Titre: Title:	Reinforcement Learning for Stair Climbing with Ascento: A Two-Wheeled Legged Robot
Auteur: Author:	Simon Chamorro
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Chamorro, S. (2024). Reinforcement Learning for Stair Climbing with Ascento: A Two-Wheeled Legged Robot [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/57991/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/57991/
Directeurs de recherche: Advisors:	Christopher J. Pal
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Reinforcement Learning for Stair Climbing with Ascento: A Two-Wheeled Legged Robot

SIMON CHAMORRO
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Mars 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Reinforcement Learning for Stair Climbing with Ascento: A Two-Wheeled Legged Robot

présenté par **Simon CHAMORRO**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Giovanni BELTRAME, président
Christopher J. PAL, membre et directeur de recherche
Glen BERSETH, membre

ACKNOWLEDGEMENTS

This thesis was conducted during my Master's at Polytechnique Montreal under the supervision of Prof. Christopher Pal. I would like to thank him for his guidance and supervision throughout my studies. I would also like to thank all my lab mates for their help and support, especially Florian Golemo and Martin Weiss.

Most of the work presented in this thesis was conducted at the Autonomous Systems Lab (ASL) in ETH Zurich, in partnership with Ascento Robotics. The content of this document was also used to submit a Master's thesis report at ETH during my research visit under the Mitacs Globalink program. Additionally, it was submitted to the IEEE International Conference on Robotics and Automation (ICRA 2024) and accepted for presentation.

I am very grateful to have had the opportunity to work on this exciting project. I would like to thank my supervisors at ETH and Ascento, Miguel de la Iglesia Valls and Victor Klemm, for their continuous support and guidance throughout the whole project, as well as Prof. Roland Siegwart for his supervision and valuable advice. I would also like to thank the rest of the Ascento team: Alessandro Morra, Ciro Salzmann, Dominik Mannhart, Giuseppe Rizzi, Sijmen Huizenga, and Jonas Enke, for their warm welcome and for hosting me for the duration of the project. It was a pleasure to be part of the team and I really appreciated all the help, especially with hardware tests.

Finally, I would like to acknowledge the support of this work and my Master's degree from the Quebec Research Funds (FRQNT) and the Institute for Data Valorization (IVADO).

RÉSUMÉ

Ascento Robotics, une entreprise de sécurité, utilise un robot à roues et à pattes pour effectuer des patrouilles de sécurité dans de vastes espaces extérieurs. Actuellement, Ascento est capable de naviguer sur un terrain plat et est robuste face à de légers obstacles. Cependant, une limitation critique du système actuel est l'incapacité du robot à monter des escaliers, ce qui limite sa fonctionnalité dans les environnements à plusieurs étages. Ce mémoire propose une nouvelle méthode pour surmonter cette limitation, en utilisant l'Apprentissage par Renforcement (RL) pour apprendre un contrôleur en simulation. L'approche proposée adopte une formulation de tâche RL basée sur la position, en contraste avec les contrôleurs existants basés sur la vitesse. Elle exploite la structure asymétrique acteur-critique pour utiliser des informations privilégiées provenant du simulateur pendant l'entraînement, tout en éliminant le besoin d'avoir accès à ces données pendant le déploiement. Cette approche simplifie considérablement les exigences opérationnelles du robot, car elle ne repose sur aucune information de capteur extéroceptif. Une nouvelle observation booléenne introduite dans le contrôleur sert de mode d'escalade d'escalier pouvant être activé ou désactivé. Cette recherche présente également les résultats du processus de transfert de la simulation au monde réel, en utilisant des techniques telles que la randomisation de domaine, ainsi qu'une généralisation de la méthode présentée et son application à d'autres robots. Le contrôleur résultant, qui ne nécessite pas d'informations de capteur ni de système de positionnement externe, permet au robot de monter avec succès des marches de 15 cm, une tâche qui était auparavant impossible pour le robot Ascento. Cette avancée élargit le champs opérationnel du robot, améliorant son potentiel dans les applications de sécurité.

Mots-clés: Robotique, Locomotion à pattes, Robots à roues et à pattes, Apprentissage par renforcement, Apprentissage automatique.

ABSTRACT

Ascento Robotics, a security company, utilizes a wheeled-legged robot to do security patrols in large outdoor areas. Ascento currently has the capacity to navigate flat terrain and is robust to minor obstacles. However, one critical limitation of the current system is the robot's inability to climb stairs, limiting its functionality in multi-story settings. This thesis proposes a novel method to overcome this limitation, employing Reinforcement Learning (RL) to train a controller in simulation. The proposed approach adopts a position-based RL task formulation, contrasting with the existing velocity-based control. It leverages the asymmetric actor-critic structure to utilize privileged information from the simulator during training while eliminating the need for this data during deployment. This approach significantly simplifies the operational requirements of the robot, since it does not rely on any exteroceptive sensor information. A new boolean observation introduced to the controller serves as a stair-climbing mode that can be activated or deactivated. This research also presents findings from the sim-to-real transfer process, using techniques such as domain randomization. The resultant controller, which does not require sensor information or a positioning system, successfully enables the robot to climb 15 cm steps, a task that was previously impossible for Ascento. This advancement broadens the operational terrain for the robot, enhancing its potential in security applications.

Keywords: Robotics, Legged locomotion, Wheeled-legged robots, Reinforcement Learning, Machine Learning.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

RÉSUMÉ iv

ABSTRACT v

TABLE OF CONTENTS vi

LIST OF TABLES viii

LIST OF FIGURES x

LIST OF SYMBOLS AND ACRONYMS xii

LIST OF APPENDICES xiv

CHAPTER 1 INTRODUCTION 1
1.1 Contributions 2
1.2 Thesis Structure 2

CHAPTER 2 LITERATURE REVIEW 4
2.1 Overview of RL for Robotics Control 4
2.2 Legged Locomotion using RL 5
2.3 Hybrid Wheeled-Legged Locomotion 7
2.4 Work on Balancing Robot Control 8
2.5 Ascento Robot 8
2.6 Sim-to-Real Transfer 10

CHAPTER 3 METHODOLOGY 12
3.1 Simulation Environment 13
3.1.1 Terrains 14
3.1.2 Robot Model 15
3.2 Task Formulation 15
3.2.1 State 16
3.2.2 Actions 18
3.2.3 Rewards 18

3.3 Curriculum 20
3.4 Learning Algorithm 21
3.5 Sim-to-real Transfer 22
3.6 Deployment 23

CHAPTER 4 EXPERIMENTS & RESULTS 25
4.1 Simulation Results 25
4.1.1 Position Control Analysis 25
4.1.2 Terrain Boolean Importance 26
4.2 Real World Results 27
4.3 Sim-to-real Transfer 28

CHAPTER 5 GENERALIZATION OF THE METHOD 29
5.1 Robots & Method adjustments 30
5.2 Results & Discussion 30

CHAPTER 6 CONCLUSION 33
6.1 Limitations & Future Work 33

REFERENCES 35

APPENDICES 40

LIST OF TABLES

Table 3.1 **Observation:** This table presents the observation given to our agent. It also presents the normalizing coefficient used as well as each input dimension and percentage of noise applied during training. 17

Table 3.2 **Privileged information:** This table presents the privileged information given to the critic during training. This information is only available in simulation. We present the normalizing coefficient applied to each input as well as their dimension. There is no noise applied to this data, as it will not be used during deployment. 17

Table 3.3 **Rewards:** This table presents all the reward terms used for training our stair-climbing agent with the Ascento robot as well as their respective coefficients. Rewards #1 to #3 are task-specific, while rewards #4 to #13 are used to obtain a smooth motion and penalize undesired behaviors. 19

Table 3.4 **Domain randomization:** This table presents the physical parameters that were randomized during training to facilitate sim-to-real transfer. 22

Table 5.1 **Simulation results for step climbing:** This table presents the results of our method on different robots. For each robot, we test our method with the boolean observation set to 1 and 0, as well as two ablations. We Each experiment is conducted for 2000 episodes. 32

Table A.1 **PPO learning parameters:** Training hyperparameters for the PPO algorithm. 40

Table B.1 **Unitree Go1 shaping rewards:** Similar shaping rewards to those presented in table 3.3. The only new term introduced is reward #10, encouraging feet air time to learn a policy that does not drag its feet too close to the ground. 41

Table B.2 **Cassie shaping rewards:** Similar shaping rewards to those presented in tables 3.3 and B.1. Additionally, reward #7 penalizes motor torques, and reward #9 penalizes lifting both feet off the ground for the bipedal robot Cassie to maintain good balance. 42

Table B.3 **ANYmal on Wheels (quadruped) shaping rewards:** Combination of shaping rewards from tables 3.3, B.1, and B.2 used to learn a policy from the ANYmal on Wheels in quadruped mode. 42

Table B.4 **ANYmal on Wheels (biped) shaping rewards:** Combination of shaping rewards from tables 3.3, B.1, and B.2 used to learn a policy from the ANYmal on Wheels in biped mode. 43

LIST OF FIGURES

Figure 1.1 **Ascento Robot:** Latest version of the Ascento robot patrolling an outdoor site. 1

Figure 2.1 **Ascento Robot evolution:** The previous version (on the left) had a linkage system for each leg. The new version has pseudo-direct drive motors at the hips and knees, introducing additional degrees of freedom and allowing for more complex maneuvers. 9

Figure 2.2 **Stair climbing attempt with current velocity controller:** Images of the Ascento robot attempting to climb a 15cm step with a constant velocity of 1m/s (left) and its corresponding linear velocity profile during the attempt (right). 10

Figure 3.1 **System overview during training:** We present an overview of the complete training setup. An agent interacts with the simulated environment and is trained using an asymmetric version of the PPO algorithm. The observation and privileged information are retrieved from the Isaac Gym simulator and provided to the agent, which in turn outputs an action. 12

Figure 3.2 **Different training terrains.** Rough terrains used during training in addition to the stairs environments. From left to right: smooth pyramid, rough pyramid, discrete obstacles, smooth slope. 13

Figure 3.3 **Stairs terrains:** Single step terrain (left), and staircase terrain (right). 15

Figure 3.4 **Task setup:** Example for one of the 4096 simulated robots during training. The robot spawns in the environment and a goal pose is sampled. The goal pose is marked in green and red. The yellow dots represent the terrain height measurements which are part of the privileged information about the terrain. 16

Figure 3.5 **Training curriculum:** Different training terrains, progressively harder from left to right. The stairs terrains, where the terrain boolean is set to 1 during training, are delimited in green. In the other terrains, this observation is set to 0. 20

Figure 3.6 **Stair climbing motion in Gazebo:** This figure presents the robot going up a step in the simulator Gazebo. We use this simulator as a sim-to-sim validation tool before deploying policies on the real robot. 23

Figure 3.7 **System overview during deployment:** We present an overview of the complete deployment setup. The trained agent interacts with the real environment and is composed only of the actor-network. The observation is retrieved directly from proprioceptive sensors on the robot and provided to the agent, which in turn outputs an action. 24

Figure 4.1 **Step motion:** This figure presents an overlaid illustration of the Ascento robot at different stages of the step-up motion. We also present its velocity profile. 26

Figure 4.2 **Real world results:** Ascento robot going up a 15cm step with our learned policy. Images are taken at a 0.25-second interval. 28

Figure 5.1 **Different robots:** This figure presents the robots used for evaluating the generalization capabilities of our method. Top: Cassie, Ascento Robot, ANYmal on Wheels (Biped). Bottom: ANYmal on Wheels (Quadruped), Unitree Go1. 29

LIST OF SYMBOLS AND ACRONYMS

π	Policy
ϕ	Policy parameters
T	Episode length
γ	Discount factor
r	Reward
J	Expected cumulative reward
s	State
a	Action
Q	Action-value function
L	Loss
ϵ	Clip parameter
θ	Heading
$\vec{\dot{\theta}}$	Angular velocity (vector)
$\vec{\gamma}$	Projected gravity (vector)
$\vec{g}$	Goal position (vector)
b	Terrain boolean
$\vec{q}$	Joint positions (vector)
$\vec{\dot{q}}$	Joint velocities (vector)
$\vec{v}$	Linear velocity (vector)
$H_{terrain}$	Terrain height measurements (matrix)
$\vec{f}$	Force (vector)
$\vec{\tau}$	Torque (vector)
γ	Discount factor
μ	Friction coefficient
c	Contact (boolean)
$\|\|\|$	Euclidean norm operator

ADP	Adaptive Dynamic Programming
ASL	Autonomous Systems Lab
CAD	Computer-Aided Design
CoM	Center of Mass
FRQNT	Quebec Research Funds
GPS	Global Positioning System
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit
IVADO	Institute for Data Valorization
MPC	Model Predictive Control
PID	Proportional – Integral – Derivative
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
RMA	Rapid Motor Adaptation
ROS	Robot Operating System
RSL	Robotics Systems Lab
SLAM	Simultaneous Localization and Mapping
TRPO	Trust Region Policy Optimization
W-SLIP	Wheeled-Spring-Loaded Inverted Pendulum

LIST OF APPENDICES

Appendix A Learning Algorithm Parameters 40
Appendix B Reward Shaping Rewards for Different Robots 41

CHAPTER 1 INTRODUCTION

Mobile ground robots have been widely studied and used for various applications, such as delivery, inspection, manufacturing, and exploration [1]. One such application that stands out is in the realm of security, where robotics is in the process of redefining traditional practices. In this sector, Ascento Robotics has made a notable contribution through the development of a wheeled-legged security robot [2]. Traditionally, there are two main types of mobile robots: wheeled robots and legged systems. Wheeled robots are efficient at traveling long distances, but lack the agility to traverse obstacles such as stairs, while legged systems are more agile, but have limited battery life and are harder to control [3]. By leveraging the strengths of both wheeled and legged systems, Ascento has the potential to perform complex tasks in real-world environments. Figure 1.1 shows the Ascento robot. It has two legs, with three degrees of freedom each. We refer to the joints as the hip, knee, and wheel for each one of the legs. This robot, equipped with the ability to navigate flat terrain and small obstacles, has been significantly useful in a wide array of security scenarios. Despite these notable strengths, the existing system presents a challenge when deployed in multi-level environments. Specifically, the robot's inability to climb stairs is a key limitation that reduces its effectiveness in these settings. This constraint emphasizes the need for innovative approaches that can overcome this limitation and expand the robot's operational versatility.

Figure 1.1 **Ascento Robot:** Latest version of the Ascento robot patrolling an outdoor site.

With this in mind, our research aims to address this gap by proposing a controller method that employs Reinforcement Learning (RL) to equip the robot with stair-climbing capabilities. Our approach represents a shift from the current velocity-based control to a position-based task formulation. The introduction of an asymmetric actor-critic algorithm is another key aspect of our methodology. This approach enables the efficient use of privileged information from the simulator during training. Additionally, our study introduces a new boolean observation to the controller. This innovative feature serves as a mode switch for stair-climbing, thereby allowing a good performance on both regular terrain and stair ascent with the same control policy. We also present the findings from our efforts to bridge the sim-to-real gap, incorporating techniques such as domain randomization as well as some preliminary results regarding parameter estimation. The end result is a more versatile controller that operates without the need for exteroceptive sensor information or a positioning system, enabling the robot to climb stairs. This advancement not only overcomes a key limitation of the existing system but also significantly expands the operational scope of the robot.

1.1 Contributions

Our main contributions are the following:

1. We present an RL task formulation to train policies capable of climbing stairs with a wheeled-legged balancing robot.
2. Our proposed system can successfully climb 22cm steps in simulation and 15cm steps in the real world, previously impossible with the Ascento robot.
3. Our method does not require any perceptive data or a positioning system such as Simultaneous Localization and Mapping (SLAM) or Global Positioning System (GPS), making it easily integratable into a standard control stack.
4. We study how our method generalizes to other robot morphologies such as the Unitree Go1 quadruped robot, the bipedal robot Cassie, and the wheeled-legged robot ANYmal on Wheels.

1.2 Thesis Structure

The subsequent chapters of this thesis are structured as follows:

- **Chapter 2** contains an overview of the relevant literature and explains how it relates to this work.

- **Chapter 3** presents our proposed methodology in detail, including how it works both during training and deployment.
- **Chapter 4** shows our experiments and results, both in simulation and in the real world.
- **Chapter 5** presents our generalization results using other robot morphologies.
- **Chapter 6** concludes this thesis, summarizing our contributions and giving an outlook on future work directions.

CHAPTER 2 LITERATURE REVIEW

Hybrid wheeled-legged systems have been developed to harness the advantages of both technologies, but these systems pose significant challenges for control due to their complex dynamics [4]. Recently, RL has emerged as a promising technique for developing control policies for these complex robotic systems. In particular, interesting results have been achieved using RL to develop controllers for wheeled-legged quadrupeds [5, 6]. In this chapter, we first give an overview of RL for robotics control, then cover previous work on legged locomotion, hybrid wheeled-legged locomotion, and sim-to-real transfer.

2.1 Overview of RL for Robotics Control

RL is a widely used machine learning framework that has shown promising results in solving complex robotic control tasks [7]. In this framework, an agent learns to perform a task by interacting with an environment and receiving feedback in the form of rewards or penalties. The agent aims to learn a policy, which is a mapping from states to actions, that maximizes the cumulative reward over time. Robotic applications are a natural fit for RL algorithms since they often involve complex tasks that are difficult to program explicitly. RL has been applied to a wide range of robotic tasks, such as locomotion, manipulation, and navigation [8–10]. Mathematically, the goal is to find a policy π that maximizes the expected cumulative reward $J(\pi)$ given by:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \right], \tag{2.1}$$

where $\mathbb{E}_\pi$ denotes the expected value under policy π, γ is a discount factor that prioritizes immediate rewards over future rewards, r_t is the reward received at time step t, and T is the final time step of an episode. One type of RL algorithm that has gained significant attention in recent years is policy gradient methods [11]. These methods directly optimize the policy parameters by estimating their gradient, and they have been shown to be effective in high-dimensional and continuous action spaces. The goal is to update the policy parameters ϕ to increase the expected cumulative reward $J(\pi)$, which can be achieved by performing stochastic gradient ascent on the objective function:

$$\nabla_\phi J(\pi) = \mathbb{E}_\pi \left[\nabla_\phi \log \pi(a_t|s_t) Q_\pi(s_t, a_t) \right], \tag{2.2}$$

where $\nabla_\phi \log \pi(a_t|s_t)$ is the log-derivative of the policy π with respect to its parameters ϕ, and $Q_\pi(s_t, a_t)$ is the action-value function, which gives the expected cumulative reward starting from state s_t, taking action a_t, and following policy π thereafter. One popular policy gradient algorithm is Proximal Policy Optimization (PPO), which has been shown to achieve state-of-the-art performance on a variety of tasks, including robotic locomotion [12]. It uses a clipped surrogate objective to prevent large policy updates. The clipped surrogate objective is given by:

$$L_{\text{clip}}(\phi) = \mathbb{E}_t \left[\min(r_t(\phi)A_t, \text{clip}(r_t(\phi), 1-\epsilon, 1+\epsilon)A_t) \right], \tag{2.3}$$

where $r_t(\phi) = \frac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{\text{old}}}(a_t|s_t)}$ is the likelihood ratio between the new policy and the old policy, A_t is the advantage function, which measures the advantage of taking action a_t in state s_t compared to following the current policy, and ϵ is a hyperparameter that controls the extent of the clipping.

In summary, policy gradient methods update the policy parameters using stochastic gradient ascent on the objective function, and PPO uses a clipped surrogate objective to prevent large policy updates. These methods have shown great promise in developing control policies for robotic locomotion tasks.

2.2 Legged Locomotion using RL

Traditional control methods, such as Model Predictive Control (MPC) and trajectory optimization were previously used for legged locomotion problems. However, these methods require a detailed dynamic model of the robot, which is never perfectly accurate. Furthermore, they are complex for high dimensional systems, such as quadrupeds, and can't take into account sensor data. Recently, RL has emerged as a promising alternative to these traditional methods for controlling legged systems.

Multiple recent works have successfully tackled locomotion for the ANYmal robot [13] using RL. In 2019, Hwangbo et al. [14] proposed an RL control policy using Trust Region Policy Optimization (TRPO) [15], and an actuator network to model the motors more accurately in simulation. The policy is goal-conditioned with the desired linear and angular velocities of the base and outputs position targets for the joints. The actuator network uses this as an input plus the joint state history to compute torques. The policy is trained in RaiSim [16], and the actuator network is trained using supervised learning with data from the real robot.

Another work, presented by Lee et al. [17], builds on this method and proposes a teacher-student architecture to include exteroceptive information about the environment during the training procedure. A teacher policy is trained using RL and privileged information from the simulator to output joint position targets. The student policy is then trained via imitation learning from the teacher policy and uses only proprioceptive data as input. Finally, in 2022, Miki et al. [18] proposed a controller that uses proprioceptive and exteroceptive information to navigate rough terrain more efficiently. A key component of this method is the belief encoder learned by the student network. It allows the agent to reconstruct a latent representation of the state of its surroundings only from its own state history and sparse sensor readings. Overall, the result of these works is a very robust controller capable of navigating rough terrain reliably. However, one downside of these methods is the training time needed to learn a policy. Recently, Rudin et al. [19] proposed a new environment capable of simulating thousands of robots in parallel using a single Graphics Processing Unit (GPU). Using this framework, and a standard PPO implementation to learn a locomotion policy, they achieve stable results that transfer to the real world in under 20 minutes of training.

Furthermore, others have been able to use RL to allow robots to achieve complex maneuvers that require more dynamic behaviors than regular locomotion. For instance, Fu et al. [20] presented a unified policy for whole-body control of a legged quadruped and a manipulator using RL, addressing the limitations of standard modular control pipelines. Their method allows the legged manipulator system to achieve dynamic and agile behaviors across various tasks such as opening doors and picking up objects by coordinating the control of the base and the arm. Additionally, Rudin et al. [21] proposed an end-to-end policy for local navigation in challenging environments such as steep stairs and large gaps in terrain, overcoming the limitations of traditional path planning and locomotion control. The main contribution is the task formulation they propose, which gives the agent a position goal to reach by the end of the episode instead of a continuous velocity profile to track. This allows for a more flexible path and locomotion gait selection. The approach enables robots to learn complex behaviors such as jumping, resulting in improved energy efficiency and higher success rates. This work is very relevant as we use a similar formulation for our position-based task. However, we have constructed our observations differently such that the robot does not need any perceptive information at deployment.

Moreover, recent works have leveraged RL to develop agile controllers capable of performing dynamic tasks while interacting with their environment. For example, DribbleBot [22] demonstrates a legged robotic system capable of dribbling a soccer ball under real-world

conditions, using RL in simulation and including image observations from a camera into the control loop. In another study, quadruped robots are trained to perform tasks such as wall climbing, button pressing, and object interaction using a combination of locomotion and manipulation skills. A high-level behavior tree is employed to create a robust, long-term plan that can be executed in both simulation and real-world scenarios, and the low-level policy is learned with RL [23]. Additionally, Ji et al. [24] proposed a hierarchical framework to enable quadrupedal robots to perform precise soccer shooting skills using RL. This framework trains a robust motion control policy alongside a planning policy to decide the desired kicking motion, resulting in the successful deployment of a robot capable of accurately shooting a soccer ball to random targets in real-world conditions. Finally, recent works have also proposed methods to control bipedal robots using both RL and traditional methods, allowing them to stand, walk, and even ride a segway [25, 26].

In this work, we study the problem of stair-climbing. This is particularly difficult for bipeds since it requires a very dynamic movement where the robot keeps balance on only one point of contact while stepping up. We propose a method to climb stairs by developing step reflexes using privileged terrain information in an asymmetric actor-critic setup [27]. Interesting reflex behaviors have been achieved on quadrupeds, presented by Lee et al. [17]. Siekmann et al. have also shown that it is possible to blindly traverse stairs with the robot Cassie [28].

2.3 Hybrid Wheeled-Legged Locomotion

Hybrid wheeled-legged robots aim to combine the agility benefits of legged robots with the efficiency of wheeled robots. Wheeled ANYmal, for instance, is a wheeled quadruped robot developed by researches from ETH and one of the first of its kind. There has been a lot of work in the last few years to develop control algorithms for this robot using MPC and other methods such as trajectory optimization [29–31]. However, these methods are very challenging to develop due to the complex nature of the highly nonlinear system dynamics at play. RL has shown very promising results as an alternative method. In 2022, Lee et al. [5] proposed a model-free RL approach trained in simulation to learn a velocity-based controller. The controller receives a target velocity for the robot base and outputs commands for the joints and wheels. This work shows promising results, with the policy effectively learning to dynamically transition between driving and walking modes based on user command and terrain conditions. Furthermore, Vollenweider et al. [6] proposed a framework to learn complex motions such as standing up on two wheels and sitting back down using motion priors and RL. They also demonstrate the ability of the robot to balance on two wheels and follow

velocity commands while maintaining this configuration.

2.4 Work on Balancing Robot Control

In 2021, Chen et al. [32] proposed a planning and control system for a jumping wheeled-legged balancing robot. They introduce a new model called the Wheeled-Spring-Loaded Inverted Pendulum (W-SLIP) to facilitate planning using quadratic programming. Furthermore, the system employs a Linear Quadratic Regulator (LQR) based wheel controller and a task-space whole-body controller for the other joints. They demonstrate achieving feasible jumping trajectories in simulation but do not conduct any real-world experiments on a physical robot. Other work has investigated the control problem of a wheel-legged balancing robot specifically in scenarios where an accurate dynamic model is absent [33]. Their proposed strategy leverages RL and adaptive dynamic programming (ADP) to derive a learning-based solution for adaptive optimal control. They demonstrate that suboptimal controllers can be learned directly from input-state data collected along the robot's trajectories. However, this work is limited to balancing in place and does not tackle locomotion. Finally, Zhu et al. [34] proposed an RL-based hierarchical control framework for path tracking of a wheeled bipedal robot. The system is structured into three levels of control: A high-level RL policy, a middle-level Lyapunov-based non-linear controller used to track a desired line, and a low-level Proportional – Integral – Derivative (PID) controller for balancing and velocity tracking. This method shows good results for locomotion on flat terrain with no obstacles or other disturbances.

2.5 Ascento Robot

The Ascento robot is a two-wheeled and legged balancing robot, which was first introduced in 2019 by Klemm et al. [2]. In 2020, a whole-body controller approach was presented to control the robot. The work incorporates the modeling of the linkage mechanisms used for the two legs and proposes a trajectory optimization method to find feasible non-smooth trajectories for tasks such as jumping up steps. An LQR-based controller is proposed for trajectory tracking. The work includes real-world experiments, demonstrating the robot's ability to traverse steps, drive up ramps, jump, and stand up. While this work shows impressive results, the method needs perfect information about the terrain before computing the trajectories. For this reason, it is not a suitable method for deployment outside of the lab without human guidance. Furthermore, since the first iteration of Ascento, a few changes have been made. Figure 2.1 illustrates these differences, the most notable being that the linkage system of the legs was replaced by direct drive joints at the hips and knees. This design change adds

degrees of freedom to the system, unlocking new possibilities, such as climbing stairs with different motions. For example, with the previous system, it was not possible to move the legs unsymmetrically without inducing a yaw motion. With the new robot, it would be possible to perform a step motion, with one leg going up an obstacle before the other, while maintaining a stable base orientation.

Figure 2.1 **Ascento Robot evolution:** The previous version (on the left) had a linkage system for each leg. The new version has pseudo-direct drive motors at the hips and knees, introducing additional degrees of freedom and allowing for more complex maneuvers.

More recently, a method for standing up from different ground configurations has been presented for Ascento by Salzmann [35]. In this work, an agent is trained using RL to directly output motor commands from a proprioceptive input to perform a stand-up maneuver. The method also works to develop a velocity-based controller for the robot. The controller has demonstrated robust results in slightly rough terrain and small obstacles but is incapable of traversing bigger steps such as stairs. In fact, this method is trained to follow base linear and angular velocity commands. This is not optimal for a task such as climbing a step, since the optimal velocity profile for this is not known a priori. Figure 2.2 shows a picture of the Ascento robot attempting to climb a step using the aforementioned controller as well as its velocity profile. A video of this experiment is also presented in our results video [36] at 0:05. We can see that the controller tries to track a commanded velocity of $1m/s$, but fails to overcome the $15cm$ obstacle. We will present how we propose to overcome this problem in this thesis.

Figure 2.2 **Stair climbing attempt with current velocity controller:** Images of the Ascento robot attempting to climb a 15cm step with a constant velocity of 1m/s (left) and its corresponding linear velocity profile during the attempt (right).

2.6 Sim-to-Real Transfer

In this section, we discuss different techniques used in the literature to successfully transfer RL policies trained in simulation to the real world.

Domain Randomization: Domain randomization is a technique used in RL and robotics to make policies more robust [37]. The idea is to expose the model to a wide variety of environments during training so that it becomes more robust to changes in the environment during deployment. The concept behind domain randomization is that we know that the simulator used for training does not perfectly model the real world. Therefore, generating a wide range of possible scenarios during the training process is a way to make it more robust to these discrepancies. The scenario variations include randomization in whatever environmental factors are important for the specific task. For example, in the case of a perception task, one could vary object sizes, textures, lighting conditions, and more. In our case, we are dealing with locomotion, so we focus on physical parameters such as masses, forces, friction coefficients, etc. By doing this, the model can learn to generalize from its training and adapt to new, unseen conditions. This is particularly useful in robotics, where the robot must operate in the real world and handle a wide range of scenarios that might not have been explicitly included in the training data. It would also be very impractical to train in the real world since RL methods are very data expensive.

Parameter Encoding Techniques: Kumar et al. [38] proposed Rapid Motor Adaptation (RMA), a method to overcome sim-to-real challenges specifically for quadruped robots. In this work, they use an adaptation module in addition to the learned locomotion policy in simulation. The goal of the adaptation module is to infer the physical parameters of the

real-world robot, or a latent encoding representing them, from a history of proprioceptive information at runtime. The inferred parameters are fed to the policy so it can adapt its behavior. This method allows for more aggressive domain randomization since the goal is not for the policy to be robust to all variations with the same behavior, but to learn to modify its behavior depending on the changes. They demonstrate that this method allows the policy to adapt to situations such as changes in friction coefficients of the terrain or changes in payload or robot mass. This technique has shown promising results and has been subsequently used for legged manipulator systems, among others [20].

Actuator Modeling: One approach that has been particularly useful for work on the ANYmal quadruped robot is actuator modeling [14]. This method tackles specifically the discrepancies between the actuator response for a given command in simulation versus the real world. The proposed method to overcome this problem is to collect real-world data from robot trajectories and use supervised learning to train an actuator network to estimate torque at the joints given a history of position errors and velocities. This data can be collected using any existing controller for the robot, it is then used during RL training to mimic the behavior of the actuators. This could be considered to be a real-to-sim technique, which aims to make the simulation training more realistic to improve performance of the learned controller once deployed in a real environment.

CHAPTER 3 METHODOLOGY

In this chapter, we present the methodology used to develop a controller for stair-climbing with the Ascento robot. Figure 3.1 presents an overview of the proposed architecture while figure 3.7 presents how the system works during deployment, which is explained further in section 3.6. At a high level, we use a standard RL formulation, where an agent, or algorithm, interacts with an environment. At every timestep, the environment provides the state of the robot to the agent and the agent outputs an action. The action is applied to the environment, which then produces the next state, and so on. In the following sections, we will present each of these components in detail. We will present the simulation environment used for training, the RL task formulation, the training procedure, the learning algorithm, our sim-to-real transfer efforts, as well as deployment details.

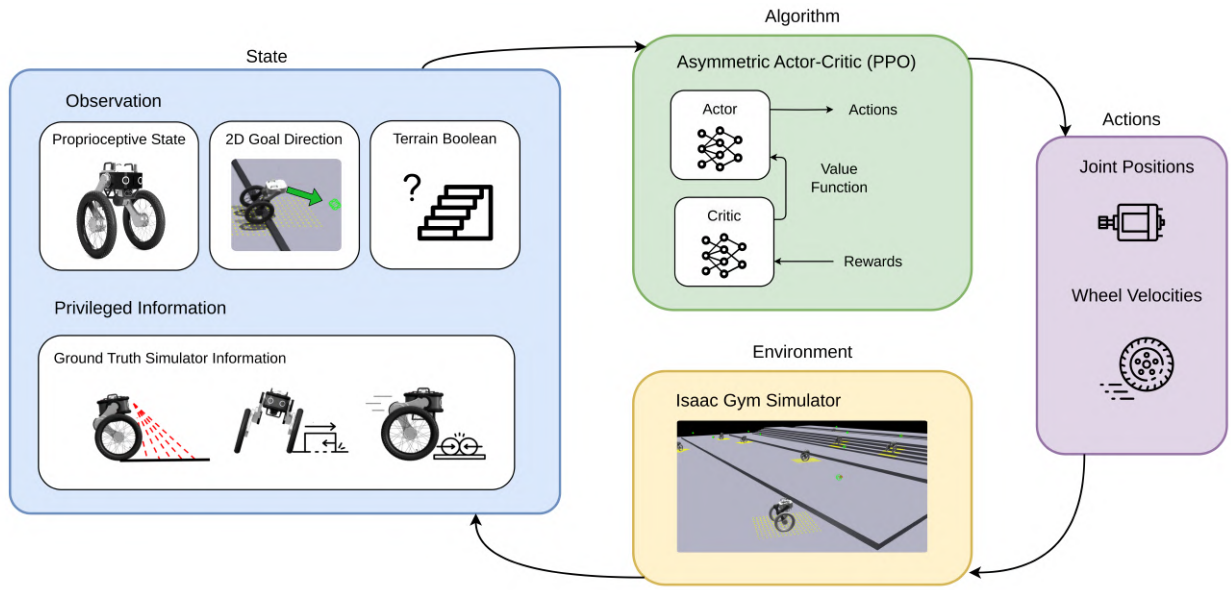

Figure 3.1 **System overview during training:** We present an overview of the complete training setup. An agent interacts with the simulated environment and is trained using an asymmetric version of the PPO algorithm. The observation and privileged information are retrieved from the Isaac Gym simulator and provided to the agent, which in turn outputs an action.

3.1 Simulation Environment

For this thesis, we used Isaac Gym [39], a physics simulation environment developed by NVIDIA, as the training platform for our RL-based controller. Isaac Gym is specifically designed for RL applications in robotics and other domains, offering unparalleled training speed and scalability by fully leveraging GPU resources. Isaac Gym is built on top of the NVIDIA PhysX engine, and one of its main advantages is its seamless integration with popular deep learning frameworks such as PyTorch, which is the framework we use, facilitating rapid development and deployment of RL agents. Additionally, the simulator is highly customizable, allowing for the creation of diverse environments and tasks, as well as importing custom robot models. Isaac Gym supports domain randomization techniques, which can enhance the robustness of RL agents by introducing variations in the environment during training, thus aiding in the transfer of learned policies from simulation to the real world. This will be further discussed in section 3.5.

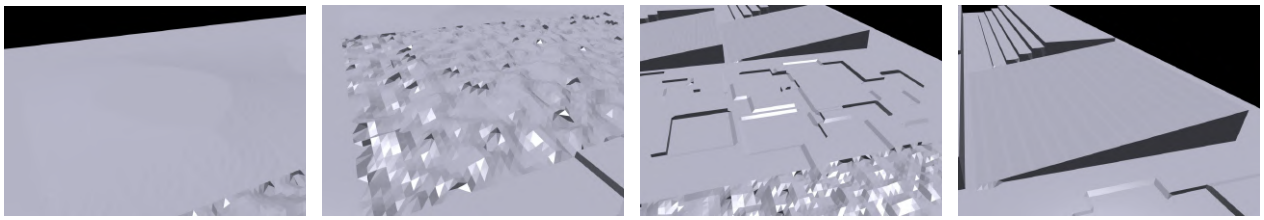

Figure 3.2 **Different training terrains.** Rough terrains used during training in addition to the stairs environments. From left to right: smooth pyramid, rough pyramid, discrete obstacles, smooth slope.

Compared to other widely-used simulators such as Raisim [16], MuJoCo [40], and Gazebo [41], Isaac Gym's GPU-accelerated architecture enables significantly faster simulations and training of agents, thanks to its high parallelization. We were able to simulate 4096 robots in parallel in a single GPU. While Raisim and MuJoCo offer accurate physics simulations, they can be more computationally demanding and do not scale as efficiently as Isaac Gym in parallel environments. Gazebo, on the other hand, provides a comprehensive simulation framework with support for various sensors and plugins but typically suffers from slower simulation speeds. In fact, we used Gazebo for sim-to-sim validation as a stepping stone toward real-world deployment. As the real robot software stack runs under the Robot Operating System (ROS), we were able to test the complete pipeline in simulation using Gazebo and validate performance.

3.1.1 Terrains

To construct our training environment, we use multiple types of different terrain. The environment is 60 meters wide by 120 meters long and is divided into 10m $\times$ 10m terrains. There are six columns for six types of terrain and twelve rows for increasing curriculum difficulty. We have noted that the policies are more robust when exposed to a bigger variety of obstacles. These are all the types of terrain that we use:

- **Stairs:** Since the main goal is to learn a controller capable of climbing stairs, two out of the six columns are composed of stairs terrains. The stairs columns are divided into two sections. The first half, i.e. six rows, are terrains with only one step of gradually increasing height. The height ranges from 5cm to 17cm. There is some space before the step and some space after. These terrains are meant for the agent to learn the basic motion of going up a step before tackling continuous stairs terrains. The second half of the rows contain normal stairs terrains. These have stairs flights of five steps. Step height increases gradually from 12cm to 20cm, while step width decreases gradually from 1m to 35cm. Figure 3.3 shows an example of a single-step terrain and a staircase terrain.

- **Smooth slopes:** Smooth slopes are simply slanted planes. These environments are important to reduce drift. In fact, since we use a balancing robot, training it to reach a precise goal on a slope forces it to maintain balance while holding a stable position. The slopes range from 0 degrees to 8.5 degrees.

- **Discrete obstacles:** The discrete obstacle are random rectangular obstacles that are created on flat terrain. Their height goes up to 10cm and their dimensions in length and width vary between 1m and 2m. This environment simulated small obstacles that the robot may encounter in the real world such as debris.

- **Smooth & rough pyramid slopes:** These environments are pyramids with a flat platform in the middle, which is where the robot spawns at the beginning of an episode. Their inclination goes from 0 to 8.5 degrees just like the slopes. In the case of the rough pyramids, they have randomized noise in the terrain height of up to 5cm, which gives them a bumpy surface. This helps the robot learn to remain stable in rough terrains such as gravel or grass.

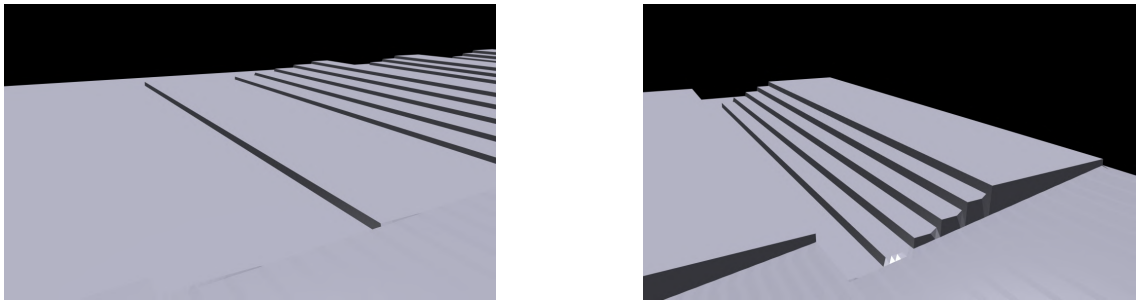

Figure 3.3 **Stairs terrains:** Single step terrain (left), and staircase terrain (right).

3.1.2 Robot Model

In order to train the RL controller in Isaac Gym, we need a model of the robot to simulate its behavior. In this work, we use the same model as presented by Salzmann [35] as we use the same robot. In summary, the model was obtained from the computer-aided design (CAD) of the robot. The contact geometries for interaction with the environment were simplified to reduce simulation time and memory usage. Joint limitations, such as velocity and torque limits, were modeled to match real-world operation ranges. The dimensions, inertia, masses, and motor limitations were based on the CAD model and verified on the real system after its assembly.

3.2 Task Formulation

We formulate our RL task as a position-based navigation task. The robot spawns in the environment, and a goal position is sampled. The episodes last 6 seconds, and to solve the task, the robot's base needs to be at the goal location before the end of the episode. For the step and stairs terrains, the robot always spawns at the bottom of the steps roughly facing them ($\pm 0.1 rad$). The goal is sampled at the top and also has a heading within the same range. For the other terrains, the robot spawns in the middle of the terrain and a goal is sampled somewhere within a 1m radius of the robot with a random orientation. Figure 3.4 illustrates the setting of our RL task in a step environment. Our position-based task formulation is inspired by the work of Rudin et al. [21], however, the observation we provide to our RL agent differs in order to avoid the need for exteroceptive information during deployment. This difference will be explained in more detail in the following sections.

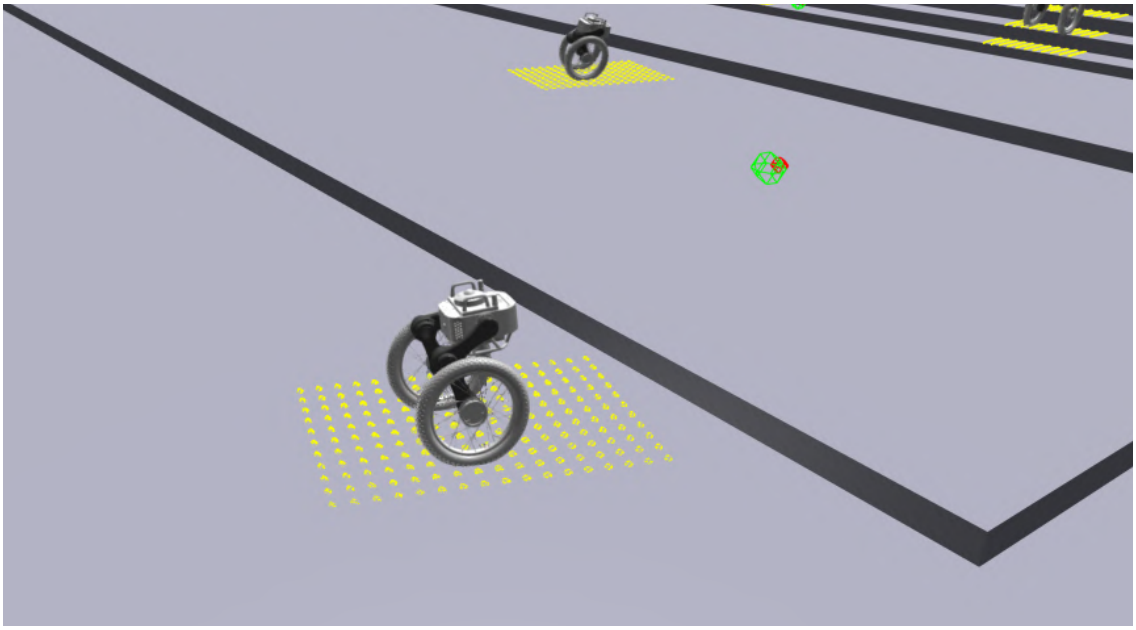

Figure 3.4 **Task setup:** Example for one of the 4096 simulated robots during training. The robot spawns in the environment and a goal pose is sampled. The goal pose is marked in green and red. The yellow dots represent the terrain height measurements which are part of the privileged information about the terrain.

3.2.1 State

Because we use an asymmetric actor-critic structure, we divide our state into two: an observation, to which both the actor and critic have access, and privileged information, which only the critic can access during training. Table 3.1 presents the observations used by the actor during training and deployment. These values were scaled by the normalization coefficients presented in the table as a pre-processing step before being used for training and inference by the neural networks to keep them in a reasonable range. During training, we add noise to this observation to make the agent more robust during deployment. We choose the scale of the noise according to how reliable these measurements are on the real robot. For example, commanded values do not require noise, since they will be provided by the user. Moreover, joint positions are very accurate, so the scale of noise applied is low. On the other hand, joint velocities can have very noisy measurements, as they are measured numerically from position variations. The agent should not learn to rely on these values being very precise, so we add a high level of noise to them.

The privileged observations are presented in table 3.2. First, the critic gets the linear velocity and base height of the robot. These values are not straightforward to obtain on the real

Table 3.1 **Observation:** This table presents the observation given to our agent. It also presents the normalizing coefficient used as well as each input dimension and percentage of noise applied during training.

Symbol	Observation	Units	Coeff.	Size	Noise (%)
$\vec{\dot{\theta}}$	Angular velocity	rad/s	0.25	3	±20
$\vec{\gamma}$	Projected gravity	N/A (unit vec)	1.0	3	±5
$\vec{g_{dir}}$	Direction to goal	N/A (unit vec)	1.0	2	±0
θ_{err}	Heading error	rad	1.0	1	±0
h_{target}	Base height command	m	1.0	1	±0
b	Terrain boolean	N/A	1.0	1	±0
$\vec{q}$	Joint positions	rad	1.0	4	±1
$\vec{\dot{q}}$	Joint velocities	rad/s	0.05	6	±150
$\vec{a_{last}}$	Last action	rad & rad/s	1.0	6	±0

robot, which is why they are not included in the actor's observation. The critic also gets the goal relative position in x, y, z instead of only a 2D direction in x, y coordinates. Finally, we also provide some ground truth information from the simulator, such as terrain height measurements, contact forces, and friction coefficient of the terrain the robot is currently in. The height measurements are sampled from a 1.6×1.0m grid around the robot and spaced equally. The contact forces are filtered using a sliding window with a window of 5 timesteps, equivalent to 0.1 seconds.

Table 3.2 **Privileged information:** This table presents the privileged information given to the critic during training. This information is only available in simulation. We present the normalizing coefficient applied to each input as well as their dimension. There is no noise applied to this data, as it will not be used during deployment.

Symbol	Observation	Units	Norm. Coeff.	Size
v_x	Linear velocity	m/s	0.25	1
h	Base height	m	1.0	3
$\vec{g_{rel}}$	Relative goal position	m	1.0	3
$H_{terrain}$	Terrain height	m	5.0	187 (17×11)
$\vec{f_{wheels}}$	Contact forces	N	0.01	6
μ	Friction coefficient	N/A	1.0	1

3.2.2 Actions

The actions are composed of a vector with six values, which are directly interpreted as commands for all the joints. The first four actions are joint position targets for the joints of the hips and knees, and the last two actions are angular velocity targets for the wheels. The actions are scaled by a 0.5 factor and then used as targets for PID controllers for each of the joints. However, for the articulated joint commands, we have an additional post-processing step before using them as position targets. We clip the angular position commands to be within $\pm\pi$ of the current joint positions. In fact, without this clipping, it would be impossible to know which way the motor should turn to reach the desired position, which makes an important difference in the robot's behavior. By clipping the values, we can then simply find the closest angle from the current position to the desired position, since it will never be more than half of a complete rotation. For PID control, the articulated joints use only a P gain of 20.0, while the wheel controllers have a P gain of 20.0 and a D gain of 2.0. These values are the same in the simulation and the real robot.

3.2.3 Rewards

All the rewards and their weighting coefficients are presented in table 3.3. In this section, we explain how each of these rewards is important and contributes to learning a good behavior. The rewards are separated into two categories. Rewards #1 to #3 are the main task-specific rewards, whereas rewards #4 to #13 are manually tuned dense rewards and penalties to encourage or discourage some behaviors. This second category of rewards was carefully tuned to obtain smoother and improved behaviors once the rough motion was discovered using the main rewards after training some agents and analyzing preliminary results.

The main idea behind our position-based task formulation is to give more freedom to the actor to explore different strategies to climb steps and find a good solution that might have been difficult to obtain by crafting it manually. Therefore, the main reward of our task, reward #1 only comes into play during the last 2 seconds of the episode ($T_r = 2s$). This reward encourages the robot to be at the goal location at the end of the episode but does not add any further constraints. However, this makes it very difficult for convergence. Therefore, we introduce rewards #2 and #3. Reward #2 is a temporary reward that is only given to the agent until it reaches half of the best possible performance for reward #1. It gives an incentive for the robot to have a linear velocity toward the goal. This is exactly what we want to avoid because it would encourage high velocity throughout the whole episode, which is why it is only temporary to accelerate convergence. Reward #3, on the other hand, pe-

Table 3.3 **Rewards:** This table presents all the reward terms used for training our stair-climbing agent with the Ascento robot as well as their respective coefficients. Rewards #1 to #3 are task-specific, while rewards #4 to #13 are used to obtain a smooth motion and penalize undesired behaviors.

#	Reward	Formula	Coefficient
1	$r_{position}$	$\frac{1}{T_r}\frac{1}{1+\|\vec{x}-\vec{goal}\|^2}$ if $t > T - T_r$	10.0
2	r_{pos_bias}	$\frac{\dot{\vec{x}}\cdot(\vec{goal}-\vec{x})}{\|\dot{\vec{x}}\|\|\vec{goal}-\vec{x}\|}$	1.0
3	r_{stall}	-1 if $\|\dot{\vec{x}}\| < 0.1m/s$ and $\|\vec{x}-\vec{goal}\| > 0.5m$	1.0
4	r_{face_goal}	$-\|\theta - \theta_{goal}\|$ if $\|\vec{x}-\vec{goal}\| > 0.5m$	0.1
5	$r_{wheel\_forces\_xy}$	$-\sum \|f_{\vec{wheels}\_x} + f_{\vec{wheels}\_y}\|$	0.01
6	r_{base_height}	$\exp(-4\cdot(h - h_{target}))$	1.0
7	$r_{orientation}$	$-\sum(\vec{\gamma_{xy}})^2$	0.2
8	r_{def_pos}	$-\sum \|\vec{q} - \vec{q_{default}}\|$	0.1
9	r_{dof_acc}	$-\sum \ddot{\vec{q}}$	2.5e-7
10	r_{action_rate}	$-\sum \|\vec{a} - \vec{a_{last}}\|^2$	0.014
11	$r_{lin\_vel\_z}$	$-v_z^2$	1.0
12	$r_{ang\_vel\_z}$	$-\omega_z^2$	0.1
13	$r_{ang\_vel\_xy}$	$-\sum \vec{\omega_{xy}}^2$	0.5

nalizes stalling unless the robot is already close to the goal. This reward is kept throughout the training process since it does not have any undesired effects.

Then, we have rewards #4 to #13. These rewards can be separated into different categories. First, rewards #4 and #5 help shape how we want the robot to go up the steps. Reward #4 encourages the robot to locomote forward and not backward, always facing the goal, while reward #5 penalizes forces on the wheels in the xy to discourage the robot from going full speed and relying only on friction with the steps to go up. Second, rewards #6 to #8 are used to maintain the base in a stable position. Reward #6 encourages a good base height, measured from the ground. The target base height is sampled from the operating range of the robot, which for the Ascento robot is between 45cm and 65cm. Rewards #7 and #8 penalize deviating from a stable base rotation and deviating from the default joint configuration of

the legs respectively. Finally, rewards #9 to #13 are penalties to ensure smooth motions. They penalize action variation rate to avoid jerky movements, as well as linear velocity along the vertical axis, and angular velocities of the base.

3.3 Curriculum

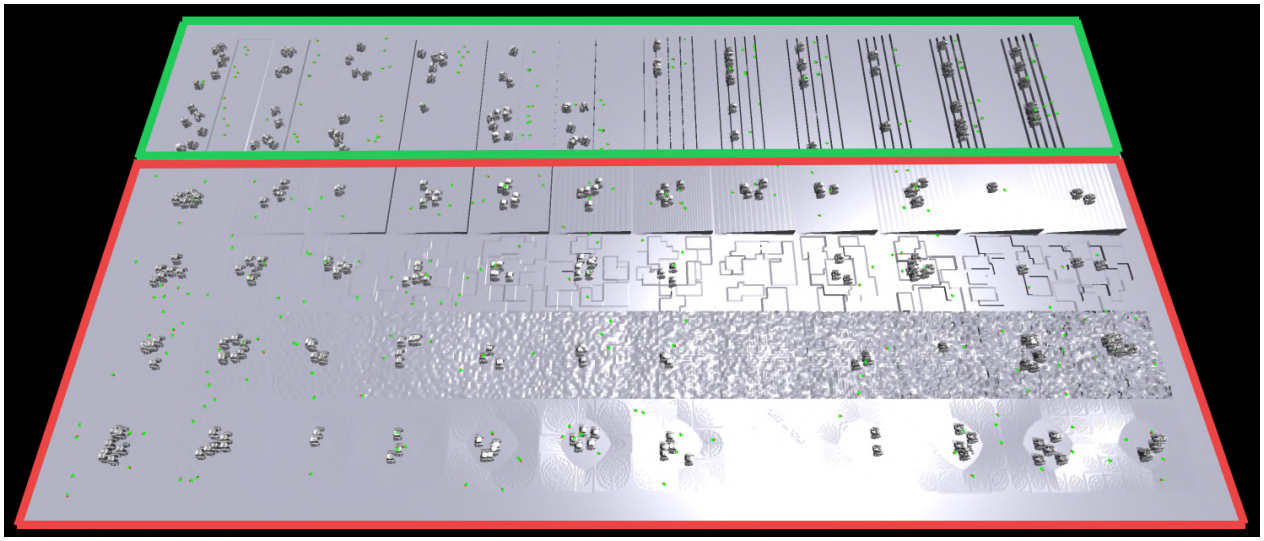

Figure 3.5 **Training curriculum:** Different training terrains, progressively harder from left to right. The stairs terrains, where the terrain boolean is set to 1 during training, are delimited in green. In the other terrains, this observation is set to 0.

As mentioned in section 3.1, we use a mix of stairs terrains and rough terrains of gradually increasing difficulty to train our policy. The use of a curriculum has proven to help with sample efficiency and convergence in the field of machine learning in general [42], and is particularly useful in RL, where the agent learns from interactive experience. By presenting easier terrains to the robot in the beginning, we facilitate convergence towards a stable policy before facing more challenging obstacles. Furthermore, for the stairs environments, we start with a single step so the robot can learn the general motion needed to climb stairs. In the second half of the curriculum, we introduce continuous staircases of increasing difficulty. In this work, we use a curriculum inspired by the one proposed in the work of Rudin et al. [19]. At the beginning of training, 4096 robots are spawned in the environment, all starting in the easiest terrains of each type. Every time an episode ends, either because of reaching the time limit, or because of a terminal state such as a fall, we evaluate the performance of the robot and decide if it should progress in the curriculum, regress, or remain at the same level. If the robot is within 20cm of the goal, we move it one level up, and if the robot is more than 50cm away from the goal, we move it one level down, unless it is already at the lowest

level. If a robot reaches the highest level of its terrain, it is randomly assigned a new level to start from. By doing this shuffling after the final level, we make sure that easier levels are still visited later on during training to avoid catastrophic forgetting. Figure 3.5 illustrates a top view of the environment when the agent already masters all the levels and the robots are equally distributed.

3.4 Learning Algorithm

We used the RL algorithm PPO, with an asymmetric actor-critic structure. This concept was first introduced by Pinto et al. [43] in 2017 for robotics control. It is a variant of the standard actor-critic algorithm in RL, where the actor and critic use separate networks, allowing them to update independently [44]. This setup enables more efficient and stable learning as the two components can evolve at different rates, reducing the risk of oscillations or divergence. The asymmetric version means that the actor and critic do not share the same observations. In our case, we provide the actor only proprioceptive information and commands, while the critic has access to privileged information. Privileged information refers to additional data or observations that are not typically accessible during the deployment or execution of the learned policy, but accessible from the simulator during training. By incorporating this extra information into the critic network's learning process, the algorithm can improve its value estimation and guidance to the actor, resulting in better policy updates and more efficient learning. The use of privileged information by the critic network can be particularly advantageous in scenarios where such information aids in understanding the environment dynamics, task structure, or long-term consequences of actions. While the actor is constrained to learn a policy solely based on the observable state, the critic's access to privileged information can enhance its ability to guide the actor towards better decisions, leading to improved overall performance. In our case, providing terrain information helps to locate the stairs in the environment and provides better guidance to the actor. We also feed the critic with contact forces, friction coefficients, and velocity measurements, which are not available during deployment.

In this work, we use the open-source PPO implementation published by the Robotics Systems Lab (RSL) from ETH [19] [1]. This implementation is highly parallelized and is specifically designed to be used with the Isaac Gym simulator using large batch sizes. We simply remove the bootstrapping element from this implementation, since we deal with a finite horizon task formulation, while their implementation is meant for infinite horizon tasks such as tracking

[1] https://github.com/leggedrobotics/rsl_rl

velocity commands. Our policies are capable of solving all the levels of the curriculum after 5k learning steps, and we train them for a total of 20k learning iterations. For all of our training and experiments, we used a single GPU, an Nvidia DGX A100 with 16 GB of memory. The PPO parameters we used are presented in appendix A.

3.5 Sim-to-real Transfer

Table 3.4 **Domain randomization:** This table presents the physical parameters that were randomized during training to facilitate sim-to-real transfer.

Parameter	Operation	Range	Units
Friction	N/A	[0.2, 1.0]	N/A
Base mass	Addition	[-1.0, 1.0]	kg
Leg links mass	Multiplication	[0.9, 1.1]	N/A
Base CoM	Addition	[-0.02, 0.02]	m
Push robots	N/A	[-0.5, 0.5]	m/s
Delay	N/A	[0, 20]	ms

During training, we use domain randomization to facilitate transfer to the real world [45,46]. First, we randomize friction with the terrain as this parameter is very important to our task. We noticed that if friction is too high, the robot learns to use its grip between the wheels and the step to go up. While this strategy works in simulation with perfect collisions and friction, it does not transfer to the real world because of slippage due to the wheel momentarily losing contact with the step. Furthermore, we randomize masses and center of mass placement to remain robust to slight mass changes on the real robot. We also apply a random "push" to the robot, which is simulated by assigning a random base linear velocity sporadically throughout training. This helps with overall robustness to disturbances and deviations from desired trajectories. Finally, we randomize delays in the control loop [47]. In fact, the controller in simulation always runs exactly at the desired frequency, which is not the case in the real world due to computing, sensing, and actuation delays. The control loop runs at 50Hz, so we choose to add random delays between 0 and 20ms. This is a very important part of our sim-to-real process, and we highlight its effect in section 4.3. In practice, we introduce delays by randomly using old states (from the last time step) to compute actions. Table 3.4 presents a summary of our randomizations.

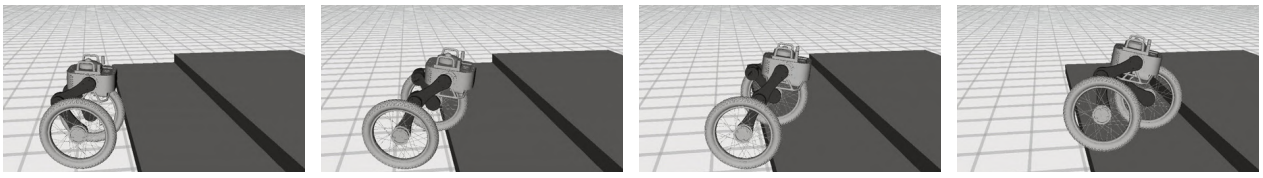

Figure 3.6 **Stair climbing motion in Gazebo:** This figure presents the robot going up a step in the simulator Gazebo. We use this simulator as a sim-to-sim validation tool before deploying policies on the real robot.

Next, as a validation step before deploying our policy on a real robot, we use the simulator Gazebo[2] to evaluate sim-to-sim transfer. We have found that Gazebo seems to better match the real-world behavior of our real robot. Moreover, Gazebo integrates easily with ROS, which we use as a framework for the onboard software stack on the robot. Therefore, during this validation step, we run exactly the same code that will run on the robot hardware. Figure 3.6 shows the results of our policy going up a 20cm step in Gazebo. This is also demonstrated in our video [36] at 1:20.

3.6 Deployment

During deployment, we no longer need the privileged information nor the critic. To control the robot with the learned policy, i.e. the actor network, we simply need to reconstruct the observation:

- **Proprioceptive state:** The proprioceptive state includes joint information as well as orientation information. The joint positions are measured from the encoders, and the joint velocities are calculated numerically from joint position change during the last control iteration. Angular velocities and projected gravity are calculated from Inertial Measurement Unit (IMU) data.
- **Commands:** We also need to provide the agent with commands as part of the observation. These include goal direction, base height target, heading error, terrain boolean, and last actions. For our experiments, we use a remote controller to provide these commands. We map the x component of the goal direction to the vertical axis of a joystick, and the heading error to the horizontal axis of the same joystick, while the y component of the goal direction component is always 0. In practice, this behaves similarly

[2]https://gazebosim.org/home

to controlling a velocity-based controller in terms of user experience, with the added benefit that the robot can slightly deviate from the commands to successfully overcome obstacles. The base height command is simply kept constant to a stable height but can be changed through buttons The terrain boolean is also mapped to a button for ease of use, and the last actions are internally stored from the previous iteration. While we use a remote controller for our experiments, these commands could be sent by a high-level planner.

Figure 3.7 shows an overview of the full system.

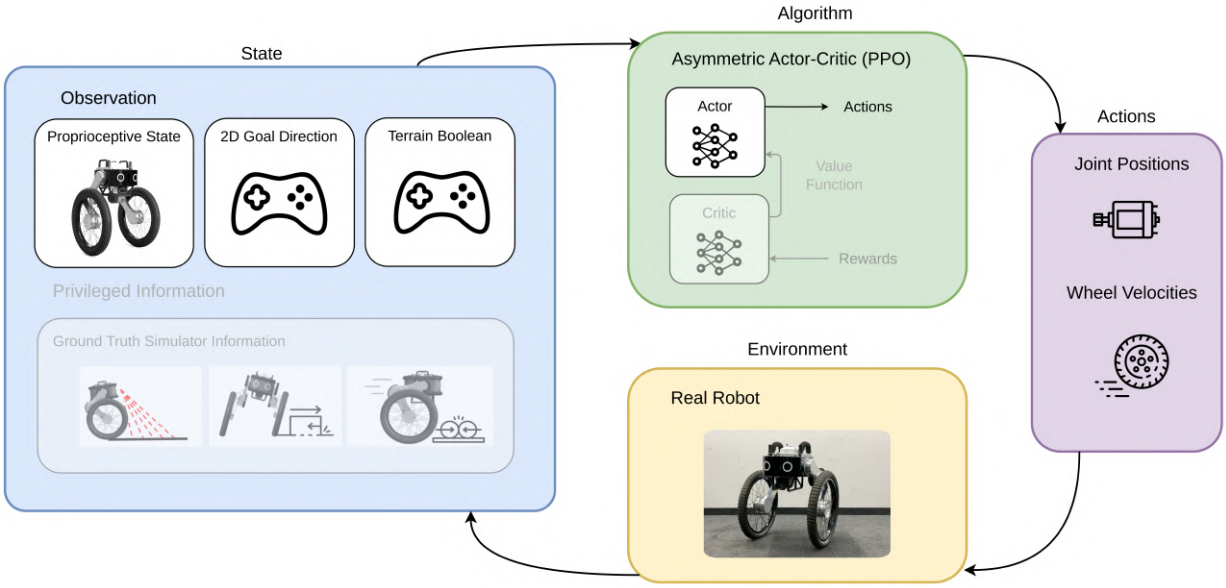

Figure 3.7 **System overview during deployment:** We present an overview of the complete deployment setup. The trained agent interacts with the real environment and is composed only of the actor-network. The observation is retrieved directly from proprioceptive sensors on the robot and provided to the agent, which in turn outputs an action.

CHAPTER 4 EXPERIMENTS & RESULTS

In this chapter, we present the results of our method in simulation and on the real robot. Overall, the trained policy was able to develop good motions for the Ascento robot to climb stairs. Different motions can be achieved by shaping the rewards and applying different weights to each reward term. In simulation, the robot was able to climb staircases with steps of up to 22cm of height and down to 35cm of width. As for the real robot, we show that it is able to climb steps of up to 15cm. In the following sections, we discuss these results and highlight some important components of our method through our experiments.

4.1 Simulation Results

We present our simulation results in our video [36]. At 0:18, we can see different behaviors that can be obtained from different tuning of the reward coefficients. In the first clip, we can see that the robot developed a jumping motion and goes up with its two wheels at once. In the second clip, we can see the robot performing a step motion that is slower and steps up with one leg at a time. Between these two clips, the difference is that we remove the wheel contact force penalty for the first one, and increase it for the second one. Without this penalty, the robot can pick up more linear velocity and use its friction against the step to move up, while with a heavier penalty, it learns to stop and step up vertically to avoid a higher contact force with the step. Our final policy is an in-between, using the coefficients presented in section 3.3, shown in our video [36] at 0:30. In the following subsections, we highlight the importance of the position task formulation as well as the terrain boolean observation we introduced.

4.1.1 Position Control Analysis

Figure 4.1 shows the velocity profile of our policy when going up a 20cm step in the simulation. In contrast to the velocity controller shown in figure 2.2, we can see that the controller adopts a very different velocity profile. We notice that the robot accelerates at the beginning of its trajectory to approach the goal that is in front of it. However, when it reaches the step, it decelerates and performs the learned step-up motion, before accelerating again to reach the goal. This would not have been possible with a velocity-based task formulation, where the robot is constrained to follow a specific velocity. Here, we only provide a direction toward the goal and the robot is free to adjust its velocity to overcome the step that is in the way.

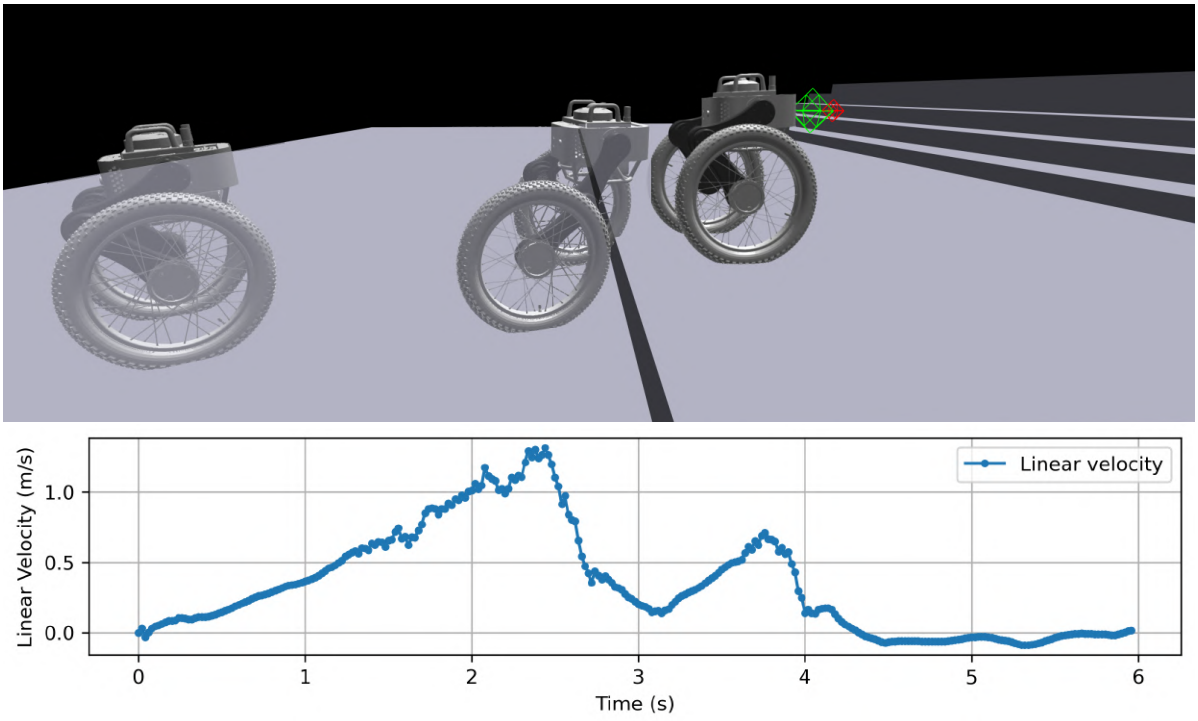

Figure 4.1 **Step motion:** This figure presents an overlaid illustration of the Ascento robot at different stages of the step-up motion. We also present its velocity profile.

Additionally, it is important to keep in mind that the controller does not use any perception, and it has no knowledge of the exact location of the goal. The detection of the step happens only through proprioception. The robot learns to advance with one leg slightly in front of the other, which allows it to infer the presence of the step once it bumps into it.

4.1.2 Terrain Boolean Importance

Our proposed terrain boolean observation is a very important component of our method. In fact, it allows the robot to learn specialized knowledge for stair-climbing skills without compromising performance on rough terrain. In our video [36], we highlight the importance of this observation at 0:45. First, we analyze our final policy by modifying its observation. We notice that if we turn the boolean off and test the robot in a step terrain, it is not able to overcome it. However, the robot remains stable and does not fall. It presses against the step and leans forward since it continuously keeps receiving a goal direction commanding it to go forward. Once we turn the terrain boolean back on, the robot is able to go up the step. Second, we show the robot getting to a goal in flat terrain containing no steps with the terrain boolean turned on. This is an interesting experiment allowing us to better

understand what the robot learned from this observation. We notice that the robot gets to the goal position successfully, but it adopts a different locomotion strategy. It showcases a leg split, a position allowing it to better detect a step if it encounters one. This locomotion style is less smooth, and not ideal for terrains that do not contain stairs. Finally, at 1:07, we show an ablation where the policy was trained without the terrain boolean concept. We notice that his policy always adopts a locomotion strategy with a leg split, similar to our proposed policy when the terrain boolean is turned on, but it is not as good at climbing steps, failing to overcome a 20cm step. Overall, this shows that when trying to combine stair-climbing skills and all-around robustness, the policy needs to make compromises and performance decreases for both tasks. Having a distinction through our proposed boolean observation allows it to specialize and achieve better performance.

4.2 Real World Results

During the hardware experiments, our policy was able to climb 15cm steps with the Ascento robot. Figure 4.2 illustrates its behavior step by step, and we also show two successful climbs in our video [36] at 2:00. We can see that the robot drives toward the step with a slight leg split and some speed. The left wheel initially makes contact with the step and, with some grip, starts moving up. There is some slippage between the wheel and the step, but as the robot infers the presence of the step from its proprioceptive state and the fact that the terrain boolean is turned on, it lifts the left leg up. Then, the right leg follows and the robot stabilizes at the top of the step. These results prove that our method is promising for complex tasks such as stair-climbing, as the Ascento robot was previously unable to overcome steps of this height. During our experiments, we noticed that it is important to compromise between encouraging a behavior that uses velocity and grip with the step, versus a behavior that relies only on stepping up one leg at a time. In fact, a policy that adopts the formerly described behavior will fail due to slippage. Colliding with the step causes a momentary loss of contact, therefore making it impossible to go up. Collisions and friction with the edge of the step are not perfectly modeled in simulation, which makes it hard for the robot to learn how to overcome these challenges. On the other hand, a behavior relying only on stepping up one leg at a time fails due to the sim-to-real gap in actuation response. This will be discussed further in section 6.1.

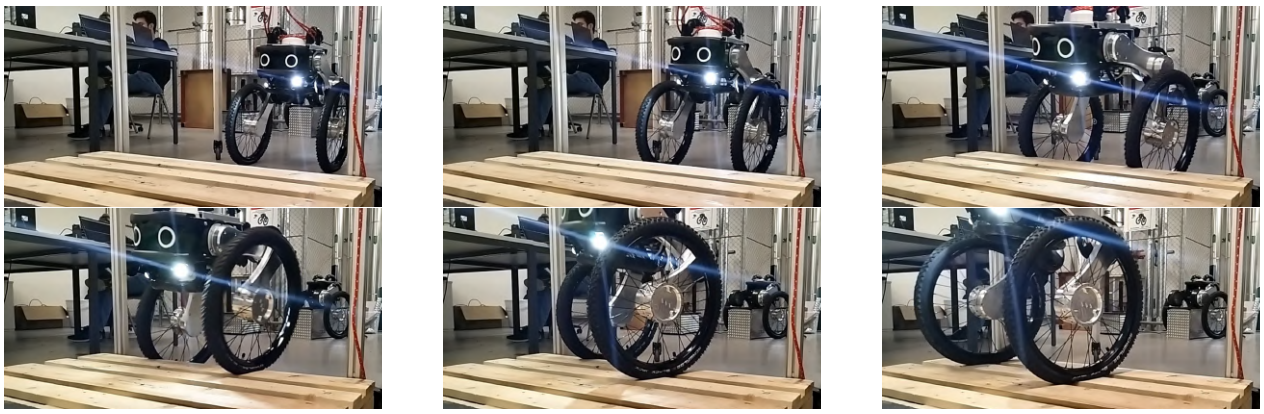

Figure 4.2 **Real world results:** Ascento robot going up a 15cm step with our learned policy. Images are taken at a 0.25-second interval.

4.3 Sim-to-real Transfer

In this section, we highlight the importance of adding random delays during training our policy. As discussed in section 3.5, when deploying our policy in the real robot, there are some delays that are not captured by the simulation. These could be due to sensor delays, actuation delays, or even computing delays. This is especially important for the purpose of climbing steps because it is a very dynamic motion where timing is crucial. In our video [36], at 1:30, we show that policies learned without accounting for this sim-to-real gap and omitting delays in the simulation are very unstable. They are not robust to disturbances and have very jerky behavior when going up a step. When climbing a 10cm step, the joints exhibit high-frequency oscillations and some joint velocities reach a peak of 15 rad/s. After adding the delay randomization, we can see that the policies are more robust and smoother, successfully climbing the same step without any joint exceeding a velocity of 1.5 ras/s. This is shown at 1:42 in our video [36].

CHAPTER 5 GENERALIZATION OF THE METHOD

In this chapter, we study the generalization of our method to other legged and wheeled-legged robots. In fact, although it was designed for the Ascento robot, we believe it is applicable to other robot morphologies with minor adaptations.

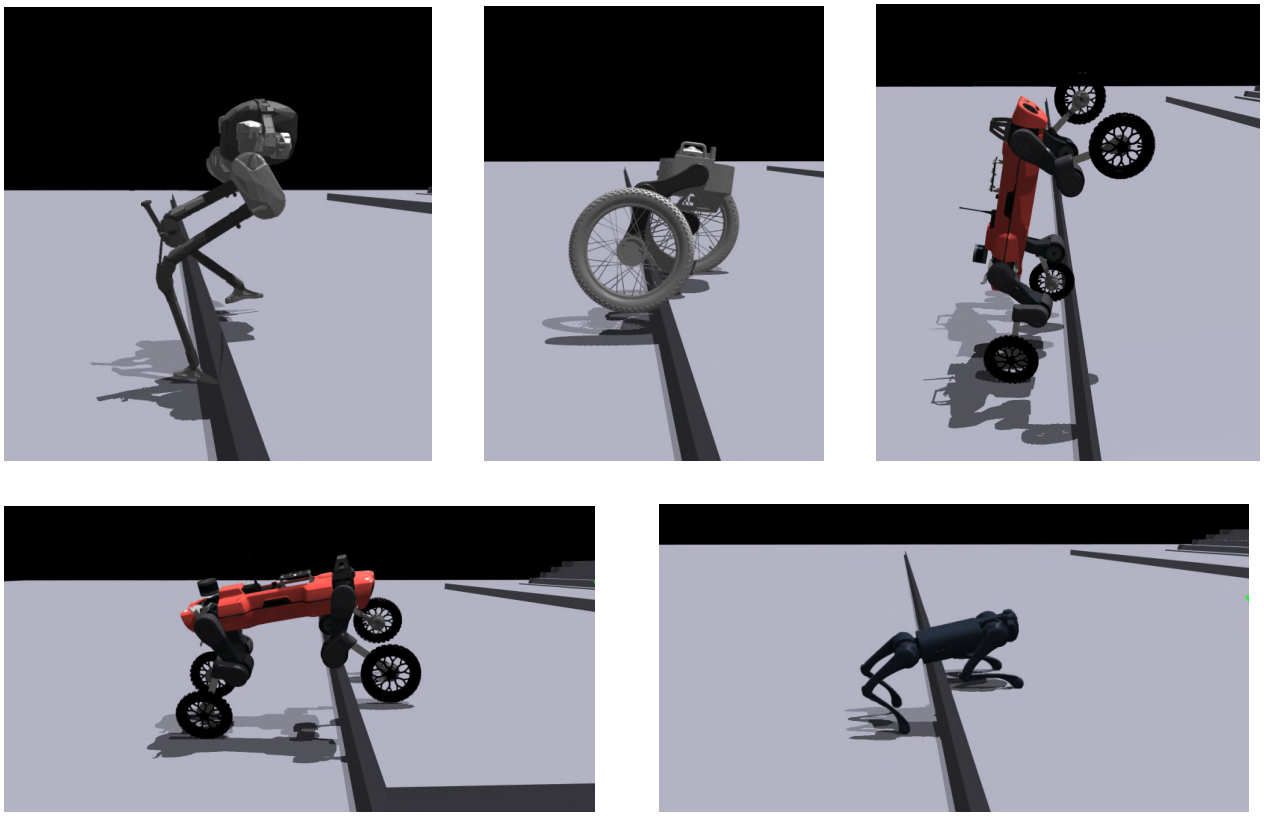

Figure 5.1 **Different robots:** This figure presents the robots used for evaluating the generalization capabilities of our method. Top: Cassie, Ascento Robot, ANYmal on Wheels (Biped). Bottom: ANYmal on Wheels (Quadruped), Unitree Go1.

5.1 Robots & Method adjustments

We evaluate our proposed method on four different robots: The Unitree Go1 quadruped [48], the bipedal robot Cassie [49], the wheeled-legged balancing robot Ascento [2] and the wheeled-legged quadruped ANYmal on Wheels [29], illustrated in figure 5.1. For ANYmal on Wheels, we conduct experiments in quadruped mode and biped mode. Our method is applicable without any modifications, except for the reward function. We conserve reward terms #1 to #3 from table 3.3, but the others were specific to the Ascento robot. Appendix B presents the different additional rewards for each robot.

5.2 Results & Discussion

We evaluate our method on all robots and conduct an ablation study to showcase the importance of the asymmetric actor-critic structure using privileged information as well as the terrain boolean observation. Table 5.1 shows the success rate obtained for each robot for climbing a single step. For each experiment, 2000 trials are conducted per robot. First, we conclude that the privileged information is critical for successfully learning the task. We attribute this mainly to the terrain information, which helps guide the actor to a terrain-cautious behavior. On the other hand, the proposed boolean observation is particularly important for bipedal robots because it allows them to learn distinct skills for stair-climbing while learning a robust general policy to navigate the other terrains. All the trained robots achieve a success rate higher than 75% on standard height steps of 18cm with our method. Similarly to the Ascento robot, the other robots also showcase gaits capable of detecting and traversing the steps through proprioception information only. We present successful runs with our method in a supplementary material video [50].

Finally, the importance of the terrain boolean observation for the proposed method can be especially seen when applied to bipeds and wheeled-legged robots. In fact, it allows the robots to learn specialized knowledge for stair-climbing skills without compromising performance on rough terrain. Table 5.1 shows that Cassie, Ascento, and ANYmal on Wheels achieve a higher success rate when on stairs in the *Bool On* experiment than the *No bool* ablation, whereas the *Bool Off* variant outperforms it in other terrains. The same policy successfully learns these two behaviors conditioned on the boolean observation. For example, we notice that our learned policy for Cassie adopts a slower gait with higher strides when the terrain bool is present. It allows it to maintain more stability and climb over the steps when its feet collide with it. When we turn the bool off, it adopts a quicker gait which is more efficient in the

other terrains. Similarly, Ascento showcases a bigger leg split when in stair mode, a position allowing it to lift its front leg when detecting a step through proprioception. When omitting the terrain boolean in the *No bool* ablation, a compromise has to be made to maximize performance on all terrains, which results in a less specialized policy. On the other hand, the boolean is not essential for the Go1, as all experiments, except the *No priv.* ablation, achieve a good policy. This is easier because of its four legs, providing more stability when traversing steps. Lastly, the ANYmal on Wheels also exhibits interesting behavior. In both quadrupedal and bipedal modes, it learns to mostly use its wheels to locomote and step when encountering a step. This behavior is not developed without the boolean, hence the low success rates. We conclude that our task formulation as well as the use of privileged terrain information during training are the key components to discover this behaviour.

Table 5.1 **Simulation results for step climbing:** This table presents the results of our method on different robots. For each robot, we test our method with the boolean observation set to 1 and 0, as well as two ablations. We Each experiment is conducted for 2000 episodes.

Robot	Exp.	Success Rate (%)					
Step height		14cm	18cm	22cm	26cm	30cm	Other terrains
Unitree Go1	Bool on	99.7	99.7	98.5	43.1	*0.0	97.7
	Bool off	98.8	98.3	93.5	19.1	*0.0	98.6
	No bool	100.0	99.8	99.6	56.4	*0.0	98.7
	No priv.	15.4	12.9	8.8	3.9	*1.0	95.3
Cassie	Bool on	100.0	100.0	99.7	99.0	98.3	98.8
	Bool off	97.4	92.2	87.8	78.2	64.7	98.1
	No bool	99.2	97.2	90.6	79.6	74.3	69.4
	No priv.	0.0	0.0	0.0	0.0	0.0	22.3
Ascento	Bool on	87.6	82.3	69.4	*31.4	*1.4	97.7
	Bool off	23.2	18.6	9.6	*0.1	*0.0	98.1
	No bool	87.9	82.5	47.9	*4.4	*0.0	62.5
	No priv.	0.0	0.0	0.0	*0.0	*0.0	80.2
ANYmal on Wheels (Quadruped)	Bool on	93.2	93.8	90.2	61.1	18.7	95.9
	Bool off	73.8	40.8	18.9	4.8	0.5	97.7
	No bool	34.3	37.9	26.4	34.4	14.8	20.9
	No priv.	0.3	0.0	0.0	0.0	0.0	62.7
ANYmal on Wheels (Biped)	Bool on	93.9	92.8	93.6	93.6	93.4	85.0
	Bool off	71.0	45.4	27.8	14.2	6.2	94.9
	No bool	68.9	70.0	58.2	35.0	34.4	85.7
	No priv.	0.0	0.0	0.0	0.0	0.0	78.8

Experiments: *Bool on:* Our proposed method with terrain bool set to 1.
Bool off: Our proposed method with terrain bool set to 0.
No bool: Ablation trained without a terrain boolean observation.
No priv.: Ablation trained without privileged information.
Legend: *Success* means the robot reached the goal ±20cm.
***Note:** This height exceeds the step height the robot was trained on.

CHAPTER 6 CONCLUSION

The goal of this thesis was to overcome a critical limitation in the deployment of Ascento's wheeled-legged robot in multi-story settings. The inability of the robot to traverse stairs constrained its usefulness in security patrols across a variety of terrains. To overcome this, we proposed a method using RL to train a controller that equips the robot with stair-climbing capabilities. Our proposed methodology diverged from the traditional velocity-based task formulation during the training phase, favoring a position-based RL task. We used an asymmetric actor-critic structure, allowing the critic network to leverage privileged information during training, and removing the need for this data during deployment. Furthermore, we introduced a boolean observation that served as a mode switch for the stair-climbing skill, a feature that proved instrumental to achieve good results at this task while remaining robust in other types of terrain. The application of these techniques and methodologies led to the creation of a "blind" controller that does not require exteroceptive sensor information or a positioning system. Our method allowed Ascento to climb 15cm steps, an achievement previously impossible with the current iteration of this robot. These advancements, once integrated into the robot control stack, will significantly enhance the robot's operational range. Finally, we evaluated our proposed method on different robot morphologies and showed that it can be generalized. In the following section, we discuss the limitations of our method and possible directions for future work.

6.1 Limitations & Future Work

While the implementation and successful real-world application of our RL controller provides a strong foundation for further improvements and extensions, some limitations remain. First, although our method does not require any precise information about its surrounding terrain, it does need access to the terrain boolean observation that we introduced. This is already a great advancement in reducing the perception complexity requirements to traverse obstacles compared to other locomotion methods, which require a complete elevation map of the environment for locomotion [51, 52]. An interesting direction for future work could explore the possibility of inferring this simple boolean observation from perceptive data, such as image or depth information. This could potentially be achieved using supervised learning.

Furthermore, another direction would be to improve the sim-to-real gap. Transferring the learned skills from simulation to reality continues to be a challenge for our method and the

field of RL in general. For example, it was possible to learn policies that would climb continuous staircases in simulation using very low velocities and going up one leg at a time. However, these were not successful when deployed on the real system, as actuation responses were too different from the simulation. In fact, the controller learned to provide very aggressive commands to the robot joints when stepping up the step. In simulation, this behavior worked well, allowing the robot to climb steps of up to 22cm while maintaining a smooth motion and respecting the joint velocity limits of 10 rad/s. In the real system, however, a high overshoot in the commands caused some of the joints to reach velocities of up to 25 rad/s, which is unsafe and triggers a soft e-stop on the robot. Techniques such as modeling the actuators with real-world data have been successful to improve sim-to-real transfer for quadruped locomotion [14]. This would likely be beneficial for the Ascento robot as well.

Finally, another interesting approach that would be interesting to explore to improve sim-to-real transfer is encoding the randomized parameters during training and inferring them at deployment. Methods of the sort, such as RMA, have been successfully used for different articulated robots [20, 38]. This could even potentially allow the learned policies to be used across different robots of the same morphology.

REFERENCES

[1] F. J. Rubio Montoya, F. J. Valero Chuliá, and C. Llopis Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, pp. 1–22, 2019.

[2] V. Klemm *et al.*, "Ascento: A two-wheeled jumping robot," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7515–7521.

[3] P. Biswal and P. K. Mohanty, "Development of quadruped walking robots: A review," *Ain Shams Engineering Journal*, vol. 12, no. 2, pp. 2017–2031, 2021.

[4] M. Bjelonic *et al.*, "A survey of wheeled-legged robots," *Robotics in Natural Settings: CLAWAR 2022*, pp. 83–94, 2022.

[5] J. Lee, M. Bjelonic, and M. Hutter, "Control of wheeled-legged quadrupeds using deep reinforcement learning," in *Robotics in Natural Settings: CLAWAR 2022*. Springer, 2022, pp. 119–127.

[6] E. Vollenweider *et al.*, "Advanced skills through multiple adversarial motion priors in reinforcement learning," *arXiv preprint arXiv:2203.14912*, 2022.

[7] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[8] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 3. IEEE, 2004, pp. 2619–2624.

[9] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 590–595.

[10] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[11] R. S. Sutton *et al.*, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[12] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[13] M. Hutter *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.

[14] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[15] J. Schulman *et al.*, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[16] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[17] J. Lee *et al.*, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[18] T. Miki *et al.*, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[19] N. Rudin *et al.*, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[20] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: learning a unified policy for manipulation and locomotion," in *Conference on Robot Learning*. PMLR, 2023, pp. 138–149.

[21] N. Rudin *et al.*, "Advanced skills by learning locomotion and local navigation end-to-end," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2497–2503.

[22] Y. Ji, G. B. Margolis, and P. Agrawal, "Dribblebot: Dynamic legged manipulation in the wild," *International Conference on Robotics and Automation*, 2023.

[23] X. Cheng, A. Kumar, and D. Pathak, "Legs as manipulator: Pushing quadrupedal agility beyond locomotion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[24] Y. Ji *et al.*, "Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot," *International Conference on Intelligent Robots and Systems*, 2022.

[25] Y. Gong *et al.*, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *2019 American Control Conference (ACC)*, 2019, pp. 4559–4566.

[26] G. A. Castillo *et al.*, "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.

[27] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12 149–12 155.

[28] J. Siekmann *et al.*, "Blind bipedal stair traversal via sim-to-real reinforcement learning," *arXiv preprint arXiv:2105.08328*, 2021.

[29] M. Bjelonic *et al.*, "Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2116–2123, 2019.

[30] ——, "Rolling in the deep–hybrid locomotion for wheeled-legged robots using online trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3626–3633, 2020.

[31] ——, "Whole-body mpc and online gait sequence generation for wheeled-legged robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8388–8395.

[32] H. Chen *et al.*, "Underactuated motion planning and control for jumping with wheeled-bipedal robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 747–754, 2020.

[33] L. Cui *et al.*, "Learning-based balance control of wheel-legged robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7667–7674, 2021.

[34] W. Zhu, F. Raza, and M. Hayashibe, "Reinforcement learning based hierarchical control for path tracking of a wheeled bipedal robot with sim-to-real framework," in *2022 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2022, pp. 40–46.

[35] C. Salzmann, "Recovery control in different terrains for a two-wheeled robot with reinforcement learning," Master's thesis, Eidgen ossische Technische Hochschule (ETH), Zurich, Switzerland, 2022.

[36] S. Chamorro. Reinforcement learning for stair climbing with ascento: A two-wheeled legged robot. Youtube, https://youtu.be/GmEBpgXxbmQ. [Online]. Available: https://youtu.be/GmEBpgXxbmQ

[37] J. Tobin *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[38] A. Kumar *et al.*, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[39] V. Makoviychuk *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[40] 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*. IEEE, 2012, pp. 5026–5033.

[41] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[42] Y. Bengio *et al.*, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[43] L. Pinto *et al.*, "Asymmetric actor critic for image-based robot learning," *arXiv preprint arXiv:1710.06542*, 2017.

[44] I. Grondman *et al.*, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[45] J. Tobin *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[46] J. Tan *et al.*, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[47] A. Rupam Mahmood *et al.*, "Setting up a reinforcement learning task with a real-world robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4635–4640.

[48] Unitree, "Unitree go1," https://www.unitree.com/go1/, accessed: 2023.

[49] Agility Robotics, "Cassie," https://agilityrobotics.com/, accessed: 2023.

[50] S. Chamorro. Reinforcement learning for blind stair climbing with legged and wheeled-legged robots. Youtube, https://youtu.be/Ec6ar8BVJh4. [Online]. Available: https://youtu.be/Ec6ar8BVJh4

[51] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.

[52] T. Miki *et al.*, "Elevation mapping for locomotion and navigation using gpu," 2022.

APPENDIX A LEARNING ALGORITHM PARAMETERS

In this section, we present the parameters used for training our PPO agent.

Table A.1 **PPO learning parameters:** Training hyperparameters for the PPO algorithm.

Parameter	Value
Actor hidden layers	[512, 256, 128]
Critic hidden layers	[512, 256, 128]
Activation layers	ELU
Initial noise std.	1.0
Value loss coefficient	1.0
Clipping parameter	0.2
Entropy coefficient	0.2
Number of learning epochs	5
Number of parallel environments	4096
Number of steps / environment	24
Number of mini-batches	4
Mini batch size	24576
Learning rate	0.01
Discount factor	0.99
Generalized advantage estimation parameter	0.95
Desired KL coefficient	0.01
Max. gradient norm	1.0

APPENDIX B REWARD SHAPING REWARDS FOR DIFFERENT ROBOTS

In this appendix, we present the rewards used to train the Unitree Go1 robot, Cassie, and the ANYmal on Wheels both in biped and quadruped modes. These rewards are used in addition to rewards #1 to #3 from table 3.3 to encourage smooth behaviors.

Table B.1 **Unitree Go1 shaping rewards:** Similar shaping rewards to those presented in table 3.3. The only new term introduced is reward #10, encouraging feet air time to learn a policy that does not drag its feet too close to the ground.

#	Reward	Formula	Coefficient
1	r_{face_goal}	$-\|\theta - \theta_{goal}\|$ if $\|\vec{x} - \vec{goal}\| > 0.5m$	0.1
2	r_{base_height}	$\exp\left(-4 \cdot (h - h_{target})\right)$	1.0
3	$r_{orientation}$	$-\sum(\vec{\gamma_{xy}})^2$	0.2
4	r_{def_pos}	$-\sum \|\vec{q} - \vec{q_{default}}\|$	0.1
5	r_{dof_acc}	$-\sum \vec{\ddot{q}}$	2.5e-7
6	r_{action_rate}	$-\sum \|\vec{a} - \vec{a_{last}}\|^2$	0.014
7	$r_{lin\_vel\_z}$	$-v_z^2$	1.0
8	$r_{ang\_vel\_z}$	$-\omega_z^2$	0.1
9	$r_{ang\_vel\_xy}$	$-\sum \vec{\omega_{xy}}^2$	0.5
10	$r_{feet\_air\_time}$	$\sum(t_{air} - 0.5)$	1.0

Table B.2 **Cassie shaping rewards:** Similar shaping rewards to those presented in tables 3.3 and B.1. Additionally, reward #7 penalizes motor torques, and reward #9 penalizes lifting both feet off the ground for the bipedal robot Cassie to maintain good balance.

#	Reward	Formula	Coefficient
1	r_{face_goal}	$-\|\theta - \theta_{goal}\|$ if $\|\vec{x} - \vec{goal}\| > 0.5m$	0.1
2	$r_{orientation}$	$-\sum(\vec{\gamma_{xy}})^2$	0.2
3	r_{def_pos}	$-\sum \|\vec{q} - \vec{q_{default}}\|$	0.1
4	r_{dof_acc}	$-\sum \ddot{\vec{q}}$	2e-7
5	$r_{lin\_vel\_z}$	$-v_z^2$	0.5
6	$r_{ang\_vel\_xy}$	$-\sum \vec{\omega_{xy}}^2$	0.2
7	$r_{torques}$	$-\sum \vec{\tau_q}^2$	5e-6
8	$r_{feet\_air\_time}$	$\sum(t_{air} - 0.5)$	5
9	r_{no_fly}	-1 if not c_{l_foot} or c_{r_foot}	0.25

Table B.3 **ANYmal on Wheels (quadruped) shaping rewards:** Combination of shaping rewards from tables 3.3, B.1, and B.2 used to learn a policy from the ANYmal on Wheels in quadruped mode.

#	Reward	Formula	Coefficient
1	r_{face_goal}	$-\|\theta - \theta_{goal}\|$ if $\|\vec{x} - \vec{goal}\| > 0.5m$	0.1
2	$r_{orientation}$	$-\sum(\vec{\gamma_{xy}})^2$	0.2
3	r_{def_pos}	$-\sum \|\vec{q} - \vec{q_{default}}\|$	0.1
4	$r_{dof\_acc\_legs}$	$-\sum \vec{q_{legs}}$	3e-7
5	$r_{dof\_acc\_wheels}$	$-\sum \vec{q_{wheels}}$	1e-7
6	r_{action_rate}	$-\sum \|\vec{a} - \vec{a_{last}}\|^2$	0.02
7	$r_{lin\_vel\_z}$	$-v_z^2$	0.5
8	$r_{ang\_vel\_xy}$	$-\sum \vec{\omega_{xy}}^2$	0.5
9	$r_{torques}$	$-\sum \vec{\tau_q}^2$	5e-6

Table B.4 **ANYmal on Wheels (biped) shaping rewards:** Combination of shaping rewards from tables 3.3, B.1, and B.2 used to learn a policy from the ANYmal on Wheels in biped mode.

#	Reward	Formula	Coefficient
1	r_{face_goal}	$-\|\theta - \theta_{goal}\|$ if $\|\vec{x} - \vec{goal}\| > 0.5m$	0.1
2	$r_{orientation}$	$-\sum(\vec{\gamma_{xy}})^2$	0.2
3	r_{def_pos}	$-\sum \|\vec{q} - \vec{q_{default}}\|$	0.1
4	$r_{dof\_acc\_legs}$	$-\sum \vec{\ddot{q}_{legs}}$	3e-7
5	$r_{dof\_acc\_wheels}$	$-\sum \vec{\ddot{q}_{wheels}}$	1e-7
6	r_{action_rate}	$-\sum \|\vec{a} - \vec{a_{last}}\|^2$	0.02
7	$r_{lin\_vel\_z}$	$-v_z^2$	0.5
8	$r_{ang\_vel\_z}$	$-\omega_z^2$	0.2
9	$r_{ang\_vel\_xy}$	$-\sum \vec{\omega_{xy}}^2$	0.5