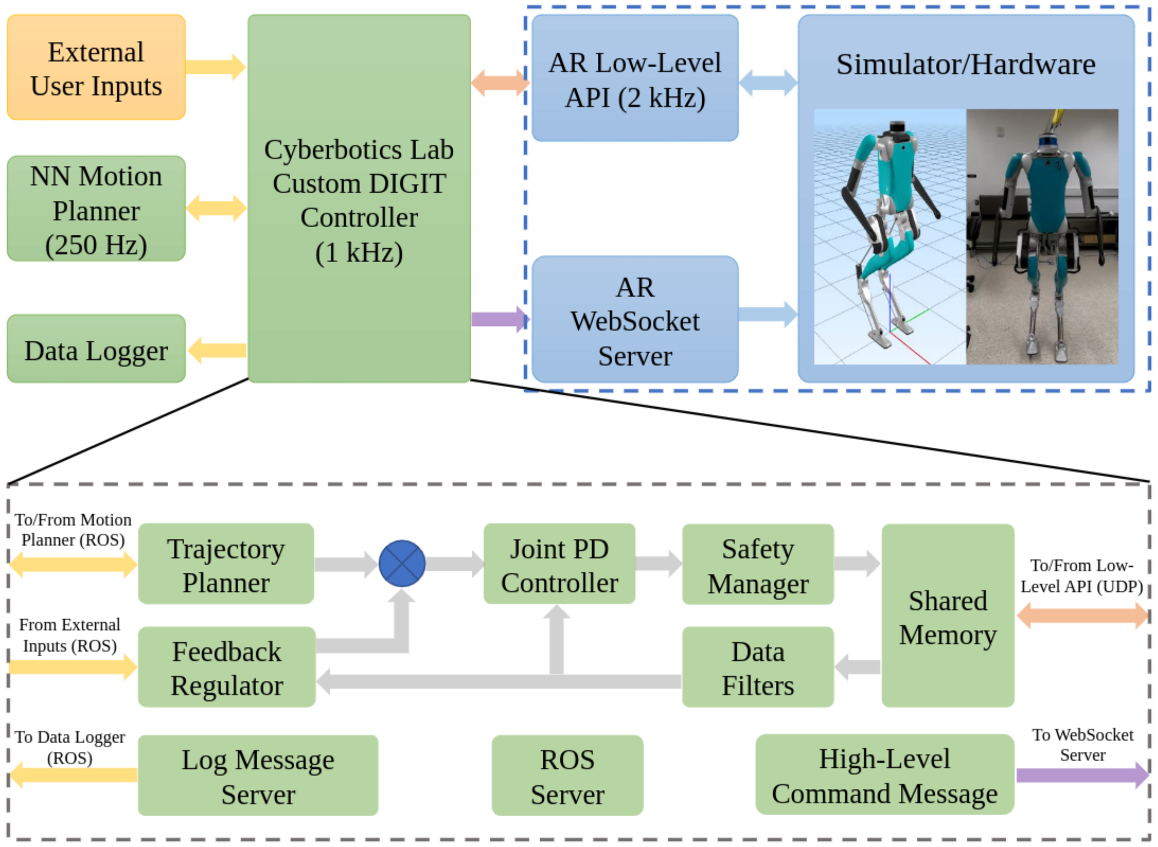

Figure 8.1: Our controller architecture allows asynchronous operation of the high-level and low-level controllers. Moreover, it enables fast and reliable communication between different control layers, external user inputs, and the Digit simulator/hardware. Note that the same controller architecture can be used for different tasks, e.g., walking, standing, crouching.

describe the architecture of our controller and its integration with Digit's low-level API and communication system. This architecture is presented in Fig. 8.1.

The Agility Robotics low-level API utilizes the UDP protocol to stream data, facilitating access to low-level sensor data and direct command issuance to motor drives. To maintain an active connection with the low-level API for continuous sensor data reception, periodic command transmission is essential. Once connected and engaged in command sending, torque control activation is achieved by requesting a transition to low-level API operation through JSON messages communicated via the WebSocket protocol.

Additionally, our system leverages the Robot Operating System (ROS) to coordinate the communication between various components. This integration enhances the versatility of our controller architecture, enabling functionalities such as data logging, external command reception, and interfacing with Digit's perception system.

Since the low-level API needs to run at a very high frequency (2 kHz), we use shared memory to enable fast and reliable communication between the low-level API and our custom main control code (1 kHz). The low-level API reads the sensor measurements and writes them into the shared memory. It also reads from the shared memory the torque information and velocity commands written by the main control code and sends the commands to the motors.

The main control code manages the integration and synchronization of all the components in the overall control structure. It retrieves the sensor data from the shared memory and publishes it in ROS topics to make it available for the other components of the systems, such as the high-level planner and the low-level controller. For the works presented in this dissertation, the high-level planner is the RL policy, which runs in a separate node, retrieving the information from the corresponding ROS topics to compute the high-level commands at a lower frequency compared with the low-level planner. Then, the high-level planner node publishes the desired high-level commands in the corresponding ROS topics to make the data available to the other nodes in the ROS network. Depending on the controller framework, these commands could be high-level actions in the joint space or action space. In the framework presented in Chapter 4, the high-level commands are the Bézier coefficients used to compute the corresponding Bézier Polynomials that become the reference trajectories for the robot's joints. In the frameworks presented in Chapter 5 - 7, the high-level commands are task space actions, including swing foot landing location, foot clearance, CoM height, and torso orientation.

The low-level controller takes the sensor feedback and the trajectories from the Trajectory planner module to compute the torque needed to track the desired joint space or task space trajectories. Similar to the high-level module, the low-level module could look different depending on the particular structure of the control framework. For example, in the framework proposed in [18, 23], the low-level controller is composed of a feedback regulator module and PD controllers at the joint level. The integration of these low-level feedback regulations into the learning framework is a key part that makes the controller structure very robust to uncertainties in the model and makes possible an almost zero effort transfer from the policy learned in simulation into the real hardware. In the frameworks proposed in [95, 174], the low-level controller is composed of an inverse dynamics whole-body controller that tracks the desired task space trajectories for the robot's feet and CoM.

### 8.1.1 Learning pipeline

Although standard sim-to-real transfer techniques like domain randomization and learning curriculum are useful to increase the robustness of the policy, there are also other things important to consider when training an RL policy. In order to realize a policy that is transferable to the actual robot, we need to create a pipeline for the training process in simulation that renders working conditions as close as possible to

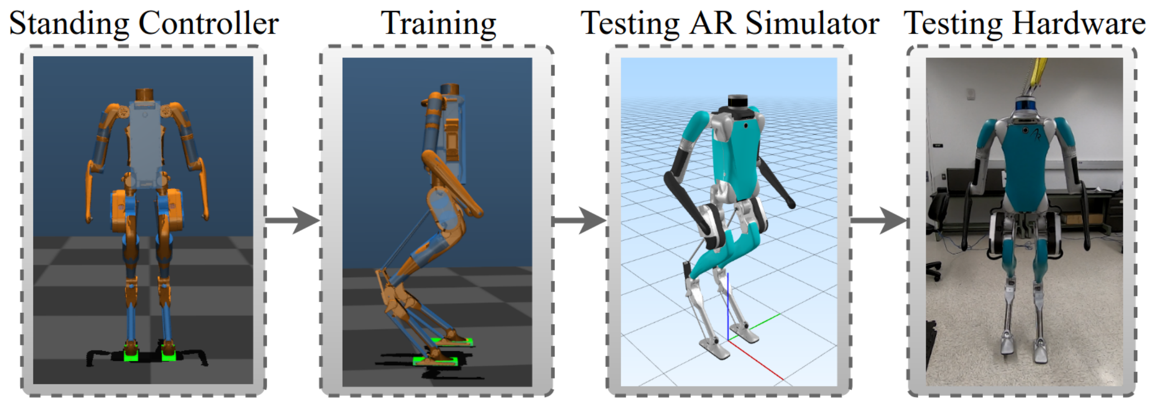

Figure 8.2: Pipeline of the proposed learning framework. A standing controller is included within the pipeline of the learning process to obtain a feasible set of initial states for the policy training using a customized Mujoco environment. Then, the trained policy is tested in a more realistic real-time simulator. Finally, it is tested on hardware.

the real hardware. A diagram of the training pipeline used in the works proposed in this dissertation is shown in Fig. 8.2.

The initial state for each training episode is chosen randomly from a pool of initial states whose kinematics and dynamics are feasible to implement on the real robot. To achieve this, we utilize a standing controller that allows the robot to start up from an arbitrary resting position when the robot is hanging up from the lab's crane [23]. We tested the controller successfully in hardware and used it to replicate the standing process in simulation.

After the standing controller is activated and the robot is able to stand and balance steadily in simulation, we capture the state of the robot and save it in a pool of initial states. We repeat the process 40 times to create a diverse enough set of initial states that we can use for training our controller. It is important to denote that this process does not provide a fixed initial state for the walking gait but a whole set of different initial configurations that encourage the randomness of the initial state during the training while providing feasible and safe starting conditions for the walking gait. Moreover, this process does not provide any reference trajectories for the walking gait that could bias the learning process to specific or predefined walking gaits. Different from other RL approaches to bipedal locomotion, our method learns walking gaits from scratch without the need for predefined reference trajectories or imitation learning.

Given the pool of initial states, we use a customized environment using MuJoCo [130] to start the training process. We denote that the proposed learning pipeline is very general, and any learning algorithm that handles continuous action spaces could be used, including evolution strategies, and gradient-based methods like PPO.

When the training is finished, we test the trained policy in the Agility Robotics' proprietary simulation software. This simulation software provides a more realistic representation of the robot's dynamic behavior since it runs in real-time and shares the same features with the hardware, such as low-level API, communication delay, and dynamic parameters of the robot.

Finally, once we verify the trained policy is deployed safely in the AR simulation, we proceed to test it in the hardware and verify its performance to do additional tuning of the low-level controller gains if needed.

## 8.2 Sim-to-real transfer results

In this section, we show the effectiveness of the learning pipeline introduced above to realize RL policies that can work effectively on hardware with minimal tunning. We compare the simulation and hardware results of the policy trained using the framework proposed in Chapter 4. Moreover, we analyze the performance of the policy in hardware for different test cases, including robustness to external forces and robustness to challenging terrain.

### 8.2.1 Stability and feasibility of the walking gait

Figure 8.3 shows the phase portrait of the actuated joints of the robot's leg while walking at a constant desired velocity. Similar to the simulation results, the plot shows the convergence of the orbits to a periodic limit cycle, empirically demonstrating the stability of the walking gait. As expected, the limit cycles of the joints are noisier than the ones obtained in the simulation, particularly for the joints that are being modified by the feedback regulator policy.

### 8.2.2 Robustness

We perform two tests to evaluate the robustness of the cascade controller: i) robustness to external disturbances and ii) robustness when walking on challenging terrain. For the first test, external disturbances are applied to the robot's torso while walking forward ($v_x = 0.11m/s, v_y = 0m/s$). Figure 8.4 shows the performance of the controller to keep tracking the desired walking speed, while Figure 8.5 shows the limit walking cycle of some of the robot joints before, during, and after the disturbance. These results show the controller can recover effectively from disturbances while keeping a good tracking performance of the desired walking speed and maintaining the stability of the walking limit cycle.

For the second test, we set Digit to walk blindly on a series of challenging irregular terrains. To test the robustness of the controller on different slopes, we conducted

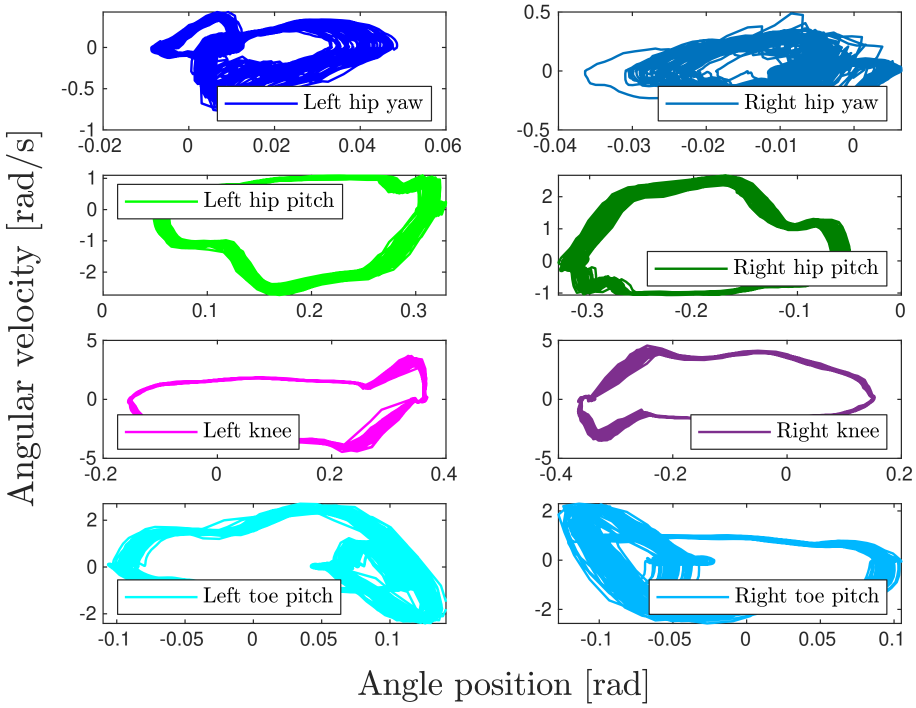

Figure 8.3: Walking limit cycle of the learned policy when tracking a longitudinal velocity $v_x^d = 0.0\ m/s$, and lateral velocity $v_y^d = 0\ m/s$.

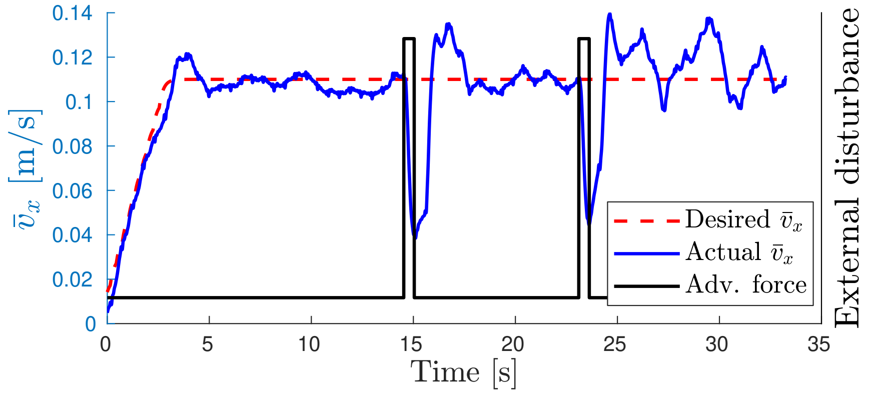

Figure 8.4: Disturbance rejection when external forces are applied in the backward direction

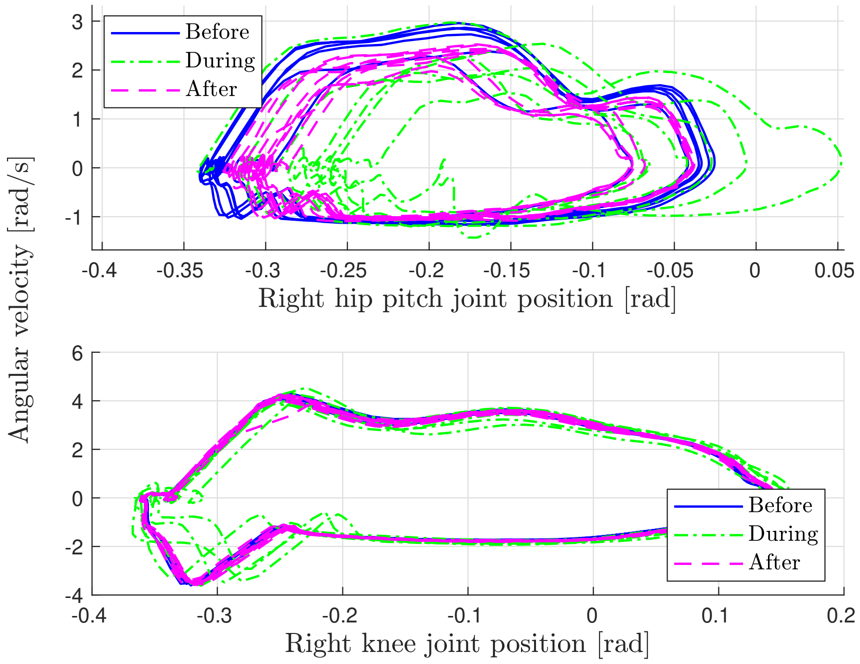

Figure 8.5: Disturbance rejection when adversarial forces are applied in the forward and backward direction

rigorous experiments on a treadmill varying the slope inclination from 0 to 11 degrees. In addition, we evaluate the controller's performance to real-world scenarios by conducting experiments outdoors on different terrains, including concrete ground, vinyl, pavement, grass, and slopes of different inclinations. Figure 8.6 shows tile plots

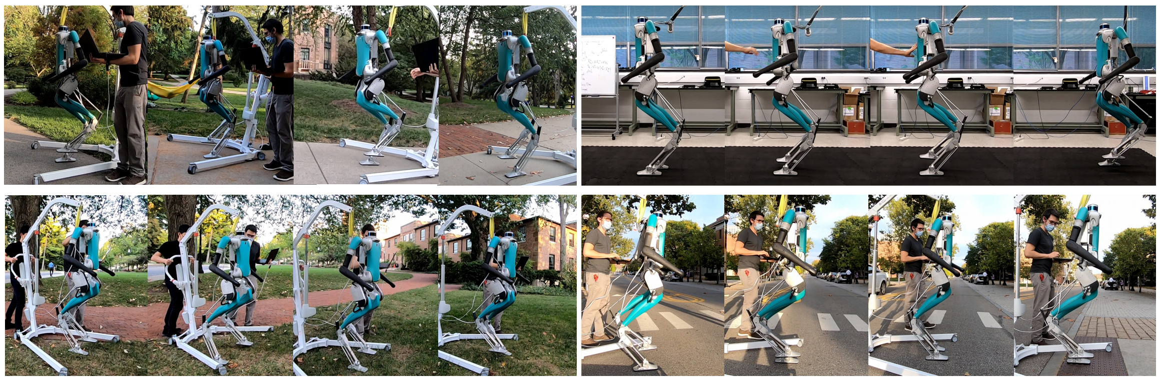
Figure 8.6: Digit walking on different terrains using the learned policy: concrete slopes (top-left), rubber surface (top-right), grass (bottom-left), and pavement (bottom-right). The top-right tile plots also show the robustness of the policy against external forces applied to the robot's torso.

of the robot walking on some of these terrains. More details about these experiments can be seen in the video: https://www.youtube.com/watch?v=ocAZtHr07Fw.

We evaluate the speed-tracking performance of the controller along all the different terrains. The results presented in Figure 8.7 show the proposed controller structure is able not only to keep stable walking but also to keep a good speed tracking performance on every single terrain. This demonstrates that our learned policy can be used with confidence for navigation in real-world scenarios.

We denote that the same learned policy is used to navigate the robot in all the terrains mentioned above without the need for additional training or tuning between different terrains. It is important to note that no disturbances or terrain randomization were applied during the training. Therefore, the robustness of the policy is the result of the enhanced structure of the controller that allows the external and internal loops to be updated at different rates. The inner loop (feedback regulation) facilitates the feedback response of the controller to external disturbances while the

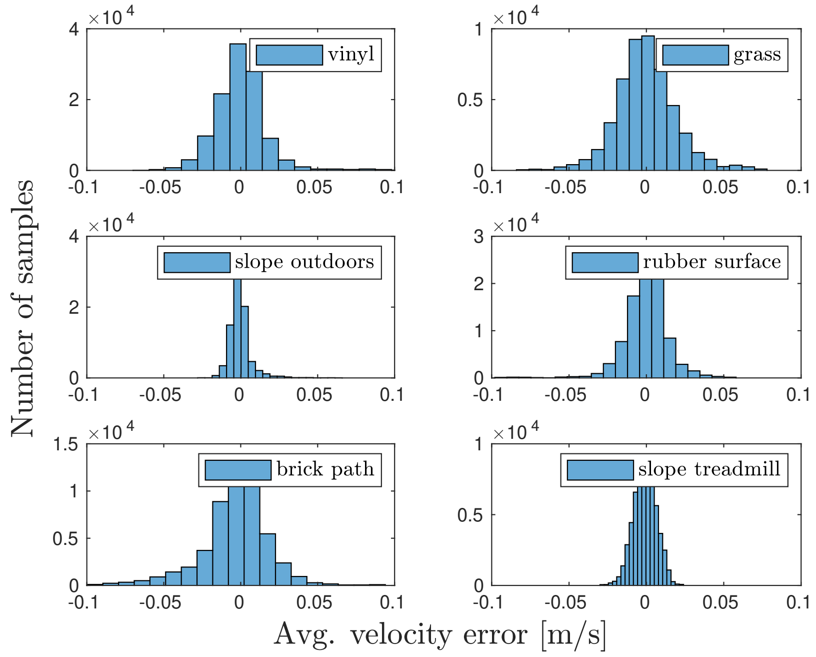

Figure 8.7: Speed tracking performance of the controller while walking blindly on different challenging terrains.

outer loop (NN-based trajectory planning) keeps updating the reference trajectories for different desired speeds at a lower rate.

## 8.3 Towards Standardized Disturbance Rejection Testing - A Linear Impactor Approach

This section further showcases the sim-to-real capabilities of the policies trained with the RL frameworks proposed in this dissertation. Moreover, we discuss an important problem in the legged robots community that is directly connected with the testing of locomotion controllers in real scenarios.

Even though legged robots seem to be ready for industrial collaboration, there is still one critical thing that is hindering their spread commercialization: the lack of standardized testing. Contrary to the vehicle industry, there are limited studies regarding standard testing equipment and infrastructure for legged robots, which becomes evident when we analyze the experimental and robustness tests of legged locomotion in the last decades. Historically, the methods used for testing legged robots, such as hand-pushing, foot-kicking, rope-dragging, stick-poking, and ball-swinging, have lacked standardization.

In the work presented in [175], we analyze this lack of standardization problem and propose alternatives for the design of standard methods that guarantee the repeatability, adaptability, and accuracy of the disturbance rejection testing. In particular, we propose the use of the linear impactor, a well-established tool in other standardized testing disciplines. We use a pneumatic linear impactor to test three different locomotion controllers on the Digit robot. These controllers correspond to a template model-inspired whole-body controller (TM), a manufacturer controller provided by Agility Robotics (AR), and our template model-based task space learning controller (TL) introduced in Chapter 5.Fig. 8.8 shows the overall configuration used during the testing experiments. The center of the impactor face is 99.2 cm from the ground. The impactor weight is 6.4 kg. The impactor face is a rectangle of 6-inch $\times$ 4-inch (about 15.24 cm $\times$ 10.16 cm) [175].

This work not only showcases the sim-to-real capabilities and robustness of our controller to withstand extensive and standardized testing, but also highlights other anti-intuitive observations, demonstrations, and implications that, to the best of the authors' knowledge, are first-of-its-kind revealed in real-world testing of legged robots. For example, Fig. 8.9 shows the results of 36 impact tests against the three different

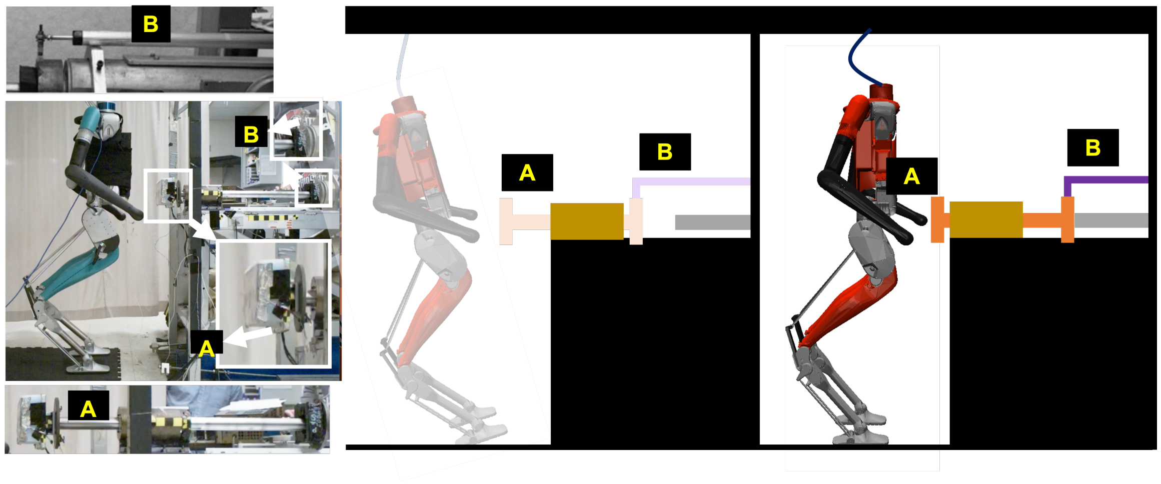

Figure 8.8: The testing configuration in-lab and the conceptual illustration before and after the impact: "A" denotes the impactor face with a rectangular surface, "B" denotes the displacement potentio-meter.

locomotion controllers. There are multiple regions where impact events sharing similar impact velocities end up with different fallover outcomes (indicated by the marker size). Analyzing this results one could observe that a more significant impact does not necessarily lead to a more severe outcome.

In particular, the TL-driven robot recovers from the impact velocities of 3.214 m/s and falls over against the 3.216 m/s velocity, and that is only 0.0128 kg $\cdot$ m/s difference in impact momentum. A similar pattern is seen with AR at notably higher impact velocities: 3.89 m/s, 4.23 m/s, and 4.26 m/s. While the robot falls in two out of the three impacts, it is noticeable that the falls do not correspond to the highest velocity impacts. Moreover, these counter-intuitive cases highlight the challenges in developing fair and effective safety performance testing algorithms [176].

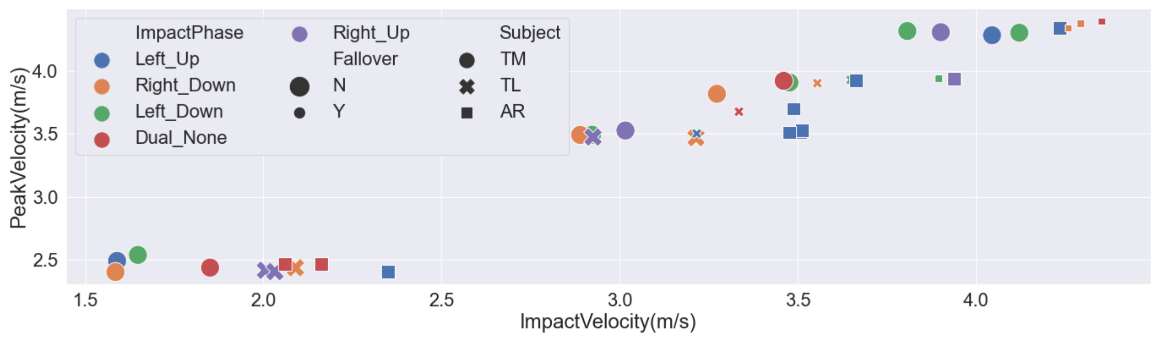

Figure 8.9: An overview of all 36 impact tests against three locomotion controllers: different impact phases are differentiated by the marker color and the label is specified in the form of stance foot phase (whether the left or the right foot is on the ground) and swing foot phase (whether the swing foot is going up or stepping down) with an underscore "_" in between. Note the dual support phase (with both feet on-ground) is specified as "Dual_None". The fallover status is characterized by the size of the marker. The three different locomotion controllers are differentiated by the marker type.

.

## 8.4 Conclusion

In this chapter, we present the learning pipeline used to implement the frameworks presented in this dissertation and demonstrate its effectiveness in realizing RL policies with successful sim-to-real transfer. Moreover, we discuss in detail the implementations of our controllers in the robot Digit, showing the integration of the high-level RL policy with the low-level feedback controller. We emphasize the importance of having a pipeline that is consistent with the hardware setup used to test the policies in real-world experiments and demonstrate that our proposed pipeline realizes policies that successfully address the sim-to-real gap challenge. Notably, we showcase the robustness of the learned policies with hardware experiments of the robot Digit walking on different terrains at different speeds around the OSU campus.

Finally, we discuss the importance of standardized methods for disturbance rejection and robustness and show the application of one of our controllers for a preliminary study where a linear impactor is used as a proposed approach for testing standardization (repeatable, fair, measurable), showing the robustness of our controller. These experiments have revealed interesting insights, challenges, and lessons, all contributing to an expected future of standardized safety performance testing for legged robots.

# Chapter 9: Conclusions

In this dissertation, we have explored advanced methodologies for achieving robust bipedal locomotion, addressing key challenges through innovative reinforcement learning and hierarchical learning-based frameworks. Our research has made important contributions to the intricate balance between theoretical insight and practical application of model-based and model-free methods for bipedal locomotion, leading to the development of dynamic and adaptable locomotion strategies.

Our first contribution is the integration of Hybrid Zero Dynamics with reinforcement learning, culminating in a novel framework that not only simplifies trajectory planning and feedback regulation but also enhances the adaptability and efficiency of bipedal robots. This approach, inspired by the use of reduced state and action spaces with additional heuristic regulations, has demonstrated remarkable improvements in sample efficiency and training time, setting a new benchmark for the field. By integrating dynamic walking insights into the RL process, a sophisticated yet efficient solution for designing feedback controllers was developed, capable of achieving stable and robust walking gaits without reliance on predefined trajectories.

Further exploration led to the development of a template-model-inspired task space learning approach, offering a hierarchical learning-based method that leverages a reduced-order state inspired by the ALIP model with a set of task space commands and a model-based whole-body controller for trajectory tracking. This approach demonstrated versatility across different robot configurations, including fully actuated and underactuated robots, proving its effectiveness in enabling dynamic locomotion under challenging conditions.

Further, our research into data-driven latent space representation has revealed the potential of hierarchical and data-driven approaches in refining locomotion control by harnessing locomotion data extracted from existing model-based controllers. These strategies have proven effective in developing compact RL policies that can extrapolate to scenarios out of the distribution of the collected data, showcasing a significant leap forward in developing versatile and robust data-driven locomotion controllers.

A pivotal contribution of our work is the successful implementation and validation of these methodologies in real-world scenarios, particularly through the sim-to-real transfer on the robot Digit. This not only attests to the practical viability of the learning pipeline developed throughout our research but also opens new avenues for the application of our controllers in higher-level navigation systems that allow the deployment of bipedal robots in indoor and outdoor scenarios.

This dissertation also ventured into terrain-aware locomotion, presenting a framework that combines latent representations of the terrain with reduced-order robot dynamics. This method emphasized the ability of the high-level RL policy to adapt to complex terrains while maintaining robust and agile locomotion, showcasing the importance of the dimension of the latent space in the robustness of the policy and sample efficiency of the learning process.

In essence, our thesis contributes to the advancement of bipedal robot locomotion by proposing and validating frameworks that combine the theoretical rigor of dynamic systems with the practical robustness required for real-world application. These innovations pave the way for the development of more autonomous, efficient, and adaptable bipedal robots, potentially transforming their application in essential sectors of industry and society.

## 9.1 Future work

While our research has made significant strides in advancing bipedal locomotion for real-world scenarios, there are several lines of research that could be explored to extend the findings and applications of this thesis:

**Real-World Hardware Testing and Application:** Future initiatives should focus on extending the application of our frameworks to a broader array of bipedal robots, including those designed for specific tasks or environments such as exoskeletons. This will not only validate the universality of our approaches but also highlight potential areas for refinement.

**Integration of Advanced Sensory and Perceptive Capabilities:** Incorporating sophisticated sensory and perception mechanisms can enhance the adaptability and decision-making capabilities of bipedal robots, especially in dynamic or unknown environments. Future work could explore the integration of these technologies and the development of more sophisticated state representations and actions that can navigate these terrains with even greater efficiency and adaptability.

**Expansion of Locomotion Behaviors:** Extending our frameworks to encompass a wider range of locomotion behaviors, such as running, jumping, or walking over stepping stones, represents an exciting frontier. This could involve the development of new learning models or the integration of existing ones into our hierarchical learning-based approaches.

**Integration of Large Language Models for Enhanced Decision-Making:** Integration of Large Language Models for Enhanced Decision-Making: The advent of sophisticated Large Language Models (LLMs) opens new horizons for robotic systems, particularly in complex decision-making scenarios. Future research could explore the integration of LLMs as high-level decision tools within our framework, enhancing the robot's ability to make informed decisions about locomotion, manipulation, and task execution based on a vast array of data and contextual understanding. This approach could leverage the natural language processing capabilities of LLMs to interpret and analyze instructions, environmental descriptions, or task requirements, translating them into actionable strategies for the robot.

Embarking on these future endeavors will undoubtedly enrich the field of bipedal locomotion, driving forward the capabilities of robots and opening new possibilities for their application in society. I am confident that the path ahead is filled with potential for groundbreaking discoveries and innovations that will continue to push the boundaries of what is currently possible.

# Bibliography

[1] A. Keay, "Humanoids hit a critical mass moment!" 2023. [Online]. Available: https://tinyurl.com/4ezheamt

[2] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. D. Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, pp. 1–20, 2023.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: The MIT Press, 2018.

[4] Y. Gong and J. W. Grizzle, "Zero Dynamics, Pendulum Models, and Angular Momentum in Feedback Control of Bipedal Locomotion," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, no. 12, p. 121006, 10 2022.

[5] D. Graas, "Steam carriage," 1868. [Online]. Available: https://patents.google.com/patent/US75874A/en

[6] M. F. Silva and J. T. Machado, "A historical perspective of legged robots," *Journal of Vibration and Control*, vol. 13, no. 9-10, pp. 1447–1486, 2007.

[7] M. H. Raibert *et al.*, *Legged robots that balance.* MIT press Cambridge, MA, 1986, vol. 3.

[8] Markets and Markets, "Humanoid robot market size, share, statistics and industry growth analysis report by component, motion type, application and region - 2028," 2023. [Online]. Available: https://tinyurl.com/yunmw4se

[9] 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," in *4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, Conference Track Proceedings*, 2016.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[11] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," *arXiv preprint arXiv:1812.11103*, 2018.

[12] R. Tedrake, T. W. Zhang, H. S. Seung *et al.*, "Learning to walk in 20 minutes," *Proc. Fourteenth Yale Workshop on Adaptive and Learning Systems*, pp. 1939–1412, 2005.

[13] J. Siekmann, S. Valluri, J. Dao, L. Bermillo, H. Duan, A. Fern, and J. Hurst, "Learning memory-based control for human-scale bipedal locomotion," in *Robotics science and systems*, 2020.

[14] G. A. Castillo, B. Weng, A. Hereid, Z. Wang, and W. Zhang, "Reinforcement learning meets hybrid zero dynamics: A case study for rabbit," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 284–290.

[15] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Hybrid zero dynamics inspired feedback control policy design for 3d bipedal locomotion using reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8746–8752.

[16] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback control of dynamic bipedal robot locomotion.* CRC press Boca Raton, 2007.

[17] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic humanoid locomotion: a scalable formulation for HZD gait optimization," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 370–387, Apr. 2018.

[18] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Reinforcement learning-based cascade motion policy design for robust 3d bipedal locomotion," *IEEE Access*, vol. 10, pp. 20 135–20 148, 2022.

[19] E. Andrews, "7 early robots and automatons," 2014. [Online]. Available: https://www.history.com/news/7-early-robots-and-automatons

[20] R. Finlay, "China, the west, and world history in joseph needham's "science and civilisation in china"," *Journal of World History*, vol. 11, no. 2, pp. 265–303, 2000.

[21] A. Marsh, "Elektro the moto-man had the biggest brain at the 1939 world's fair," 2014. [Online]. Available: https://spectrum.ieee.org/elektro-the-motoman-had-the-biggest-brain-at-the-1939-worlds-fair

[22] I. Kato, "Development of wabot 1," *Biomechanism*, vol. 2, pp. 173–214, 1973.

[23] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5136–5143.

[24] L. Krishna, G. A. Castillo, U. A. Mishra, A. Hereid, and S. Kolathaya, "Linear policies are sufficient to realize robust bipedal walking on challenging terrains," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2047–2054, 2022.

[25] Y. Ding, C. Khazoom, M. Chignoli, and S. Kim, "Orientation-aware model predictive control with footstep adaptation for dynamic humanoid walking," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 299–305.

[26] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: system overview and integration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002, pp. 2478–2483 vol.3.

[27] Y. Kuroki, M. Fujita, T. Ishida, K. Nagasaka, and J. Yamaguchi, "A small biped entertainment robot exploring attractive applications," in *2003 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2003, pp. 471–476 vol.1.

[28] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot hrp-2," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 2, 2004, pp. 1083–1090 Vol.2.

[29] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, "Introduction to humanoid robotics," in *Springer Tracts in Advanced Robotics*, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:9017724

[30] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822–10 825, 2008, 17th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016407020

[31] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski, "The darpa robotics challenge finals: Results and perspectives," *Journal of Field Robotics*, vol. 34, no. 2, pp. 229–240, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21683

[32] Toyota, "Toyota unveils third generation humanoid robot t-hr3," 2017. [Online]. Available: https://global.toyota/en/detail/19666346

[33] M. Vukobratovic and J. Stepanenko, "On the stability of anthropomorphic systems," *Bellman Prize in Mathematical Biosciences*, vol. 15, pp. 1–37, 1972. [Online]. Available: https://api.semanticscholar.org/CorpusID:121622699

[34] J. Reher and A. D. Ames, "Dynamic walking: Toward agile and efficient bipedal robots," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 535–572, 2021. [Online]. Available: https://doi.org/10.1146/annurev-control-071020-045021

[35] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, pp. 161–176, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:15473645

[36] P. M. Wensing and D. E. Orin, "Improved computation of the humanoid centroidal dynamics and application for whole-body control," *International Journal of Humanoid Robotics*, vol. 13, no. 1, 2016.

[37] R. Batke, F. Yu, J. Dao, J. Hurst, R. L. Hatton, A. Fern, and K. Green, "Optimizing bipedal maneuvers of single rigid-body models for reinforcement learning," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 714–721.

[38] Z. Zhou, B. Wingo, N. Boyd, S. Hutchinson, and Y. Zhao, "Momentum-aware trajectory optimization and control for agile quadrupedal locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7755–7762, 2022.

[39] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[40] C. Khazoom, S. Heim, D. Gonzalez-Diaz, and S. Kim, "Optimal scheduling of models and horizons for model hierarchy predictive control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9952–9958.

[41] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *IEEE International Conference on Robotics and Automation*, vol. 2, Sep. 2003, pp. 1620–1626.

[42] S. Kajita, T. Yamaura, and A. Kobayashi, “Dynamic walking control of a biped robot along a potential energy conserving orbit,” *IEEE Transactions on Robotics and Automation*, vol. 8, no. 4, pp. 431–438, 1992.

[43] S. Kajita and K. Tani, “Experimental study of biped dynamic walking,” *IEEE Control Systems Magazine*, vol. 16, no. 1, pp. 13–19, 1996.

[44] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, “The 3D linear inverted pendulum model: a simple modeling for a biped walking pattern generation,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 239–246.

[45] S. Kajita, O. Matsumoto, and M. Saigo, “Real-time 3d walking pattern generation for a biped robot with telescopic legs,” in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 3, 2001, pp. 2299–2306 vol.3.

[46] M. Vukobratovic and D. Juricic, “Contribution to the synthesis of biped gait,” *IEEE Transactions on Biomedical Engineering*, vol. BME-16, no. 1, pp. 1–6, 1969.

[47] H. Lim and A. Takanishi, “Biped walking robots created at waseda university: Wl and wabian family,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1850, pp. 49–64, 2007. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2006.1920

[48] T. Sugihara, Y. Nakamura, and H. Inoue, “Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation*, vol. 2, 2002, pp. 1404–1409 vol.2.

[49] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, “Online Walking Motion Generation with Automatic Foot Step Placement,” *Advanced Robotics*, vol. 24, no. 5-6, 2010. [Online]. Available: https://inria.hal.science/inria-00391408

[50] X. Da and J. Grizzle, “Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots,” *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1063–1097, jul 2019.

[51] A. L. Mitchell, W. X. Merkt, M. Geisert, S. Gangapurwala, M. Engelcke, O. P. Jones, I. Havoutis, and I. Posner, “Vae-loco: Versatile quadruped locomotion by

learning a disentangled gait representation," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3805–3820, 2023.

[52] S. Starke, I. Mason, and T. Komura, "Deepphase: Periodic autoencoders for learning motion phase manifolds," *ACM Trans. Graph.*, vol. 41, no. 4, jul 2022.

[53] J. Hu, "Learning control of bipedal dynamic walking robots with neural networks," Ph.D. dissertation, Massachusetts Institute of Technology, 1998.

[54] J. Hu, J. Pratt, and G. Pratt, "Stable adaptive control of a bipedal walking; robot with cmac neural networks," in *Proceedings 1999 IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 1050–1056 vol.2.

[55] N. Csomay-Shanklin, M. Tucker, M. Dai, J. Reher, and A. D. Ames, "Learning controller gains on bipedal walking robots via user preferences," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 405–10 411.

[56] L. Yang, Z. Li, J. Zeng, and K. Sreenath, "Bayesian optimization meets hybrid zero dynamics: Safe parameter learning for bipedal locomotion control," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 456–10 462.

[57] D. Widmer, D. Kang, B. Sukhija, J. Hübotter, A. Krause, and S. Coros, "Tuning legged locomotion controllers via safe bayesian optimization," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 2444–2464.

[58] J. Ahn, S. J. Jorgensen, S. H. Bang, and L. Sentis, "Versatile locomotion planning and control for humanoid robots," *Frontiers in Robotics and AI*, vol. 8, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2021.712239

[59] X. Xiong, Y. Chen, and A. D. Ames, "Robust disturbance rejection for robotic bipedal walking: System-level-synthesis with step-to-step dynamics approximation," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 697–704.

[60] M. Dai, X. Xiong, and A. D. Ames, "Data-driven Step-to-step Dynamics based Adaptive Control for Robust and Versatile Underactuated Bipedal Robotic Walking," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[61] V. C. Paredes and A. Hereid, "Resolved motion control for 3d underactuated bipedal walking using linear inverted pendulum dynamics and neural adaptation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 6761–6767.

[62] H. Benbrahim and J. A. Franklin, "Biped dynamic walking using reinforcement learning," *Robotics and Autonomous Systems*, vol. 22, no. 3, pp. 283–302, 1997, robot Learning: The New Wave. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889097000432

[63] R. Tedrake and H. Seung, "Improved dynamic stability using reinforcement learning," in *5th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR).* Paris, France: Professional Engineering Publishing Limited, September 2002, pp. 341–348.

[64] R. Tedrake, T. Zhang, and H. Seung, "Stochastic policy gradient reinforcement learning on a simple 3d biped," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2849–2854 vol.3.

[65] J. Morimoto, G. Cheng, C. G. Atkeson, and G. Zeglin, "A simple reinforcement learning algorithm for biped walking," in *2004 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3. IEEE, 2004, pp. 3030–3035.

[66] J. Morimoto, J. Nakanishi, G. Endo, G. Cheng, C. G. Atkeson, and G. Zeglin, "Poincare-map-based reinforcement learning for biped walking," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.* IEEE, 2005, pp. 2381–2386.

[67] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[68] D. Kim, G. Berseth, M. Schwartz, and J. Park, "Torque-based deep reinforcement learning for task-and-robot agnostic learning on bipedal robots using sim-to-real transfer," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6251–6258, 2023.

[69] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, Jan. 2019.

[70] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2018, pp. 1241–1246.

[71] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[72] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Trans. Graph.*, vol. 40, no. 4, jul 2021. [Online]. Available: https://doi.org/10.1145/3450626.3459670

[73] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf

[74] Z. Luo, J. Cao, A. Winkler, K. Kitani, and W. Xu, "Perpetual humanoid control for real-time simulated avatars," 2023.

[75] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "ASE: large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 94:1–94:17, 2022.

[76] Z. Luo, J. Cao, J. Merel, A. Winkler, J. Huang, K. Kitani, and W. Xu, "Universal humanoid motion representations for physics-based control," *arXiv preprint arXiv:2310.04582*, 2023.

[77] CMU, "Cmu graphics lab motion capture database," 2002. [Online]. Available: http://mocap.cs.cmu.edu/

[78] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "AMASS: Archive of motion capture as surface shapes," in *International Conference on Computer Vision*, Oct. 2019, pp. 5442–5451.

[79] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "Sfv: Reinforcement learning of physical skills from videos," *ACM Trans. Graph.*, vol. 37, no. 6, dec 2018. [Online]. Available: https://doi.org/10.1145/3272127.3275014

[80] M. Bogdanovic, M. Khadiv, and L. Righetti, "Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization," *Frontiers in Robotics and AI*, vol. 9, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2022.854212

[81] D. Kang, J. Cheng, M. Zamora, F. Zargarbashi, and S. Coros, “Rl + model-based control: Using on-demand optimal control to learn versatile legged locomotion,” *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6619–6626, 2023.

[82] C. Yang, K. Yuan, W. Merkt, T. Komura, S. Vijayakumar, and Z. Li, “Learning whole-body motor skills for humanoids,” in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 270–276.

[83] 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)*, 2021, pp. 7309–7315.

[84] D. Crowley, J. Dao, H. Duan, K. Green, J. W. Hurst, and A. Fern, “Optimizing bipedal locomotion for the 100m dash with comparison to human running,” in *IEEE IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*. IEEE, 2023, pp. 12 205–12 211. [Online]. Available: https://doi.org/10.1109/ICRA48891.2023.10160436

[85] F. Yu, R. Batke, J. Dao, J. Hurst, K. Green, and A. Fern, “Dynamic bipedal turning through sim-to-real reinforcement learning,” in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 903–910.

[86] J. Dao, K. Green, H. Duan, A. Fern, and J. Hurst, “Sim-to-real learning for bipedal locomotion under unsensed dynamic loads,” in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 449–10 455.

[87] H. Duan, A. Malik, M. S. Gadde, J. Dao, A. Fern, and J. Hurst, “Learning dynamic bipedal walking across stepping stones,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 6746–6752.

[88] R. P. Singh, M. Benallegue, M. Morisawa, R. Cisneros, and F. Kanehiro, “Learning bipedal walking on planned footsteps for humanoid robots,” in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 686–693.

[89] J. Dao, H. Duan, and A. Fern, “Sim-to-real learning for humanoid box loco-manipulation,” *arXiv preprint arXiv:2310.03191*, 2023.

[90] H. Duan, B. Pandit, M. S. Gadde, B. van Marum, J. Dao, C. Kim, and A. Fern, “Learning vision-based bipedal locomotion for challenging terrain,” *arXiv preprint arXiv:2309.14594*, 2023.

[91] 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)*, 2021.

[92] A. Meduri, M. Khadiv, and L. Righetti, "Deepq stepper: A framework for reactive dynamic walking on uneven terrain," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 2099–2105.

[93] J. Ahn, J. Lee, and L. Sentis, "Data-efficient and safe learning for humanoid locomotion aided by a dynamic balancing model," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4376–4383, 2020.

[94] M. Seo, S. Han, K. Sim, S. H. Bang, C. Gonzalez, L. Sentis, and Y. Zhu, "Deep imitation learning for humanoid loco-manipulation through human teleoperation," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2023.

[95] G. A. Castillo, B. Weng, S. Yang, W. Zhang, and A. Hereid, "Template model inspired task space learning for robust bipedal locomotion," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[96] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, 1st ed. USA: Cambridge University Press, 2019.

[97] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *SIGART Bull.*, vol. 2, no. 4, p. 160–163, jul 1991. [Online]. Available: https://doi.org/10.1145/122344.122377

[98] M. Janner, J. Fu, M. Zhang, and S. Levine, *When to Trust Your Model: Model-Based Policy Optimization.* Red Hook, NY, USA: Curran Associates Inc., 2019.

[99] N. Heess, G. Wayne, D. Silver, T. Lillicrap, Y. Tassa, and T. Erez, "Learning continuous control policies by stochastic value gradients," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 2944–2952.

[100] D. Hafner, T. P. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net, 2020.

[101] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," *arXiv preprint arXiv:2301.04104*, 2023.

[102] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[103] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2018, p. 7559–7566.

[104] T. Anthony, Z. Tian, and D. Barber, "Thinking fast and slow with deep learning and tree search," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5366–5376.

[105] 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, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017.

[106] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, "Model-based value estimation for efficient model-free reinforcement learning," *CoRR*, vol. abs/1803.00101, 2018.

[107] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 2455–2467.

[108] S. Racanière, T. Weber, D. P. Reichert, L. Buesing, A. Guez, D. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, R. Pascanu, P. Battaglia, D. Hassabis, D. Silver, and D. Wierstra, "Imagination-augmented agents for deep reinforcement learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5694–5705.

[109] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[110] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, vol. 12. MIT Press, 1999.

[111] I. Rechenberg, *Evolutions strategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Mit einem Nachwort von Manfred Eigen.* Frommann-Holzboog [Stuttgart-Bad Cannstatt], 1973.

[112] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[113] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[114] Z. Xie, H. Y. Ling, N. H. Kim, and M. van de Panne, "Allsteps: Curriculum-driven learning of stepping stone skills," in *Computer Graphics Forum*, vol. 39, no. 8. Wiley Online Library, 2020, pp. 213–224.

[115] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Proc. Conf. on Robot Learn.*, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds. PMLR, 2020, pp. 317–329.

[116] X. Da, R. Hartley, and J. W. Grizzle, "Supervised learning for stabilizing underactuated bipedal robot locomotion, with outdoor experiments on the wave field," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2017, pp. 3476–3483.

[117] T. Li, H. Geyer, C. G. Atkeson, and A. Rai, "Using deep reinforcement learning to learn high-level policies on the atrias biped," in *2019 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2019, pp. 263–269.

[118] F. Abdolhosseini, "Learning locomotion: symmetry and torque limit considerations," Ph.D. dissertation, University of British Columbia, 2019.

[119] G.-C. Kang and Y. Lee, "Finite state machine-based motion-free learning of biped walking," *IEEE Access*, vol. 9, pp. 20 662–20 672, 2021.

[120] Y. Hurmuzlu and D. B. Marghitu, "Rigid body collisions of planar kinematic chains with multiple contact points," *The International Journal of Robotics Research*, vol. 13, no. 1, pp. 82–92, 1994.

[121] O. Harib, A. Hereid, A. Agrawal, T. Gurriet, S. Finet, G. Boeris, A. Duburcq, M. E. Mungai, M. Masselin, A. D. Ames, K. Sreenath, and J. Grizzle, "Feedback control of an exoskeleton for paraplegics: toward robustly stable hands-free dynamic walking," *IEEE Control System Maganize*, vol. 38, no. 6, pp. 61–87, Dec. 2018.

[122] R. Chitnis and T. Lozano-Pérez, "Learning compact models for planning with exogenous processes," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 813–822. [Online]. Available: http://proceedings.mlr.press/v100/chitnis20a.html

[123] X. B. Peng and h. van de Panne, Michiel, "Learning locomotion skills using DeepRL: does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, ser. SCA '17. Los Angeles, California: Association for Computing Machinery, Jul. 2017, pp. 1–13.

[124] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Int. Conf. on Mach. Learn.* PMLR, 2014, pp. 387–395.

[125] M. Raibert, H. B. Brown, M. Chepponis, E. Hastings, S. Shreve, and F. C. Wimberly, "Dynamically stable legged locomotion," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-81-09, November 1981.

[126] K. Yin, K. Loken, and M. van de Panne, "Simbicon: Simple biped locomotion control," *ACM Trans. Graph.*, vol. 26, no. 3, p. 105–es, jul 2007. [Online]. Available: https://doi.org/10.1145/1276377.1276509

[127] X. Da, O. Harib, R. Hartley, B. Griffin, and J. W. Grizzle, "From 2D design of underactuated bipedal gaits to 3D implementation: Walking with speed tracking," *IEEE Access*, vol. 4, pp. 3469–3478, 2016.

[128] S. Rezazadeh, C. Hubicki, M. Jones, A. Peekema, J. Van Why, A. Abate, and J. Hurst, "Spring-mass walking with ATRIAS in 3D: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot," in *ASME 2015 Dynamic Systems and Control Conference.* American Society of Mechanical Engineers, 2015.

[129] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a Cassie bipedal robot: walking, standing, and riding a segway," *American Control Conference (ACC)*, 2019.

[130] 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*, Oct. 2012, pp. 5026–5033.

[131] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames, "Models, feedback control, and open problems of 3D bipedal robotic walking," *Automatica*, vol. 50, no. 8, pp. 1955–1988, 2014.

[132] Y. Gong and J. W. Grizzle, "Zero dynamics, pendulum models, and angular momentum in feedback control of bipedal locomotion," *Journal of Dynamic Systems, Measurement, and Control*, vol. 144, no. 12, 10 2022.

[133] G. Gibson, O. Dosunmu-Ogunbi, Y. Gong, and J. Grizzle, "Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[134] K. Green, Y. Godse, J. Dao, R. L. Hatton, A. Fern, and J. Hurst, "Learning spring mass locomotion: Guiding policies with a reduced-order model," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3926–3932, 2021.

[135] I. D. J. Rodriguez, N. Csomay-Shanklin, Y. Yue, and A. D. Ames, "Neural gaits: Learning bipedal locomotion via control barrier functions and zero dynamics policies," in *Learning for Dynamics and Control Conference.* PMLR, 2022, pp. 1060–1072.

[136] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. R. Westervelt, C. C. De-Wit, and J. W. Grizzle, "RABBIT: a testbed for advanced control theory," *IEEE Control Systems*, vol. 23, no. 5, Oct. 2003.

[137] J. Reher, C. Kann, and A. D. Ames, "An Inverse Dynamics Approach to Control Lyapunov Functions," *2020 American Control Conference (ACC)*, vol. 00, pp. 2444–2451, 2020.

[138] A. Del Prete, F. Nori, G. Metta, and L. Natale, "Prioritized motion–force control of constrained fully-actuated robots: "task space inverse dynamics"," *Robotics and Autonomous Systems*, vol. 63, pp. 150–157, 2015.

[139] B. C. Bennett, S. D. Russell, P. Sheth, and M. F. Abel, "Angular momentum of walking at different speeds," *Human Movement Science*, vol. 29, no. 1, pp. 114–124, Feb. 2010.

[140] H. Duan, A. Malik, J. Dao, A. Saxena, K. Green, J. Siekmann, A. Fern, and J. Hurst, "Sim-to-real learning of footstep-constrained bipedal dynamic walking," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 428–10 434.

[141] G. Bellegarda and K. Byl, "Training in task space to speed up and guide reinforcement learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2693–2699.

[142] A. D. Ames, "Human-inspired control of bipedal robots via control lyapunov functions and quadratic programs," in *Proceedings of the 16$^{th}$ international*

*conference on Hybrid systems: computation and control*, C. Belta and F. Ivancic, Eds., ACM. ACM, 2013, pp. 31–32.

[143] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3D-SLIP model," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5134–5140, Nov. 2013.

[144] X. Xiong, J. Reher, and A. D. Ames, "Global position control on underactuated bipedal robots: Step-to-step dynamics approximation for step planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2825–2831.

[145] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conf. Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014.

[146] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[147] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397–2404, 2023.

[148] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[149] J. Pratt, T. Koolen, T. de Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, "Capturability-based analysis and control of legged locomotion, part 2: application to m2v2, a lower-body humanoid," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1117–1133, Aug. 2012. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364912452762

[150] A. D. Prete, N. Mansard, O. E. Ramos, O. Stasse, and F. Nori, "Implementing torque control with high-ratio gear boxes and without joint-torque sensors," in *Int. Journal of Humanoid Robotics*, 2016, p. 1550044.

[151] D. Kim, S. J. Jorgensen, J. Lee, J. Ahn, J. Luo, and L. Sentis, "Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control," *The International Journal of*

*Robotics Research*, vol. 39, no. 8, pp. 936–956, 2020. [Online]. Available: https://doi.org/10.1177/0278364920918014

[152] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, “Blind bipedal stair traversal via sim-to-real reinforcement learning,” *arXiv preprint arXiv:2105.08328*, 2021.

[153] T. Li, Y. Zhang, C. Zhang, Q. Zhu, J. Sheng, W. Chi, C. Zhou, and L. Han, “Learning terrain-adaptive locomotion with agile behaviors by imitating animals,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 339–345.

[154] H. Shi, Q. Zhu, L. Han, W. Chi, T. Li, and M. Q. H. Meng, “Terrain-aware quadrupedal locomotion via reinforcement learning,” *arXiv preprint arXiv:2310.04675*, 2023.

[155] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, “Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control,” *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2908–2927, 2022.

[156] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning robust perceptive locomotion for quadrupedal robots in the wild,” *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abk2822

[157] S. Kareer, N. Yokoyama, D. Batra, S. Ha, and J. Truong, “Vinl: Visual navigation and locomotion over obstacles,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 2018–2024.

[158] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, “Legged locomotion in challenging terrains using egocentric vision,” in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 403–415. [Online]. Available: https://proceedings.mlr.press/v205/agarwal23a.html

[159] C. Zhang, N. Rudin, D. Hoeller, and M. Hutter, “Learning agile locomotion on risky terrains,” *arXiv preprint arXiv:2311.10484*, 2023.

[160] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, “Extreme parkour with legged robots,” *arXiv preprint arXiv:2309.14341*, 2023.

[161] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, “Anymal parkour: Learning agile navigation for quadrupedal robots,” *Science Robotics*, vol. 9, no. 88, 2024.

[162] X. B. Peng, G. Berseth, and M. van de Panne, "Dynamic terrain traversal skills using reinforcement learning," *ACM Trans. Graph.*, vol. 34, no. 4, jul 2015.

[163] X. B. Peng, G. Berseth, and M. Van de Panne, "Terrain-adaptive locomotion skills using deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016.

[164] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 36, 2017.

[165] B. van Marum, M. Sabatelli, and H. Kasaei, "Learning perceptive bipedal locomotion over irregular terrain," *arXiv preprint arXiv:2304.07236*, 2023.

[166] S. Kullback and R. A. Leibler, "On information and sufficiency."

[167] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[168] Q. Li, J. Zhang, D. Ghosh, A. Zhang, and S. Levine, "Accelerating exploration with unlabeled prior data," *arXiv preprint arXiv:2311.05067*, 2023.

[169] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 1110–1120.

[170] L. Smith, Y. Cao, and S. Levine, "Grow your limits: Continuous improvement with real-world rl for robotic locomotion," *arXiv preprint arXiv:2310.17634*, 2023.

[171] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen *et al.*, "Massively parallel methods for deep reinforcement learning," *arXiv preprint arXiv:1507.04296*, 2015.

[172] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

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

[174] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Data-driven latent space representation for robust bipedal locomotion learning," *arXiv preprint arXiv:2309.15740*, 2023.

[175] B. Weng, G. A. Castillo, Y.-S. Kang, and A. Hereid, "Towards standardized disturbance rejection testing of legged robot locomotion with linear impactor: A preliminary study, observations, and implications," *arXiv preprint arXiv:2308.14636*, 2023.

[176] B. Weng, G. A. Castillo, W. Zhang, and A. Hereid, "On the comparability and optimal aggressiveness of the adversarial scenario-based safety testing of robots," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 3299–3318, 2023.