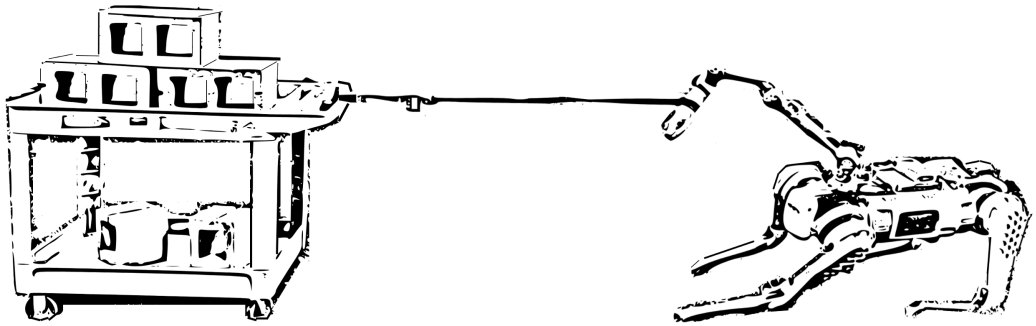
Learning Force Control for Legged Manipulation

Master's project in mobile robotics

From École Polytechnique Fédérale de Lausanne (EPFL) – School of Engineering

Carried out at the Massachusetts Institute of Technology (MIT) – Improbable AI Lab



Tifanny Pereira Portela

Supervisor :
Assistant Prof. Pulkit Agrawal
Head of the Improbable AI Laboratory, MIT

Co-supervisor :
PhD student Gabriel Margolis, MIT

EPFL Mentor Professor :
Prof. Auke Ijspeert
Head of the Biorobotics Laboratory, EPFL

September 22, 2023

Contents

1 Abstract **1**

2 Introduction **2**

2.1 Related works . . . 3

3 Hardware and Networking Configuration **5**

3.1 Unitree B1 robot . . . 5

3.2 Unitree Z1 arm . . . 6

3.3 Oculus: the teleoperation interface . . . 8

3.4 Complete system setup . . . 9

4 Method **12**

4.1 Reinforcement learning . . . 12

4.1.1 Markov Decision Process . . . 13

4.1.2 Proximal Policy Optimization . . . 15

4.2 Policy architecture . . . 16

4.3 Training procedure . . . 17

4.4 Sim-to-real transitions: tackling hardware components individually . . . 19

4.4.1 Z1 Arm: policy and training design . . . 19

4.4.2 B1 robot: policy and training design . . . 23

4.5 Whole-body controllers: policies and training designs . . . 25

4.5.1 End-effector positioning and locomotion controller . . . 26

4.5.2 End-effector force controller . . . 28

5 Results **32**

5.1 Z1 arm . . . 32

5.1.1 Z1 arm: sim-to-real . . . 32

5.1.2 Z1 arm: end-effector position controller . . . 40

5.2 B1 robot . . . 42

5.2.1 B1 robot: sim-to-real . . . 42

5.2.2 B1 robot: stand-up controller . . . 42

5.3 Whole-body position controller . . . 44

5.3.1 Simulation results . . . 44

5.3.2 Hardware deployment . . . 48

5.4 Whole-body force controller . . . 50

5.4.1 Simulation results . . . 51

5.4.2 Hardware deployment . . . 51

6 Discussion **54**

6.1 Future work 54

7 Appendix **56**

7.1 Z1 arm: position controller 56

7.2 B1 robot: stand-up controller 57

7.3 Whole body position controller 58

7.3.1 Training curves 58

7.3.2 Tracking performance in simulation 59

7.3.3 Tracking performance on hardware 59

7.4 Whole-body force controller 60

8 Acknowledgements **62**

1 Abstract

Controlling the contact force during interactions is an inherent requirement for locomotion and manipulation tasks. Current reinforcement learning approaches to locomotion and manipulation rely implicitly on forceful interaction to accomplish tasks but do not explicitly regulate it. This work proposes a reinforcement learning task specification that focuses on matching desired contact force levels. Integrating force control with the coordination of a robot's body and arm, an end-to-end policy for legged manipulator control is presented. Force control enables to realize compliant gripper and whole-body pulling movements that have not been previously demonstrated using a learned policy. It also facilitates a characterization of the force tracking performance of learned policies in simulation and the real world, indicating their performance potential for force critical tasks.

2 Introduction

Forceful manipulation using legged mobile platforms holds a great promise for a range of applications. From assisting the elderly or individuals with disabilities, to aiding in industrial tasks such as digging, construction, or pushing heavy objects for humans, the breadth of applications of a legged platform capable of forceful manipulation is vast. These systems could be especially invaluable in environments where human presence is limited or dangerous. Recent research [1, 2] highlight the demand for automation in sectors such as construction and industrial inspection, given the stagnant productivity, lack of skilled labor, and high fatal injury rates, with construction ranking as one of the most dangerous industries [3].

Legged manipulators, taking inspiration from nature's most versatile mobile manipulator, the human body, present a promising solution towards integrating robotic systems in our complex world. Embracing characteristics from both fixed-base robotic manipulators and mobile platforms, these systems extend a traditional manipulator's workspace, while enabling environment interaction for a mobile platform. Compared to wheeled or tracked robots, legged platforms have proven to be superior in navigating rough terrains [4, 5]. Additionally, having legs facilitates adopting the most favorable posture for high-force tasks. Drawing inspiration from human biomechanics, when confronted with force-intensive tasks, humans instinctively employ a whole-body approach: they use their legs for anchorage, their core for stability, and their arms to facilitate the task execution [6]. Legged manipulators, similarly, can adopt this approach, leveraging their entire structure for optimal force application and stability. To ensure stability during high-force interactions and prevent potential falls, employing a strong and robust hardware platform is a necessity. Contemporary lightweight robots, popular in the research community, lack the strength to perform these tasks. Consequently, in this study, the chosen platform is the B1 robot and Z1 arm from Unitree.

Integrating forceful manipulation with legged locomotion not only involves advanced control strategies but also requires coordination between the locomotion and manipulation subsystems. The approach proposed by Zipeng Fu et al [7]. enables whole-body control of a legged manipulator using reinforcement learning. Their method not only coordinates the arm and legs effectively, but also enhances the capabilities of the arm, by adjusting the legs to extend the reach of the end-effector. Their method is a foundational work that effectively demonstrates reinforcement learning's applicability to solve whole-body tasks, and opens a vast potential to expand the horizons of what legged manipulators can achieve. This motivates revisiting force control obtained by coordinating the body and arm of legged manipulators using learning methods, in the hope of performing force-controlled tasks under challenging conditions where reinforcement learning excels. This includes operation in challenging conditions like in-the-wild terrains, and utilizing a single neural network policy allowing computationally efficient accommodation of the full system dynamics.

Recent advancements in reinforcement learning have led to state-of-the-art robotic control in sectors of locomotion, manipulation, and drone flight [8, 9, 10, 11]. When dealing with large and robust robots, it is crucial to regulate the force exerted to their environment to ensure safe and controlled interaction. Yet, many existing learning-based approaches of loco-manipulation successful in posture control, object manipulation, and fall recovery [7, 12, 13] do not explicitly regulate the amount of applied force. In contrast to other forceful manipulation systems [14, 15, 16], such learned strategies usually rely on a coarse action space of position targets for a low-gain PD controller, a relatively low control frequency, and do not incorporate a force-torque sensor on the contact body. Nevertheless, reinforcement learning controllers are often deployed to generate forceful interactions, including walking, running, parkour, fall recovery, object reorientation, and others.

Training stability and performance are challenging in reinforcement learning. Explicit estimation of helpful attributes in learning a task has proven to significantly enhance both performance and learning stability. For instance, the study by [17] demonstrated that foot height estimation was crucial for better locomotion results. In the context of this work, estimating the actual force at the end-effector could be advantageous for the training process.

The force applied at the end-effector of a robotic arm can be estimated either from dedicated force sensors or from observers. The former is computationally costly and require knowledge of the robot dynamics [18]. The latter, while offering direct measurements, brings its own set of challenges. The physical and electrical integration of force sensors, particularly in grippers, can be a complex task. Additionally, they can undergo wear and tear, necessitating recalibration or replacement overtime [19]. Another approach involves explicitly estimating it using a trained neural network and proprioceptive state history, as demonstrated by [17] in estimating base linear velocity, foot height, and contact probability. In this framework, the learning-based state estimation network is concurrently trained with the policy network. Given the effectiveness of this method, the present research follows a similar approach for estimating the force exerted at the end-effector.

In this study, a task specification is introduced for reinforcement learning that directly commands the desired amount of contact force. With this specification, a policy is trained to apply the intended force using a manipulator attached on a large quadruped. This research facilitates a legged manipulator in performing tasks involving force control. It can transition from a compliant mode for kinesthetic demonstrations and safe operation around humans to high-force activities such as pulling and lifting. These behaviors are achieved by coordinating the entire body through an end-to-end policy. Additionally, in order to apply forces, the robot must first navigate to and grasp objects. To address this prerequisite, an end-effector positioning and locomotion controller is developed with a task specification inspired from prior work [7].

This thesis presents several advancements to the domain of legged manipulator control. Firstly, it showcases the first deployment of learning-based whole-body force application control in legged manipulators. Then, it evaluates the effectiveness of force tracking and force estimation for learned policies in simulation and reality. Finally, the ability of the quadruped to coordinate the body and legs to increase both its reach and the applied force magnitude compared to using the manipulator in isolation, is also investigated.

2.1 Related works

Previous research has controlled a quadruped with a mounted arm through sim-to-real reinforcement learning, defining the task as simultaneous control of the end effector position and locomotion velocity [7, 20]. The method adopted in this study is highly inspired by the methodology introduced by Zipeng Fu et al. [7]. Their method coordinates the arm and legs to extend the reach of the end-effector. Although their approach effectively demonstrates reinforcement learning's applicability to whole-body positioning, it is not suitable for forceful and compliant tasks. This limitation arises from the lack of force modeling during training, the competing constraint arising from explicitly commanding the gripper position relative to the body, and the smaller arm with a maximum payload rating of 0.2kg. Other works have developed policies for specific tasks that require forceful interaction, such as fall prevention and recovery using a mounted arm [13] or foot dribbling [12, 21]. While these works establish the feasibility of forceful interactions, they specialize for single tasks rather than defining an interface that can be used to teleoperate forceful tasks. Without a dedicated way to command the interaction force, these controllers cannot be repurposed for other force application tasks, and the precision of the force control cannot be directly evaluated for both simulated and real-world scenarios.

Another research group has employed a hierarchical architecture enabling robots to maneuver large objects using solely their body, without an arm [22]. Although the robot's body and legs alone are sufficient to perform some environment interactions, integrating an arm can augment the workspace and allow the robot to modulate both the direction and intensity of the force application more precisely.

For certain tasks that involve contact interactions, directly commanding position setpoints is challenging, due to the unknown geometry of the contact surface or the need for a specific degree of compliance in the motion. Instead, several control parameterizations have been introduced to regulate contact force. Classic work on compliance and force control [23, 24, 25, 26] introduced concepts like impedance and hybrid force-velocity control. These methods aim to define how a robot should balance between force and position tracking errors given that these quantities cannot be independently controlled during contact. Force control techniques can either be implemented with closed-loop feedback from a wrist force sensor or, with less precision, by estimating the contact force from proprioceptive actuators and the robot model [27]. One practical application of force control is kinesthetic teaching. In this method, a human physically guides a robot to demonstrate the movements and forces it should apply on the environment to accomplish a specific task [28, 29]. To receive a kinesthetic demonstration, it is necessary for the robot to comply with the operator's guidance and also estimate the amount of force exerted. This method of teaching has previously been considered challenging for legged robots and high-dimensional systems because of the cognitive challenge of reasoning about many degrees of freedom [30], which provides a motivation for developing low-level controllers that assume some control authority and ease the task.

Various efficient model-based methods have been developed for controlling legged manipulators across diverse tasks [31, 32, 33, 34, 35, 36, 14, 37, 38, 39]. These methodologies typically optimize a whole-body inverse dynamics model to ensure the stability of a reference trajectory determined through model-predictive control or trajectory optimization techniques. Some research has specifically focused on forceful interactions using legged manipulators [14, 36, 37, 38]. Early approaches [36] utilized trajectory optimization to generate open-loop behaviors for manipulating heavy objects, complemented by a distinct online tracking controller for motion execution. Subsequent work [14] integrated both contact point planning and control to manipulate objects using the entire body of a humanoid robot. In a more contemporary study, ALMA [38] introduced a motion planning and control framework capable of coordinating dynamic and compliant locomotion and mobile manipulation using a whole-body control task hierarchy. This system showcased precise force control at the end effector using the whole body. The ALMA system also exhibited compliant behavior in service of a collaborative payload-carrying task.

3 Hardware and Networking Configuration

The hardware employed in this thesis consists of the B1 legged robot and the Z1 arm, both from Unitree, forming a system that is teleoperated using the Oculus Quest 2 set provided by Meta, as depicted in Figure 1. In the rest of this section, the hardware and communication setup will be described, focusing on the technical specifications of these components.

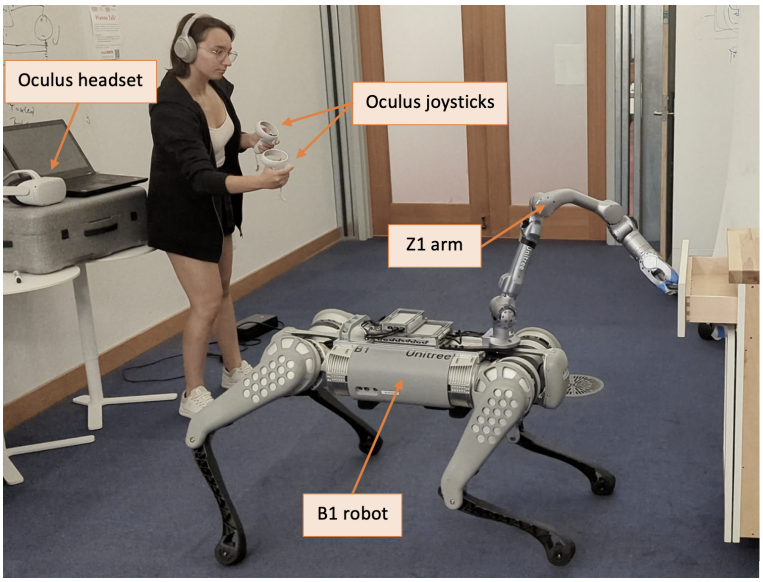


Figure 1: B1 legged robot and Z1 arm from Unitree teleoperated using the Oculus Quest 2, constituted of a headset and 2 joysticks.

3.1 Unitree B1 robot

This legged robot is equipped with three motors in each leg, corresponding to the hip, thigh, and calf. These motors can be controlled through position, speed, and torque commands. This robot comes equipped with an Inertial Measurement Unit (IMU) and joint encoders. Table 1 provides the range of motion limits for the leg joints, along with their corresponding maximum torque values.

Joint	Range of motion $[q_l^{min}, q_l^{max}]$ [deg]	Maximum torque τ_l^{max} [Nm]
Hip joints	$[-60, 60]$	140
Thigh joints	$[-38, 170]$	140
Calf joints	$[-156, -48]$	210

Table 1: Range of motion and maximum torques of the B1 robot.

Regarding computational power, the integrated computer system, built on the X86 architecture, operates on an Ubuntu Linux operating system and supports real-time communication frequency of up to 1 KHz. Additionally, the system is equipped with 3 NVIDIA Jetson NX boards.

In terms of communication capabilities, the robot establishes a wireless network segment defined by the IP range 192.168.123.xxx. This configuration facilitates remote connections by users to the robot's operating system through SSH. Within this designated network, the X86 CPU is allocated an IP address of 192.168.123.220, while the Jetson Xavier NX boards are correspondingly assigned the IP addresses 192.168.123.23, 192.168.123.24, and 192.168.123.25.

Unitree provides a Software Development Kit (SDK) to control the motors of the legged robot. Upon initiating the system's booting process, the user has the option to select high-level control or low-level control, but is unable to operate both modes simultaneously.

In the preparation of the robot for operation, two critical steps must be followed:

1. Prior to powering on the robot, the battery pack must be securely installed into the designated battery slot, accessible from the side of the robot as shown in Figure 2. It is essential to securely fasten the buckle when installing the battery, as failure to do so may cause the battery to become dislodged during movement. When the battery is fully charged, it will furnish the robot with approximately 3 hours of operational lifetime.

2. Before initiating the startup sequence, the robot should be placed on a flat surface with the belly support pad lying flush against the ground, and the legs retracted as depicted in Figure 4. Both the feet and knees must be in contact with the ground. This positioning is vital to aligning the theoretical zero position defined by Unitree's developers with the robot's actual mechanical zero position. Any misalignment between these two values could adversely affect the robot's proper functioning.

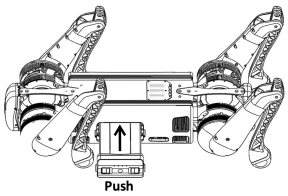


Figure 2: Battery insertion [40]

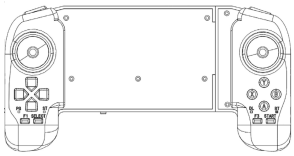


Figure 3: Robot remote controller [40]

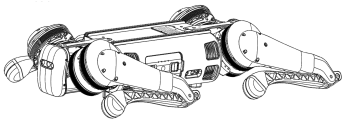


Figure 4: Zero position of the legged robot [40]

Concluding the pre-boot preparation phase, the battery can be activated through a two-step process. First, the power switch is pressed once, then it is held down for more than 4 seconds. Within 2 minutes of the battery's activation, the robot assumes a standing position under the high-level control mode. To transition the robot into damping mode, a specific control sequence on the remote controller must be followed, as illustrated in Figure 3. The sequence requires pressing the combinations [L2+A], [L2+A], [L2+B], and then concluding with [L1+L2+START]. Following this command sequence, the robot will take a seated position on the ground, allowing the joints to move freely. This process ensures a smooth transition between operational states. Once in this state, the robot is ready to be controlled in low-level mode using the algorithms developed in this project.

3.2 Unitree Z1 arm

The Z1 arm developed by Unitree originally possesses 6 degrees of freedom. These joints are depicted in Figure 6 (J1 to J6). Unitree's interface enables seamless integration of diverse end-effectors. Within this project, the selected end-effector is a gripper, which augments the robotic arm's capabilities by introducing an additional degree of freedom (J7 in Figure 6).

Table 2 exposes the range of motion for the arm's joints and their corresponding maximum torques, resulting in a maximum arm span of 730 mm. For a clearer visualization, refer to figure 7 depicting the arm's range of motion.

The robotic arm features three communication ports: one dedicated to the power supply and two for network interfaces. The arm's main control board is equipped with a primary and an

Joint	Range of motion $[q_a^{min}, q_a^{max}]$[deg]	Maximum torque τ_a^{max} [Nm]
Joint 1 (J1)	$[-150, 150]$	33
Joint 2 (J2)	$[0, 180]$	66
Joint 3 (J3)	$[-165, 0]$	33
Joint 4 (J4)	$[-80, 80]$	33
Joint 5 (J5)	$[-85, 85]$	33
Joint 6 (J6)	$[-160, 160]$	33
Joint 7 (J7)	$[-90, 0]$	33

Table 2: Range of motion and maximum torques of the Z1 arm [41].

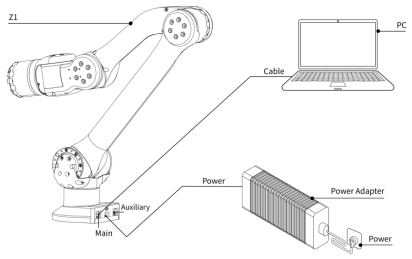


Figure 5: Cable connection of the Z1 arm [41]

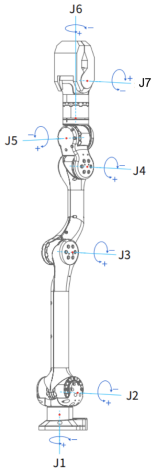


Figure 6: Degrees of freedom of the Z1 arm [41]

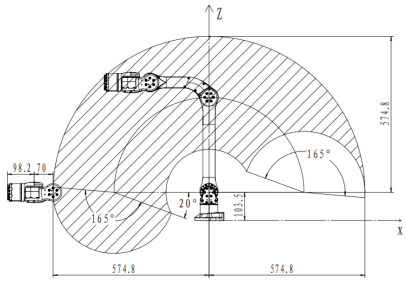


Figure 7: Range of motion of the Z1 arm [41]

auxiliary network port, with the main network port operating at a control frequency of 300Hz, and the auxiliary network port at 100Hz. Both of these Ethernet ports serve as the exclusive means of communication with the arm. In this project, the main port is employed, and the default IP address for SSH access is set to 192.168.123.110. The cable layout is shown in Figure 5.

Compared to other arms on the market, this hardware is notably heavy, with a weight of 4.5kg. This substantial weight enhances its payload capacity, allowing for more versatile applications when compared to arms like the WidowX 200 Robot Arm, which can only handle 200g. In this project, the Pro version of this arm is used allowing an impressive payload range of 3-5kg. This added weight contributes to improved arm stability by reducing vibrations and oscillations during operations. However, it comes with the drawback of increased power consumption, which can be a concern for mobile robotics applications with limited computational power. Additionally, the heavier arm may experience slower accelerations, potentially affecting overall efficiency. Nonetheless, the weight is essential for this project's goal of accomplishing forceful tasks, making this specific hardware choice necessary for its intended purposes.

A crucial point to consider is the necessity of positioning the arm in its zero position, as depicted in Figure 8, before powering it on. This crucial step ensures alignment between the theoretical zero position defined by Unitree's developers in the SDK and the actual mechanical zero position. Failure to do so may result in the low-level controller being misaligned, causing potential operational issues.

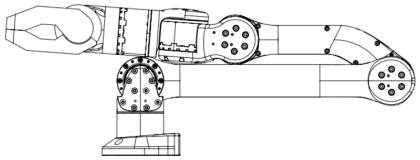


Figure 8: Zero position of the Z1 arm [41]

3.3 Oculus: the teleoperation interface

In this project, the Oculus Meta Quest 2 Virtual Reality (VR) headset from Meta is used to teleoperate the robot with an attached arm.

This VR headset comes with two controllers, one for each hand of the user. However, it's important to note that the visual aspect of the product is not employed in this project. Instead, only specific features, including the buttons and joysticks of both controllers, as well as the position tracking of the right controller are used to control the legged mobile manipulator.

In this setup, the VR headset remains static and is placed on a table. The position of the right controller in relation to the goggles defines the end-effector target position in the arm frame. This process is visually represented in Figure 10, where one can envision the arm's movement synchronized with that of the right controller.

Figure 9 illustrates the buttons utilized to define the inputs of the end-effector positioning and locomotion controller defined in section 4.5.1. The gripper opening angle can be controlled using the Trigger A and Button A (highlighted in red in Figure 9). By pressing Trigger A, the gripper incrementally opens or closes by 10 degrees. On the other hand, Button A determines the direction of the angle increment, whether the gripper opens or closes. Similarly, the control of Joint 6 is achieved through the use of Trigger and Button B (highlighted in green in Figure 9).

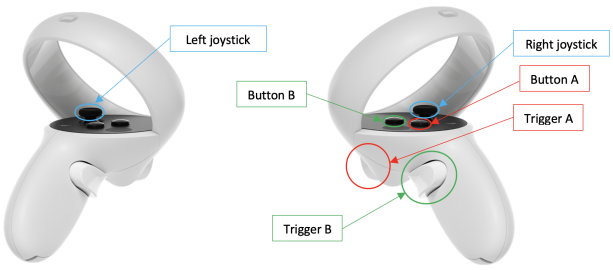


Figure 9: Oculus controller button description

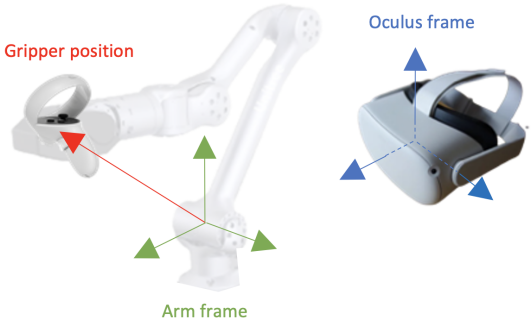


Figure 10: Oculus controller to gripper position mapping

The left and right joysticks (highlighted in blue in Figure 9) are used to define the base velocity of the B1 legged robot. More specifically, the left joystick defines the base linear velocities along the x (v_x) and y (v_y) axes and the right joystick controls the angular velocity around the z axis (v_{yaw}). Figure 11 illustrates this control system.

Note that, when teleoperating the system under the whole-body force controller, only the left and right joysticks are used and they define the end-effector force command.

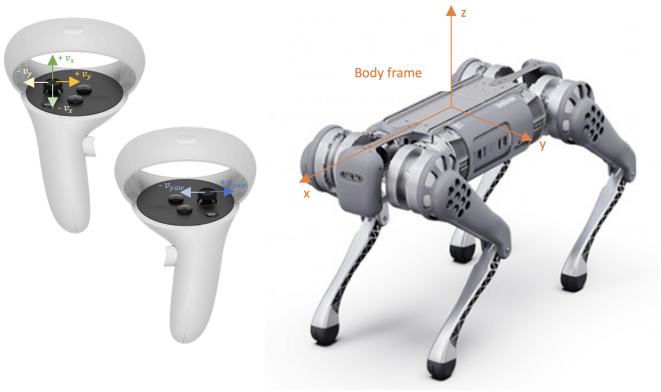


Figure 11: Base velocity commands mapping

3.4 Complete system setup

This section outlines the complete system setup, integrating the three hardware components discussed previously in this chapter.

Figure 12 illustrates the arrangement and networking configuration of the complete system. The middleware framework chosen for communication between computers is the Lightweight Communications and Marshalling (LCM) library. LCM supports multiple programming languages through data marshalling, making it suitable for this project where Python and C++ are used. It employs a publish-subscribe model for node communication, similar to the Robotic Operating System (ROS), but with low-latency and efficient data communication, making it more appropriate for real-time applications.

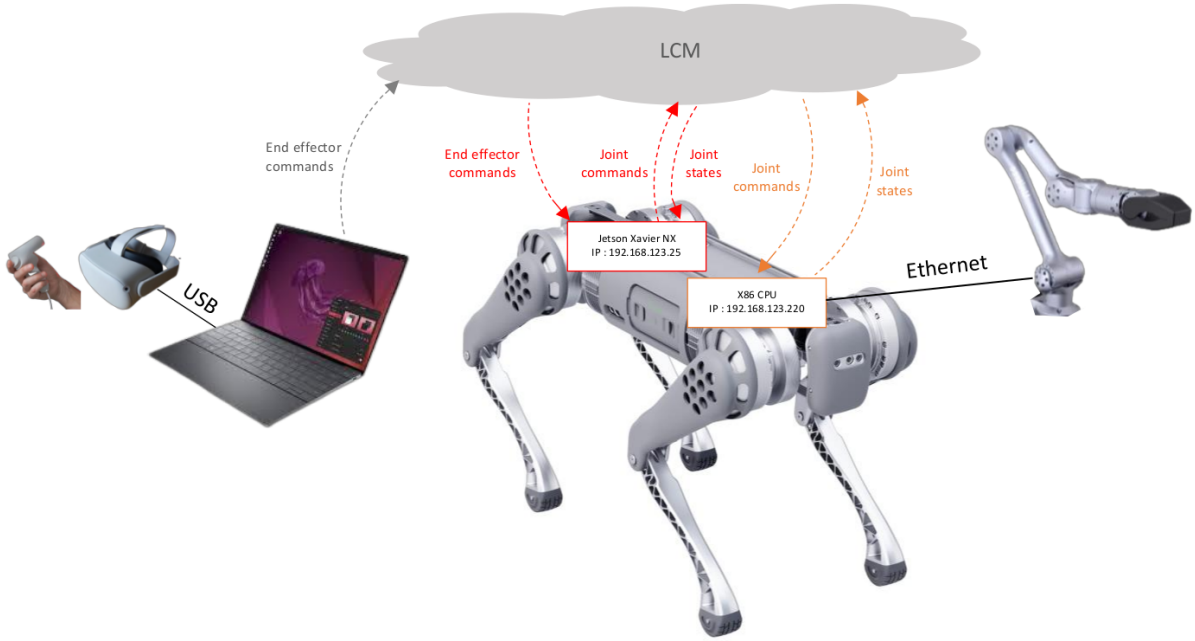


Figure 12: Complete system setup and networking configuration.

Communication with the robotic arm (X86 CPU)

The robotic arm is physically connected to the X86 integrated computer system using an ethernet cable. This computer also provides the necessary power supply for the arm's operation. Upon powering on the robot, the Z1 arm SDK, enabling communication with the robotic arm controller, runs automatically on the X86 CPU. To send commands to the arm, the joystick process (z1_joystick) needs to be manually removed from the autostart scripts. This is achieved by navigating to Unitree/autostart/z1/z1.sh on the X86 CPU and commenting out the line "sudo ./keep_arm_joystick_alive.sh", since this script continuously keeps the z1_joystick process alive, even after manually killing it.

To enable arm state reading and control, a C++ node has been established in the X86 CPU. This program takes inspiration from the examples found in the z1_sdk library developed by the Unitree developers. This node communicates via LCM with the main algorithm running on the Jetson NX computer. More specifically, it receives the arm joints state from the robotic arm controller and broadcasts it over LCM, while also receiving arm commands through LCM from the main algorithm and forwarding them to the robotic arm controller. In order to activate this C++ node, one has to run the following commands :

```
ssh unitree@192.168.123.220
cd b1_deployment/z1_sdk/build
./lowcmd_development
```

Communication with the robot motors (Jetson NX)

For reading and controlling the robot motors, another C++ communication node has been developed on the main Jetson NX. This program also communicates via LCM with the main algorithm and draws inspiration from the examples provided by the Unitree developers in the unitree_legged_sdk library. To activate this C++ node, the following commands need to be executed:

```
ssh unitree@192.168.123.25
cd b1_deployment/unitree_legged_sdk/build
./lcm_position_b1
```

Teleoperation with the Oculus headset (Laptop)

To enable teleoperation of the robot from a distance, the Oculus headset cannot be physically connected via a cable to the robot's onboard computer. Instead, a Python node has been created to receive data from the Oculus headset through a USB cable connected to a laptop and then transmit this data via LCM. This code is based on the repository developed by the Robotic AI and Learning Lab at Berkeley University (RAIL), which provides the necessary interface to read the position and pressed button information from the Oculus Quest device. This Python node can be activated by running the following commands:

```
cd walk_these_ways/laptop_oculus
python oculus_node.py
```

Network configuration

For this system to function properly and before launching the C++ nodes, it is essential to configure the network interfaces on all operating systems within the framework. The following commands should be executed on all computers (Jetson NX (IP address: 192.168.123.25), x86 CPU (IP address: 192.168.123.220) and laptop):

```
sudo ifconfig eth0 multicast
sudo route add -net 224.0.0.0 netmask 240.0.0.0 dev eth0
```

The first command configures the eth0 interface to support multicast traffic. This type of communication enables data to be sent from one or more points to a set of other points. The second command adds a route for the network 224.0.0.0 with a netmask of 240.0.0.0, specifying that the traffic for this network should be routed through the eth0 device.

Main algorithm in a docker environment (Jetson NX)

Finally, the main algorithm, serving as the central control for the system, is executed within a Docker image. This Dockerized environment facilitates rapid prototyping and deployment, while ensuring consistency and isolation of the algorithm's execution. This central node processes inputs from the robot and arm states, along with Oculus commands received via LCM, to compute arm and leg motor actions. These commands are then convey back to the arm and leg controllers through LCM. To launch the main algorithm, the following commands should be executed:

```
ssh unitree@192.168.123.25
cd go1_gym/go1_gym_deploy/docker
sudo make autostart
```

4 Method

As depicted in Figure 13, the system proposed in this work is comprised of two types of learned policies developed using reinforcement learning. The primary focus of this research is the force controller, detailed in section 4.5.2, which enables compliant and force-controlled tasks. In order to apply forces, the robot first needs to walk to and grasp objects. To address this prerequisite, an end-effector positioning and locomotion controller is trained with a task specification inspired from prior work [7], as outlined in section 4.5.1. Each controller shares an identical policy architecture, action and observation space, which are elaborated upon in sections 4.2 and 4.5.

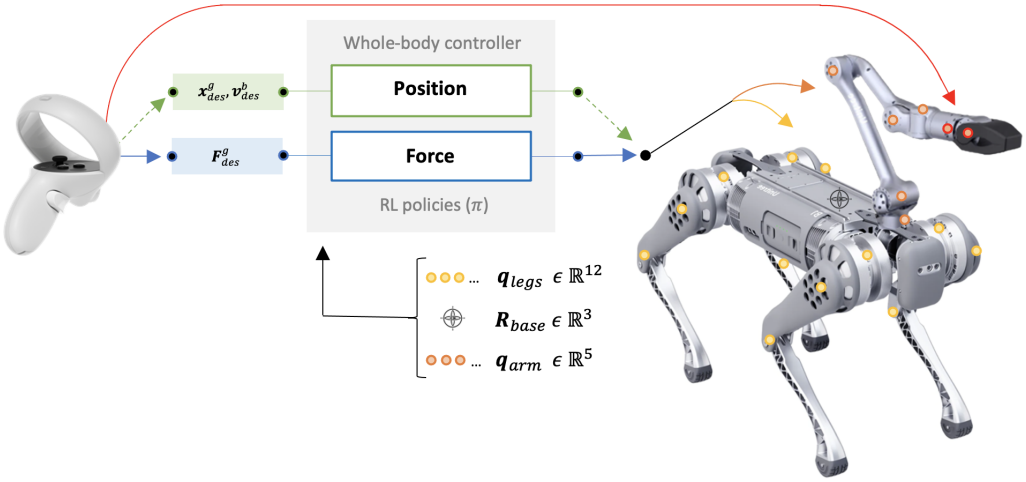


Figure 13: System diagram depicting the command flow initiated by the Oculus controller. Two modes of learned control are implemented: a base velocity $\mathbf{v}^b_{des}$ and end-effector position $\mathbf{x}^g_{des}$ controller (green), and an end-effector force $\mathbf{F}^g_{des}$ controller (blue) for a robot with consistent foot-ground contact. While Oculus directly modulates the last two arm joints, the learned policy handles the 12 leg joints and the first five joints of the arm.

In this chapter, a concise introduction to reinforcement learning and the specific algorithm used in this thesis is described in section 4.1. Then, the shared policy architecture is delineated in section 4.2. Section 4.3 discusses the common training procedure for all policies within this project. The approach used to bridge the simulation to reality gap is presented in section 4.4. Finally, the chapter concludes with the description of the policies and training designs for the aforementioned two whole-body controllers in section 4.5.

4.1 Reinforcement learning

Learning by interaction is a common learning approach observed in nature. To acquire various skills, many living species gain knowledge directly by interacting with their environment. Consider human babies when they start their journey to walk, for instance. They don't have an explicit teacher guiding them each step of the process. Instead, they acquire this new skill by trial and error. After each fall, they refine their technique based on the feedback from the environment. Similarly, when training a dog to sit, the animal does not know the human's expectation and starts experimenting random actions, such as shifting its paws or standing still. Once it achieves the desired behavior, it receives a reward, like a treat or an affectionate pat, helping the dog understand the right action to the commands. Finally, through repetition and consistent positive reinforcement, the dog associates the instruction with the desired action.

Reinforcement Learning (RL) is a computational approach to goal-directed learning from interaction [42]. In this machine learning paradigm, the entity learning and making decisions is called the *agent*, while everything external to this agent is referred to as the *environment*. These two entities interact continuously, the agent selects actions affecting the environment, which responds by presenting new situations to the agent based on its choices. Specifically, an agent learns an optimal policy from its own experience, mapping observations to actions and maximizing a reward signal. During the training phase, the agent simultaneously collects data about the goal task through interaction with the environment and optimizes its policy.

A reinforcement learning system is composed of four elements:

1. A *policy*, mapping states of the environment to actions. These policies are typically stochastic, associating states with probabilistic actions. Introducing stochasticity into the policy offers several benefits, such as helping the agent escape local minima and promoting exploration, potentially leading to the discovery of better strategies.

2. A *reward* signal defining the goal task. At each time step, this signal consists of a single numerical value, typically derived from the weighted average of several reward components. The objective of the agent is to maximize this signal.

3. A *value function* specifying the long-term desirability of states. While rewards indicate the immediate desirability of a state, the value determine the expected reward the agent foresees to accumulate from its current state into the future. The policy is optimized based on the reward predictions provided by the value function, making accurate value estimation crucial.

4. A *model* that determines the environment's transition dynamics, predicting the next state based on the current state and action. This fourth element being optional, some methods employ it, like model-based approaches, whereas others, namely model-free methods, do not.

   Model-based approaches derive optimal policies based on a provided or learned forward model of the robot's transition dynamics. Leveraging such a model significantly diminishes the number of interactions required between the robot and the environment, leading to a faster convergence to the optimal solution. Numerous researchers successfully applied model-based techniques on robotic manipulation tasks, where the model learned guides exploration, improving data efficiency. [43, 44]

   However, the performance of model-based RL algorithms is intrinsically linked to the accuracy of their transition model [45]. Obtaining accurate models is a challenge, as they are rarely available, difficult to learn from data, and any inaccuracies can introduce biases in the learning process. Given this complexity, the scientific community has increasingly turned its focus towards model-free methods, especially for challenging environments [46]. Indeed, such methods are able to take advantage of complex dynamics without explicitly modeling them, such as successfully pouring water without understanding fluid dynamics. For the aforementioned reasons, in this thesis, model-free reinforcement learning is employed.

   In the rest of this chapter, there will be a concise theoretical overview of RL, accompanied by a discussion on algorithms pertinent to this projet.

4.1.1 Markov Decision Process

Every reinforcement learning agent must be able to sense the state of its environment, take actions affecting this state and have explicit goals tied to this state. The agent's objective is

to maximize the reward gained over a series of interactions with the environment. In robotics, this interaction duration consists of a predetermined number of iterations, referred to as the episode length, or it may be shorter if a terminal state is reached (e.g exceeding torque limits).

Specifically, at every time step, t, within an episode, the agent senses the environment's state, $S_t \in \mathbb{S}$, and takes a state-dependent action $A_t \in \mathbb{A}(S_t)$. This action leads the agent to a new state S_{t+1} for which it receives a reward, $R_{t+1} \in \mathbb{R}$. After each iteration, the agent establishes a mapping from states to likelihood of choosing each possible action available from these states. This correlation is termed the agent's *policy*, represented as π_t , where $\pi_t(a|s)$ represents the probability that $A_t = a$ when $S_t = s$. Figure 14 illustrates the basic interaction loop between the environment and the agent.

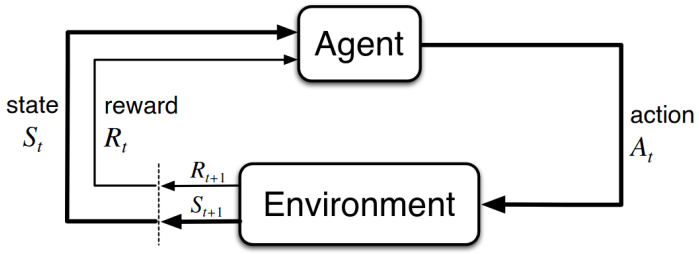


Figure 14: Diagram of the interaction between the agent and the environment [42].

The environment's state, $S_t \in \mathbb{S}$, contains the current understanding of the agent about its environment. This comprehension might encompass not just immediate sensory inputs, but can also reflect past interactions. The goal is to select a state that captures all important information from the past in a concise manner to predict the future effectively. A state signal retaining all relevant information is said to follow the *Markov* property.

A reinforcement learning problem, for which the states have the Markov property, is called a Markov Decision Process (MDP). This property assumes that each state is only dependent on the previous state. In this case, the probability density of the next state $s' \in \mathbb{S}$ and reward, $r \in \mathbb{R}$, can be defined using only the current state, $s \in \mathbb{S}$, and action $a \in \mathbb{A}$. The dynamics of the MDP are completely defined by these quantities.

$$p(s', r|s, a) = \Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{1}$$

In the context of a MDP, the expected cumulative reward, $G_t \in \mathbb{R}$ is an important concept, informing about how well the agent is expected to perform in the future. This accumulated reward is computed based on the immediate reward, $r(s_t, a_t)$, and future rewards over a fixed time horizon T. The importance of future rewards are weighted by the discount factor, γ, which balances the significance of immediate versus future rewards.

$$G_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i), \quad \gamma \in [0, 1] \tag{2}$$

The expected cumulative reward is used to define the value function, denoted by $V^\pi(s)$, expressing the expected return when following a policy π. This metric gives an estimation of how good it is for an agent to be in a state if it continues acting according to policy π. Mathematically, it is represented as:

$$V^\pi(s) = \mathbb{E}_\pi \left[G_t|S_t = s\right] \tag{3}$$

While the value function quantifies the expected return from a given state, it does not inform about which action to take from that state. The action-value function, denoted as $Q^\pi(s,a)$, provides an estimate of the expected return when an agent is in state s, takes an action a and then adheres to policy π for all future actions.

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right] \tag{4}$$

The agent's main objective is to identify the policy maximizing the action-value function across all states and actions.

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{5}$$

4.1.2 Proximal Policy Optimization

RL algorithms can be classified into two distinct categories based on the way data is collected during training. On-policy algorithms only use data collected by the current policy being optimized, while off-policy methods leverage data from the current and previous versions of the policy. Compared to their counterpart, off-policy algorithms are usually more sample efficient. However, despite their low sample efficiency, on-policy methods are known to provide better convergence during learning. [47]

Building on this classification, RL techniques can be further classified as:

- **Value-based** methods estimate the optimal action-value function $Q^*(s,a)$ first and then derive the corresponding optimal policy π^* by choosing the action maximizing $Q^*(s,a)$. Value-based methods often utilize off-policy algorithms, making them sample efficient. However, this type of algorithms are incompatible with continuous action spaces, making them unsuitable to solve robotic tasks.
- **Policy-based** methods directly optimize the policy, parameterized by θ, which usually represents the weights and biases of a neural network. These weights are usually optimized through gradient descent. This type of methods can handle continuous action spaces, but usually experience a slower convergence rate compared to value-based methods. [42]

Actor-Critic algorithms are a combination of value and policy-based methods. [42] Indeed, they optimize a parameterized policy, the *actor* network, while estimating the value function by learning the *critic* network. The critic network evaluates the present strategy that the actor is employing making Actor-Critic methods on-policy. This type of method combine the best characteristics of both approaches, namely sample efficiency and continuous action space. Such attributes have propelled actor-critic methods to achieve state of the art performance among other RL approaches in mobile robotics. [4, 7]

Among Actor-Critic methods, Proximal Policy Optimization (PPO) possesses an architecture that improves the training stability by avoiding too large policy updates. [48] The idea behind PPO is to constrain the policy update using the clipped surrogate objective function, $L^{CLIP}(\theta)$:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{6}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta,\text{old}}(a_t|s_t)}$ is the probability ratio between the current and previous policy, highlighting the divergence between these, and $\hat{A}_t = Q_t^\pi(s,a) - V_t^\pi(s)$ represents the estimated

advantage function. The role of the clip function is to prevent the probability ratio from deviating too much from 1, forcing the updated policy to be similar to the old policy.

This cautious approach to policy optimization, ensuring stability during learning, turned PPO into a popular and very effective algorithm in many RL scenarios. PPO has become a preferred policy optimization method for many robotics applications, as demonstrated by prior studies [12, 13, 49]. The consistent success of this method in previous works constitute the main rationale for its selection in this study.

4.2 Policy architecture

In this chapter, a description of the overall control structure including the training framework is provided. This structure draws significant inspiration from the Concurrent State Estimation architecture proposed by M. Ji et al. [17]. As represented in Figure 15, the proposed structure comprises three different neural networks, namely the *Actor*, *Critic* and *Estimator*.

The primary function of the *Estimator* network is to derive several estimates of unobservable variables based on a 30-step history of observations, $o_{t-30dt}, .., o_{t-dt}$ ($dt = 0.02$ seconds). Structurally, this network is a simple two layers Multilayer Perceptron (MLP), consisting of hidden layer with sizes 256 and 128, and employing Exponential Linear Unit (ELU) activation functions.

These state estimates are then input to the *Actor* network which computes actuator actions, denoted by a^t. These actions are scaled and added to the nominal joint positions, as shown in Equation 7, producing position targets, q_{target}, that are then tracked using a Proportional-Derivative (PD) controller.

$$q_{target} = q_{def} + \sigma_a a^t, \qquad \sigma_a = 0.25 \tag{7}$$

Rather than directly determining the target joint angles, this strategy allows to deal with normalized actions with standard deviation close to 1.0 and mean 0.0. This approach is known to help the learning algorithm converge faster to an optimal solution and to lead to stable learning and control [17]. Furthermore, generating a perturbation around the default configuration has proven to result in a good initial policy. [13] In this project, joint-space control is selected over operational space control. This method has proven to help the policy avoid self-collision and reduce the simulation-to-reality gap, as highlighted in prior bimanual manipulation research. [50]

Meanwhile, the *Critic* network aids learning by estimating the quality of the actions taken by the agent. While the *Actor* and *Critic* are distinct networks, they share the same architectural design. They are both constructed as three-layered MLPs, with hidden layer sizes 512, 256 and 128, using ELU activations.

PPO is used to train the *Actor* and *Critic* networks, while the *Estimator* is trained via supervised learning leveraging privileged information, I_{priv}, from the Isaac Gym simulator. Privileged information is known to improve PPO's performance (at least for locomotion tasks), by stabilizing the learning process and helping the agents learn faster as it guides the learning towards useful representations for the task to solve.[17, 51]

The overall training process goes as follows: first, a series of trajectories is collected in Isaac Gym and then the three networks are updated based on their respective loss functions. This process is repeated until the performance metrics stabilize. This parallel training ensures that the policy network is able to handle unstable estimations from the *Estimator* network.

Note that every policy designed in this thesis follows this identical structure, while the type of the privileged information, observations and actions varies upon the task.

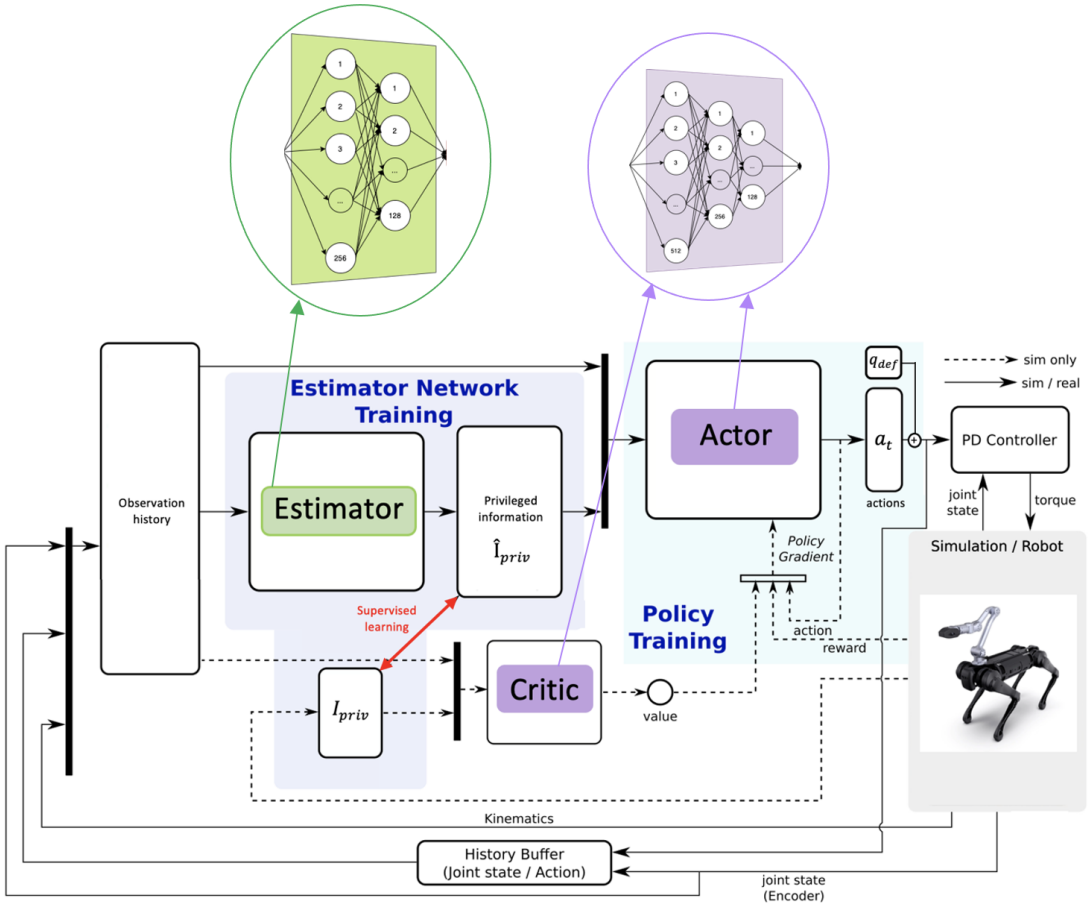


Figure 15: Overall control diagram comprising the training framework. Diagram inspired by [17]

4.3 Training procedure

NVIDIA Isaac Gym [52] is the platform used to collect simulated data to train all policies. Every policy is trained on flat terrain in 4096 different environments to efficiently collect samples. In each environment, the center of mass of the robotic system is initialized with a random pose and velocity, while its joint angles are randomized about a nominal configuration. Such variability helps the robot to recover from unexpected external disturbances and generalize better.

Robots are trained and deployed on flat terrain. While the flatness of real-world ground aligns with the simulation, there can be variations in the friction coefficient. To facilitates the sim-to-real transfer, the friction of the terrain is randomized. Additionally, to further ease this transition, noise is added to all sensor readings, such joint positions, joint velocities, and base orientation.

Network parameters

Every policy trained in this research uses the same network hyperparameters, defined in Table 3.

Hyperparameter	Value
Discount factor γ	0.99
GAE parameter λ	0.95
Number of timesteps per rollout	24
Number of epochs per rollout	5
Number of minibatches per epoch	4
Entropy bonus	0.01
Value loss coefficient	1.0
Clip range	0.2
Reward normalization	yes
Learning rate	1.e-3
Number of environments	4096
Episode length	20s
Optimizer	Adam

Table 3: Network hyperparameters common to all policies trained in this study.

Rewards

The reward for each policy is defined by a combination of positive and negative reward terms. The total reward, r_{tot}, is computed as follows:

$$r_{tot} = r_{+} e^{\sigma_r r_{-}}, \quad \sigma_r = 50 \tag{8}$$

where r_{+} (resp. r_{-}) is the sum of all positive (resp. negative) rewards. This calculation ensures the final reward to be positive, thereby discouraging early termination and improving numerical stability. Indeed, since in the PPO formulation some computations involve ratios, introducing negative rewards could lead to large fluctuations, potentially leading to unstable training. Moreover, the exponential emphasis on the negative rewards ensures the agent's avoidance of areas of high negative rewards. Specifically, even a small negative reward when multiplied by σ_r significantly reduces the final reward.

Finally, for every controller, both the policy and the estimator are running synchronously at 50 Hz. The desired joint positions are computed at 50 Hz (policy) and converted to joint torques by a PD controller at 40 kHz on the real robotic platform.

4.4 Sim-to-real transitions: tackling hardware components individually

In this project, whole-body controllers are designed for the legged platform equipped with an attached arm described previously. The development of these controllers relying on a reinforcement learning approach, an analysis and comparison between the simulation models and the actual hardware is necessary to ensure a smooth transition between simulation to reality. Since the B1 legged robot and the Z1 arm represent novel platforms never investigated in the research community, there are no previous work available to determine whether the simulated models provided by Unitree reflect reality.

To facilitate the identification process, a sequential approach is adopted. Instead of developing and deploying a whole-body controller for the entire system outright, first an end-effector controller is designed in simulation and transferred to the arm hardware. This first task helps narrowing the simulation-to-reality (sim-to-real) gap for the arm component. Then, a standing policy is developed in simulation and transferred to B1's hardware to tackle the same challenge for the legged robot.

In the rest of this chapter, the methods employed to achieve these tasks will be described in detail.

4.4.1 Z1 Arm: policy and training design

This chapter describes the design of an end-effector position controller for the Z1 robotic arm. This controller is a policy trained using RL and its functionality is conditioned on the end-effector position command.

As illustrated in Figure 16, this position command is parameterized using spherical coordinates within the base frame of the arm, where r is the radius of the sphere, and θ and ϕ denote the pitch and yaw angles. This choice of parameterization facilitates the definition of a feasible training range compared to using Cartesian coordinates.

Table 4 gives the range in spherical coordinates within which the end effector is commanded during training. This range is illustrated in Figure 17, where the minimum bounds are marked in blue and the maximum bounds in red. Note that, in this figure, the legged robot illustrated serves only as a visual aid to help understanding the size of the training range for the entire system. However, the legged robot remains static during training and does not influence the results of the arm policy.

Command	Training range
r_{cmd} [m]	[0.32 , 0.7]
θ_{cmd} [rad]	$[-\frac{2}{5}\pi, \frac{2}{5}\pi]$
ϕ_{cmd} [rad]	$[-\frac{3}{5}\pi, \frac{3}{5}\pi]$

Table 4: Training range of the end effector commands

End-effector position command interpolation

During the training process, the end-effector command, $p_{cmd} = (r_{cmd}, \theta_{cmd}, \phi_{cmd})$, is interpolated linearly from the current end-effector position, $p = (r, \theta, \phi)$, towards a target position, $p_T = (r, \theta, \phi)_T$. This target position is sampled in the training distribution every $T_{cmd} = 4$ seconds,

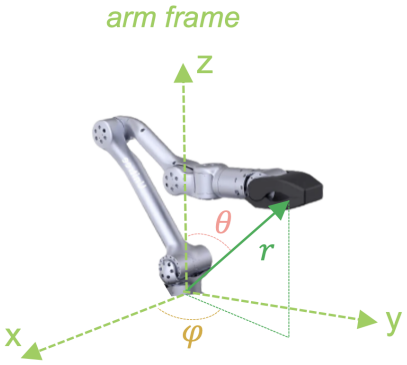


Figure 16: End effector commands defined in spherical coordinates.

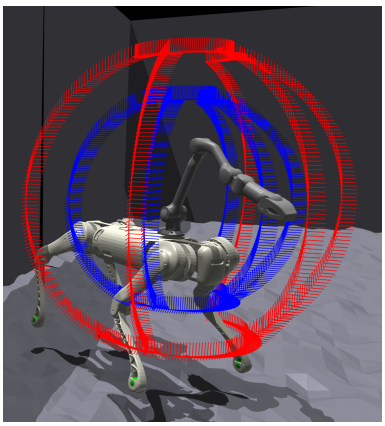


Figure 17: End effector position commands training workspace.

as expressed in Equation 9. The target position is maintained for a few seconds before a new target position is sampled.

$$p^t_{cmd} = \frac{t}{T_{cmd}} p_T + (1 - \frac{t}{T_{cmd}})p, \qquad t \in [0, T_{cmd}] \tag{9}$$

The concept is graphically represented in Figure 18, where the blue markers illustrate the interpolated commands and the red marker indicates the target command. Figure 19 provides a more detailed explanation of this concept by showcasing a sequence of three target commands within a 20-second episode. In this diagram, the blue lines represent intermediate end-effector commands, while the red lines indicate target commands.

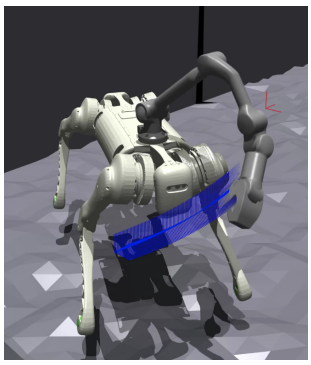


Figure 18: End-effector command interpolation: Isaac Gym illustration.

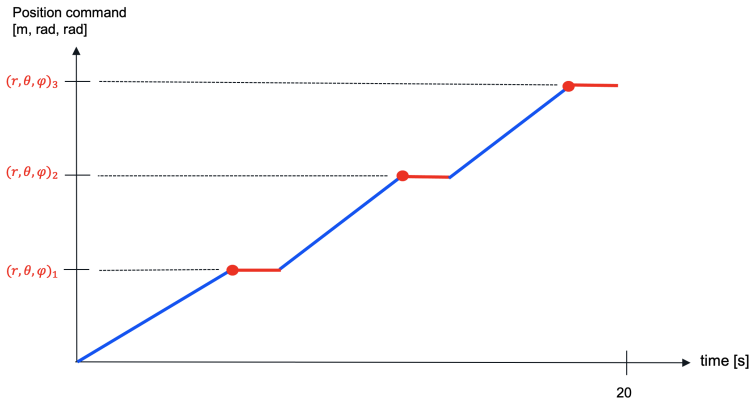


Figure 19: End-effector command interpolation: example of a sequence of three target commands.

The rationale behind this design choice to interpolate commands is made to simulate real-world interactions. Indeed, when using the Oculus set for teleoperation, the end-effector command changes at every timestep. This occurs primarily, because when the user moves the joytsick, the end-effector position commands is continuously processed and relayed to the main algorithm. Additionally, even in the absence of deliberate movement of the joystick, the command still experiences variations due to the characteristics of the Oculus tracking system.

External force application

The objective of this end-effector controller is to track a position command even under external forces. This characteristic ensures that the arm is able to interact with object without overheating.

In order to achieve this, during training, external forces are exerted to the gripper stator. The controller trains in an environment where randomized constant forces are applied to the gripper during motion for a few seconds. These force are exclusively applied along the z-axis. Their magnitude is randomly sampled between -80 and 80 N, lasting for periods varying from 1 to 3 seconds, with a resampling period set at every 8 seconds. Figure 20 illustrates this concept by showcasing a sequence of two external force applications within a 20-second episode: the first is a 70N force lasting t_1 seconds, followed by a -30N for t_2 seconds.

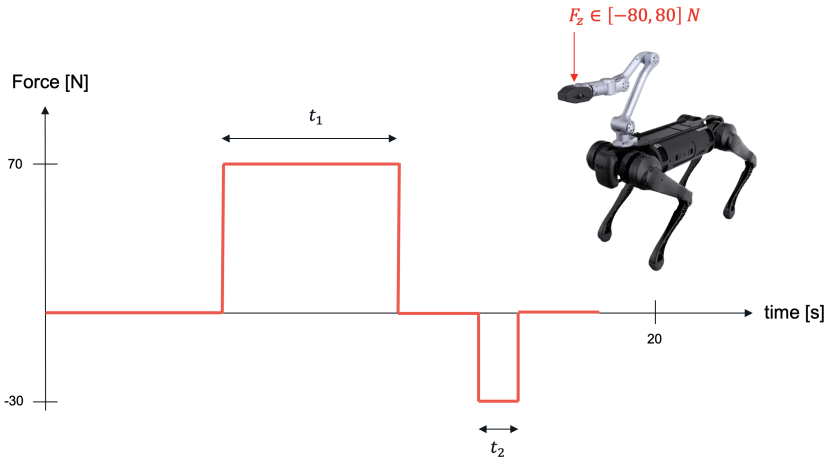


Figure 20: Example of gripper force application sequence.

This design choice stems from the understanding that there are efficient and inefficient postures for applying force. For example, when lifting a weight with an arm in either bent or straight configuration leads to different force application capabilities. Additionally, when deploying controllers that were not exposed to this external force application during training, they experienced motor overheating when applying a force to a door handle for example (see Figure 30)

Action space

This policy operates in a five-dimensional space, controlling the first five joints of the arm (J1 to J5 illustrated in Figure 6). As mentioned previously in section 3.3, the sixth and seventh joints are controlled independently using the right joystick of the Oculus headset. These two joints have been excluded from the control of the position controller, because they mainly impact the orientation rather than the position of the end-effector. Furthermore, this design choice allows to have direct control over the arm's roll angle and gripper opening angle at a fixed position, which is helpful for several manipulation tasks, like door opening for instance.

The joint target produced by the policy is the sum of the policy's output, $a^t \in \mathbb{R}^5$, and the default joint configuration, $q_{def} = [0.0, 1.0, -1.8, -0.1, 0.0][rad]$.

Observation space

One observation, o^t, from this policy consists of the end-effector position command, $(r_{cmd}, \phi_{cmd}, \theta_{cmd})$, the joint positions, $q^t \in \mathbb{R}^5$, the joint velocities, $\dot{q}^t \in \mathbb{R}^5$, the actions, $a^t \in \mathbb{R}^5$ and the previous actions $a_{t-1} \in \mathbb{R}^5$:

$$o^t = [r_{cmd}, \phi_{cmd}, \theta_{cmd}, q^t, \dot{q}^t, a^t, a^{t-1}] \in \mathbb{R}^{23} \tag{10}$$

Privileged observation

The only privileged information used for this controller is the force exerted on the gripper along the z-axis. Adding this element to the privileged information allows to have a gripper force estimate when deploying the controller on the real arm.

Reward function

The reward functions for this controller consist of the 6 elements detailed in Table 5. In this Table, *cmd* is an abbreviation of command, i represent the joints index controlled by the policy (i.e $i = 1, .., 5$) and (q_a^{min}, q_a^{max}) are the minimum and maximum joint angles as defined in Table 2. The first element guides the end-effector towards the desired position while the other five rewards encourage smooth arm movements by penalizing rapid actions and positions close to the operational limits.

Term	Equation
gripper position tracking	$\exp\{-\lvert(r,\theta,\phi) - (r,\theta,\phi)_{cmd}\rvert/\sigma\}$
joint angles violation	$\mathbb{1}_{q_i > p_q * q_a^{max} \lvert\rvert q_i < p_q * q_a^{min}}$
joint velocities	$\lvert\dot{\mathbf{q}}\rvert$
joint accelerations	$\lvert\ddot{\mathbf{q}}\rvert^2$
action smoothing	$\lvert\mathbf{a}_t - \mathbf{a}_{t-1}\rvert^2$
action smoothing, 2nd order	$\lvert\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\rvert^2$

Table 5: Reward function for the Z1 arm end-effector controller.

As mentioned previously, the final reward is defined as: $r_{tot} = r_+ e^{\sigma_r r_-}$, $\sigma_r = 50$, where r_+ (resp. r_-) is the sum of all positive (resp. negative) rewards in Table 5.

Early episode termination

A normal episode lasts 20 seconds, however in order to save computational resources and speed up the training process, early episode termination is introduced if the agent reaches a failure state. For this end-effector position controller, the episode terminates prematurely if any of the three links depicted in Figure 21 are in collision. This early termination not only prevents potential hardware damage but also improves learning by avoiding wasting time exploring physically impossible states.

It is important to mention that, in this work, agents are not penalized for early termination, contrasting with other studies [17]. However, if an agent successfully reaches the end of an episode, it receives an additional reward. This benefit is an approximation of the reward it would have earned if the episode would not have concluded. This approach, known as bootstrapping on time outs, computes this prediction based on the value estimate at the episode's end.

Finally, note that the decision to design a position, rather than a pose controller has been motivated by the complexity of simultaneously commanding the end effector's position and

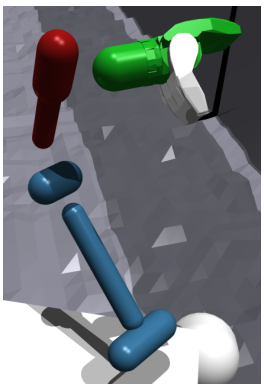


Figure 21: Collision meshes of the arm (link 02 in blue, 03 in red and 06 in green).

orientation. Given that not every orientation can be attained from a specific position, defining a feasible training range for both commands is challenging. An attempt to bypass this challenge by simply defining a range for positions and orientations independently, led to unsatisfactory results.

4.4.2 B1 robot: policy and training design

Before deploying a whole-body controller for the mobile legged manipulator, the legged robot has to stand-up first. In this section, the method used to enable B1 to stand up is explained.

In previous work, for platforms like the A1 and Go1 robots from Unitree, the stand-up controller relied on a simple linear interpolation transitioning from the seated leg joint positions to the standing configuration. Thanks to the low inertia and mass of these robots, this approach was effective. However when applied to B1, a considerably larger and heavier robot, this method proved to be ineffective. Figures 22 and 23 illustrate these results. A few adjustments were trialed for B1, such as changing the PD gains, modifying the target joint angles, trying a two-stage stand-up schedule, but none were successful. Consequently, a stand-up policy was trained in Isaac Gym using the architecture described in section 4.2.

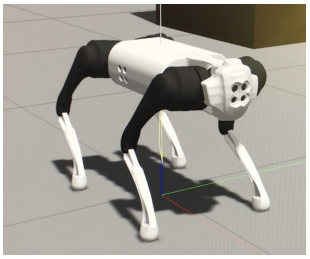


Figure 22: Unitree Go1 robot achieving successful stand-up using open-loop control.

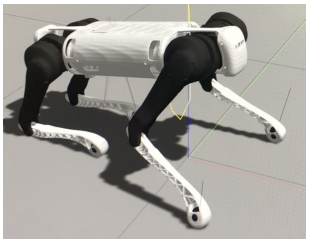


Figure 23: Unitree B1 robot's unsuccessful attempt at standing up using open-loop control.

In the following, the action space, observation space, reward function and the early termination criterion used to train the stand-up policy are described.

Action space

This policy operates in a twelve-dimensional space, controlling the thigh, calf and hip joint angles of the B1 robot. The joint target produced by the policy is the sum of the policy's output, $a_t \in \mathbb{R}^{12}$, and the default joint configuration, $\mathbf{q}_{def} \in \mathbb{R}^{12}$

In this report, the B1 robot's legs are segmented into 4 categories: Front Left (FL), Front Right (FR) and Rear Left (RL), and Rear Right (RR). Each leg has three joints: the hip, thigh, and calf. Considering this categorization, the default joint configuration, $\mathbf{q}_{def}$, is defined as follows: the hip is at 0.55 rad, the thigh at 0.78 rad, and the calf at -2.6 rad for both FL and RL legs. However, for the FR and RR legs, the hip is set at -0.55 rad, with both the thigh and calf angles remain at 0.78 rad and -2.6 rad.

In order to enable the policy to learn how to stand up with the additional weight of the arm, it is mounted on the robot during the learning phase. For this policy, the degrees of freedom of the arm are fixed and set to: $\mathbf{q}_{def} = [0.0, 0.8, 0.8, 0.0, 0.0, 0.0, 0.0][rad]$.

Observation space

One observation, o^t, from this policy consists of the joint positions, $q^t \in \mathbb{R}^{12}$, the joint velocities, $\dot{q}^t \in \mathbb{R}^{12}$, the actions, $a^t \in \mathbb{R}^{12}$ and the previous actions $a^{t-1} \in \mathbb{R}^{12}$:

$$o^t = [q^t, \dot{q}^t, a^t, a^{t-1}] \in \mathbb{R}^{48} \tag{11}$$

Reward function

The reward functions for this controller consist of the 10 elements detailed in Table 6.

Term	Equation
target joint positions tracking	$\exp\{-\|\|\mathbf{q} - \mathbf{q}_{target}\|\|_2^2/\sigma_q\}$
body height tracking	$\exp\{-(h - h_{target})^2/\sigma_h\}$
feet contact forces	$\exp\{-\frac{\text{std}(\|\|\mathbf{F}_{\text{R/L, front}}\|\|)+\text{std}(\|\|\mathbf{F}_{\text{R/L, rear}}\|\|)}{\sigma_f}\})*\mathbb{1}_{\|\|\mathbf{F}_{\text{R/L, front/rear}}\|\| \in [F_{\min}, F_{\max}]} = r_c$
thigh/calf collision	$\mathbb{1}_{collision}$
joint angle limits violation	$\mathbb{1}_{\mathbf{q} > \mathbf{q}_l^{max} \|\| \mathbf{q} < \mathbf{q}_l^{min}}$
joint torque limits violation	$\mathbb{1}_{\|\boldsymbol{\tau}\| > p * \boldsymbol{\tau}_l^{max}}$
joint velocities	$\|\dot{\mathbf{q}}\|$
joint accelerations	$\|\ddot{\mathbf{q}}\|^2$
action smoothing	$\|\mathbf{a}_t - \mathbf{a}_{t-1}\|^2$
action smoothing, 2nd order	$\|\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\|^2$

Table 6: Reward function for the B1 stand-up controller.

In this Table, the range of joint angles is represented by $[\mathbf{q}_l^{min}, \mathbf{q}_l^{max} \in \mathbb{R}^{12}]$, while $\boldsymbol{\tau}_l^{max} \in \mathbb{R}^{12}$ denotes the maximum joint torques, as specified in Table 1. $\mathbf{F}_{\text{R/L, front}} \in \mathbb{R}^3$ represent the reaction forces of the right (R) and left (L) front legs, while $\mathbf{F}_{\text{R/L, rear}} \in \mathbb{R}^3$ are the reaction forces of the right (R) and left (L) rear legs.

The first reward term guides the leg joints towards the stand-up pose $\mathbf{q}_{target}$. The target configuration for the legs is as follows: both the FL and RL legs have the hip set at -0.1 rad, the thigh at 0.67 rad, and the calf at -1.3 rad. In contrast, the FR and RR legs have their hip at 0.1 rad, but the thigh and calf angles remain the same at 0.67 rad and -1.3 rad respectively. The second reward term encourages the robot to raise its body to the desired stand-up height of $h_{target} = 0.54$ [m].

The third reward term prevents the robot from jumping and encourages a balanced weight distribution on both its sides. This reward is only nonzero when the four feet are in contact with the ground and it increases if the ground reaction force of the front (resp. rear) legs are similar.

The fourth reward penalizes collisions with the thigh and calf rigid bodies, while the last six rewards encourage smooth leg movements by penalizing rapid actions, high torques and positions close to the operational limits.

Early episode termination

A normal episode lasts 20 seconds, however in order to save computational resources and speed up the training process, early episode termination is introduced if the agent reaches a failure state. For the stand-up controller, the episode terminates prematurely if the roll and pitch angle of the base significantly deviate from a flat pose. Specifically, an environment reset is triggered if the magnitude of x and y components of the projected gravity vector in the base frame, exceeds 0.5.

4.5 Whole-body controllers: policies and training designs

In this section, the whole-body position and force controllers are described. To provide a concise overview, shared characteristics of these controllers are presented in this chapter. Notably, they possess the same action and observation space structures, along with some identical training details.

Action space

These policies operate in a seventeen-dimensional space, controlling the thigh, calf and hip joint angles of the B1 robot, as well as the first five joints of the arm (J1 to J5 illustrated in Figure 6). As mentioned previously in section 3.3, the sixth and seventh joints are controlled independently using the right joystick of the Oculus headset.

The joint target produced by the policy is the sum of the policy's output, $a_t \in \mathbb{R}^{17}$, and the default joint configuration, $\mathbf{q}_{def} \in \mathbb{R}^{17}$. The legs default joint configuration is defined as follows: the hip is at -0.1 rad, the thigh at 0.67 rad, and the calf at -1.3 rad for both FL and RL legs. However, for the FR and RR legs, the hip is set at 0.1 rad, with both the thigh and calf angles remain at 0.67 rad and -1.3 rad. For the arm, the default configuration is $[0.0, 1.0, -1.8, -0.1, 0.0]$ rad.

Observation space

A single observation, represented by o^t, from these policies consist of the task-associated command, $t_{cmd} \in \mathbb{R}^{N_c}$, the base orientation $R^t_{base} \in \mathbb{R}^3$, the feet clock timings $c^t_{feet} \in \mathbb{R}^4$, the joint positions, $q^t \in \mathbb{R}^{17}$, the joint velocities, $\dot{q}^t \in \mathbb{R}^{17}$, the actions, $a^t \in \mathbb{R}^{17}$ and the previous actions $a^{t-1} \in \mathbb{R}^{17}$:

$$o^t = [t^t_{cmd}, R^t_{base}, c^t_{feet}, q^t, \dot{q}^t, a^t, a^{t-1}] \in \mathbb{R}^{75+N_c} \tag{12}$$

The aforementioned feet clock timings c_{feet} are computed based on the methodology presented in [4]. In the current context, only the trotting gait is implemented with a phase parameter of $\theta_{trot} = 0.5$. The feet clock timings are computed as follows. At each control time-step, the global timing variable T is incremented by $\frac{f_{cmd}}{f_\pi}$ where f_{cmd} represents the stepping frequency and f_π is the control frequency. The stepping frequency is set to 2Hz and results in each foot making contact with the ground twice per second, while the control frequency is defined at

50Hz. Then, a separate timing variable for each foot is calculated and clipped between 0 and 1:

$$\begin{aligned} T_i^{t+1} &= \text{clip}_{[0,1]}([T_i^t + \frac{f_{cmd}}{f_\pi} + \theta_{trot}]), & i = \{FR, RL\} \\ T_j^{t+1} &= \text{clip}_{[0,1]}([T_j^t + \frac{f_{cmd}}{f_\pi}]), & j = \{FL, RR\} \end{aligned} \tag{13}$$

These timing variable are then utilized to define clock timings for each foot:

$$c_{feet}^t = [\sin(2\pi T_{FR}^t), \sin(2\pi T_{FL}^t), \sin(2\pi T_{RR}^t), \sin(2\pi T_{RL}^t)] \in \mathbb{R}^4 \tag{14}$$

These clock timings vary smoothly between 0 and 1 following a sinusoidal wave. In this range, a value of 0 corresponds to a foot in contact with the ground, while 1 indicates a foot at its peak elevation.

The dimension N_c of the task-related command, presented in equation 12 varies among the whole-body position and force controllers and will be defined for each in the following sections.

Practical training details

During the training phase, the initial joint angles of every agent are set to the final joint configuration of the stand-up policy described in section 4.4.2. This choice incentives a smooth transition when switching between controllers during deployment. This set of initial joint angles is the following $[0.1, 1.05, -1.3, -0.03, 0.99, -1.36, 0.45, 0.81, -0.89, -0.34, 0.88, -0.86]$ rad. This sequence of values order corresponds to $[FL_{\text{hip,thigh,calf}}, RL_{\text{hip,thigh,calf}}, FR_{\text{hip,thigh,calf}}, RR_{\text{hip,thigh,calf}}]$

As the policy trains, the robot experiences random external disturbances applied in the form of instantaneous linear base velocity impulses in the horizontal plane. These disturbances, whose aim is to enhance the robot's robustness against external disturbances, occur every 15 seconds with a magnitude randomly sampled between -0.8 and 0.8 m/s.

4.5.1 End-effector positioning and locomotion controller

This chapter describes the design of the unified policy for coordinated manipulation and locomotion. The proposed policy controls the joints of the legged robot and the arm simultaneously according to desired end-effector position commands in the arm frame and desired base velocities in the body frame.

In the rest of this section, the training design used to develop the whole-body position policy is described.

Commands

The inputs of this controller are separated into two main sets of commands: the base velocity command and the end-effector position command. Specifically, the former consists of the robot linear velocity in the base frame x- and y- axes, $(v_x^{\text{cmd}}, v_y^{\text{cmd}})$, and the robot angular velocity in the yaw axis, ω_z^{cmd}. The latter is parameterized using spherical coordinates, where r^{cmd} represent the radius of the sphere, and θ^{cmd} and ϕ^{cmd} denote the pitch and yaw angles. Similarly to the training procedure for the arm position controller described in section 4.4.1, the end-effector commands are linearly interpolated from the current end-effector position to a target one.

The commands are sampled in the ranges defined in Table 7.

Command	Range	Units
End-effector position: r^{cmd}	[0.3, 0.9]	m
End-effector position: θ^{cmd}	$[-\frac{2}{5}\pi, \frac{2}{5}\pi]$	rad
End-effector position: ϕ^{cmd}	$[-\frac{3}{5}\pi, \frac{3}{5}\pi]$	rad
Base velocity: v_x^{cmd}	[-1.0, 1.0]	m/s
Base velocity: v_y^{cmd}	[-1.0, 1.0]	m/s
Base velocity: ω_z^{cmd}	[-1.0, 1.0]	rad/s

Table 7: Whole-body position controller command ranges.

Observation space

For this policy, the task-associated command, t_{cmd}, is six-dimensional (i.e $N_c = 6$). It consists of the end-effector position command in the base frame, $(r^{cmd}, \phi^{cmd}, \theta^{cmd})$, the linear velocity commands in the base frame x- and y- axes, (v_x^{cmd}, v_y^{cmd}), and the angular velocity command in the yaw axis, ω_z^{cmd}.

Hence, as described in equation 12, one observation for this policy is 81-dimensional: $o^t \in \mathbb{R}^{75+N_c} = \mathbb{R}^{81}$.

Privileged observation

The privileged information used for this controller are the linear velocity in the base frame x- and y- axes, the angular base velocity in the yaw axis, and the end-effector position in the base frame.

Reward function

The reward functions for this controller consist of the 13 elements detailed in Table 8. Within this table, arm and legs joint angle limits are represented by $[\mathbf{q}_a^{min}, \mathbf{q}_a^{max} \in \mathbb{R}^5]$ and $[\mathbf{q}_l^{min}, \mathbf{q}_l^{max} \in \mathbb{R}^{12}]$ respectively. The combined joint angles of the entire system are denoted by $\mathbf{q} = [\mathbf{q}_a, \mathbf{q}_l] \in \mathbb{R}^{17}$.

The first element guides the end-effector towards the desired position $(r, \theta, \phi)^{cmd}$. Similarly to the method proposed by Zipeng Fu et al. [7], these commands are defined in a coordinate frame aligned with the arm's frame, but remains unaffected by changes in the robot's base height, or its roll and pitch orientation. Specifically, first an end-effector position command, p_{ee}^A, is sampled in spherical coordinates in the arm frame from the ranges defined in Table 7. Then, the corresponding position in the world frame is computed as $p_{ee}^W = \text{T}(\text{S2C}(p_{ee}^A)) + p_{base}^W$, where T is the orientation transformation solely based on the yaw heading of the robot, S2C is the operator transforming spherical coordinates to Cartesian coordinates, and p_{base}^W is the base position of the robot, with the z-coordinate set to 0.6m.

The second, third and fourth reward components perform omnidirectional velocity tracking of the robot's base, while the role of the next two rewards is to track a desired feet contact schedule. This schedule is defined by the target contact state, C_i^{cmd}, of each foot and the phase variable of the trotting gait, θ_{trot}. The function C_i^{cmd} is computed using the timing variables outlined in Equation 13 as follows:

$$C_i^{cmd}(T_i(\theta_{trot}, T_i)) = \Phi(T_i, \sigma)(1 - \Phi(T_i - 0.5, \sigma)) + \Phi(T_i - 1, \sigma)(1 - \Phi(T_i - 1.5, \sigma))$$
$$\text{i} = \{\text{FR, FL, RR, RL}\}$$

where Φ is the cumulative density function of the normal distribution:

$$\Phi(x, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \tag{15}$$

A key feature of the proposed approach is that when the magnitude of the commanded base velocities falls below 0.1, the desired contact schedule for each foot is set to 0, ensuring all feet remain grounded. This characteristic prevents the robot from stepping in place for minimal velocities, easing the teleoperation of the arm.

Finally, the seventh reward penalizes collisions with the robot's thigh, calf and hip rigid bodies and with the arm's link02, link03 and link06 rigid bodies. The last six rewards encourage smooth leg and arm movements by penalizing rapid actions and positions close to the operational limits.

Term	Equation
gripper position tracking	$\exp\{-\lvert(r,\theta,\phi) - (r,\theta,\phi)^{cmd}\rvert/\sigma_p\}$
x velocity tracking	$\exp\{-\lvert v_x - v_x^{cmd}\rvert^2/\sigma_{vx}\}$
y velocity tracking	$\exp\{-\lvert v_y - v_y^{cmd}\rvert^2/\sigma_{vy}\}$
yaw velocity tracking	$\exp\{-\lvert\omega_z - \omega_z^{cmd}\rvert^2/\sigma_{\omega z}\}$
swing phase tracking (force)	$\sum_i^4 \left[1 - C_i^{cmd}(t)\right] \exp\{-\lvert\mathbf{f}^i\rvert^2/\sigma_{cf}\}$
stance phase tracking (velocity)	$\sum_i^4 \left[C_i^{cmd}(t)\right] \exp\{-\lvert\mathbf{v}_{xy}^i\rvert^2/\sigma_{cv}\}$
collision penalty	$\mathbb{1}_{collision}$
arm joint limit violation	$\mathbb{1}_{\mathbf{q}_a > p_a * \mathbf{q}_a^{max} \lvert\lvert \mathbf{q}_a < p_a * \mathbf{q}_a^{min}}$
leg joint limit violation	$\mathbb{1}_{\mathbf{q}_l > p_l * \mathbf{q}_l^{max} \lvert\lvert \mathbf{q}_l < p_l * \mathbf{q}_l^{min}}$
joint velocities	$\lvert\dot{\mathbf{q}}\rvert$
joint accelerations	$\lvert\ddot{\mathbf{q}}\rvert^2$
action smoothing	$\lvert\mathbf{a}_t - \mathbf{a}_{t-1}\rvert^2$
action smoothing, 2nd order	$\lvert\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\rvert^2$

Table 8: Reward function for the whole-body controller.

Similarly to the previous policies, the final reward is defined as: $r_{tot} = r_+ e^{\sigma_r r_-}$, $\sigma_r = 50$, where r_+ (resp. r_-) is the sum of all positive (resp. negative) rewards in Table 8.

Early episode termination

A normal episode lasts 20 seconds, however in order to save computational resources and speed up the training process, early episode termination is introduced if the agent reaches a failure state. For this whole-body position controller, the episode terminates prematurely if the gripper mover is in collision or if the body height falls below 0.3m.

4.5.2 End-effector force controller

This chapter delves into the design of the unified policy for coordinated gripper force application. The proposed policy controls the joints of the legged robot and the arm simultaneously according to a desired force defined in the world frame. In contrast with end-effector position control, the force control task offers some distinct benefits: (1) the robot can learn to be compliant and yield to external forces, even when carrying a payload, and (2) the robot's whole body and arm posture can adapt to facilitate force application without competing objectives.

In the rest of this section, the training design used to develop the whole-body force policy is described.

Commands

This controller operates based on the desired force vector at the end-effector $\mathbf{F}_g^{cmd} \in \mathbb{R}^3$. This force is expressed in Cartesian coordinates in the world frame and is sampled between - 70 and 70 N for each axis.

Every 10 seconds, a force target is sampled, as well as a duration, t_F , which defines the duration of the force application and ranges between 2.0 and 4.0 seconds. The force command undergo linear interpolation from 0 to the sampled values in t_F seconds. Then, they maintain these values for a duration of 1.5 seconds, after which they are linearly interpolated back to 0 in t_F seconds. This process repeats every 10 seconds. To ensure that fully compliant behavior is experienced frequently, each time an environment is reset, it has a 20% chance of receiving a zero force target.

Task sampling

A key feature of the force control training is an emulated soft contact between the gripper and the environment, where the force application can be controlled by slight adjustments of the position. An external force, $\mathbf{F}_e \in \mathbb{R}^3$, that pulls on the gripper as a function of its displacement from a force setpoint, is simulated.

The force is modulated using a proportional-derivative control scheme on the position and velocity difference between the gripper position, $\mathbf{x}_g^t \in \mathbb{R}^3$ and the setpoint $\mathbf{x}_s \in \mathbb{R}^3$.

$$\mathbf{F}_e^t = K_p(\mathbf{x}_s - \mathbf{x}_g^t) + K_d(\dot{\mathbf{x}}_s - \dot{\mathbf{x}}_g{}^t) = K_p(\mathbf{x}_s - \mathbf{x}_g^t) - K_d\dot{\mathbf{x}}_g{}^t \tag{16}$$

The chosen gains, $K_p = 700$, $K_d = 6$, were tuned by inspecting the behavior of the simulator to ensure that applied force is large enough to bring the gripper to the setpoint but small enough to avoid extreme torques or oscillations.

If the force setpoint is initialized randomly, it may be unreachable or in collision with the robot's body. To circumvent this, the force setpoint is instead initialized at the gripper's initial position. To obtain diverse initial positions, the whole-body position controller, described in section 4.5.1, is used to sample robot poses by randomly generating 1000 end-effector commands. These were sampled within a subset of the complete workspace of the whole body position controller and are located in front of the robot in a restricted workspace with depth 40 cm, width 80 cm, and height 75 cm, as shown in Figure 24. During this sampling procedure, each agent is initialized with a randomized pose. Then, the whole body position controller operates for fixed number of iterations, maintaining a constant end-effector position command. If no collision is recorded during this period, the robot's current pose and joint configuration are saved. These saved postures serve as the initialization states for the whole-body force controller.

For the force controller, every reset aligns the robot's joint angles, base orientation and position with the ones collected using the whole-body position controller. Upon each robot's spawning, the gripper position in the world frame is recorded and serves as the gripper setpoint during the entire episode. This initialization method ensures that the gripper setpoints are both reachable by the legged manipulators and free from collisions.

Observation space

For this policy, the task-associated command, t_{cmd}, is three-dimensional (i.e $N_c = 3$). It consists of the end-effector force command. Hence, as described in equation 12, one observation for this policy is 78-dimensional: $o^t \in \mathbb{R}^{75+N_c} = \mathbb{R}^{78}$.

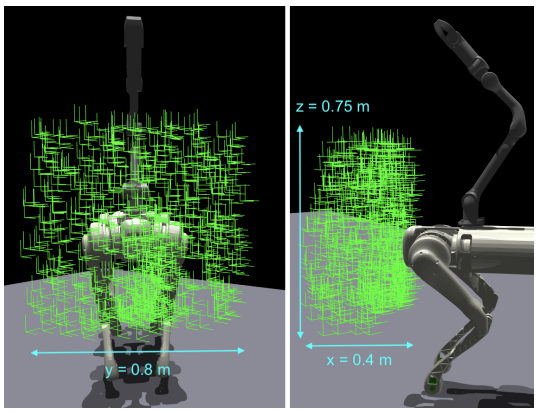


Figure 24: Gripper setpoints employed for training the whole-body force controller.

Privileged observation

The privileged information used for this controller is the three-dimensional force applied to the gripper and the joint position error provided to the proportional-derivative controller, which appeared to help reducing the arm and leg torques.

Reward function

The reward structure for this controller comprises the 10 elements detailed in Table 9. The first element incentives the end-effector to apply the desired force $\mathbf{F}_g^{cmd}$.

The second component penalizes arm torques close to their operational limits. This reward enabled the platform to exhibit a whole body behavior, allowing the legs to apply more force while the arm reduces its force contribution by stretching out.

The role of the next reward is to track a desired foot contact schedule. To encourage firm stance during force application, the desired contact schedule for each foot is set to zero.

Finally, the fourth reward penalizes collisions with the robot's thigh and the last six rewards encourage smooth leg an arm movements by penalizing rapid actions and positions close to the operational limits.

Term	Equation
force tracking	$\exp\{\text{-}\lvert\mathbf{F}_g^{cmd} - \mathbf{F}_g\rvert/\sigma_F\}$
arm torque limit violation	$\mathbb{1}_{\tau_a > p_a * \lvert\tau_a^{max}\rvert}$
stance phase tracking	$\sum_i^4 \left[C_i^{cmd}(t)\right] \exp\{\text{-}\lvert\mathbf{v}_{xy}^i\rvert^2/\sigma_{cv}\}$
collision penalty	$\mathbb{1}_{collision}$
arm joint limit violation	$\mathbb{1}_{\mathbf{q}_a > p_a * \mathbf{q}_a^{max} \lvert\lvert \mathbf{q}_a < p_a * \mathbf{q}_a^{min}}$
leg joint limit violation	$\mathbb{1}_{\mathbf{q}_l > p_l * \mathbf{q}_l^{max} \lvert\lvert \mathbf{q}_l < p_l * \mathbf{q}_l^{min}}$
joint velocities	$\lvert\dot{\mathbf{q}}\rvert$
joint accelerations	$\lvert\ddot{\mathbf{q}}\rvert^2$
action smoothing	$\lvert\mathbf{a}_t - \mathbf{a}_{t-1}\rvert^2$
action smoothing, 2nd order	$\lvert\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\rvert^2$

Table 9: Reward function for the whole-body force controller.

Early episode termination

For the whole-body force controller, an episode terminates prematurely under several conditions:

- If the robot's calf and hip rigid bodies or the arm's link02, link03, link06 and gripper mover rigid bodies are in collision.
- If the body height drops below 0.3m.
- If the arm torques exceed 90% of their limits.
- If the distance between the end-effector and the setpoint surpasses 0.4 meters.

5 Results

In this chapter, the main results of this research are presented. Sections 5.1, 5.2, and 5.3 detail the sim-to-real transition and performance evaluations for the Z1 arm, B1 robot, and the end-effector position and locomotion controller, respectively. All controllers are assessed in both simulation and real-world settings. Finally, section 5.4 describes the whole-body force controller's performance and its practical applications, including compliant tasks.

5.1 Z1 arm

In this section, the simulation-to-reality gap process for the Z1 arm, as well as the results of the end-effector position controller described in chapter 4.4.1, are presented.

5.1.1 Z1 arm: sim-to-real

Bridging the simulation-to-reality for the Z1 arm has been very challenging, mainly due to the lack of information in the datasheet provided by Unitree. This section outlines the significant steps undertaken to address this challenge.

Unitree control interface

The Unitree Z1 arm can be controlled either in task space for high-level operations or in joint space for low-level operations. In low-level control mode, users have direct control over the parameters of the Proportional and Derivative (PD) control law for the motor controllers. These parameters include the desired joint position q_{des}, the desired joint velocity $\dot{q}_{des}$, the proportional gain K_p, the derivative gain K_d and the feedforward torque τ_{ff}. For each joint, the final motor's torque is computed as follows:

$$\tau_i = K_{p,i}(q_{des,i} - q_i) + K_{d,i}(\dot{q}_{des,i} - \dot{q}_i) + \tau_{ff,i}, \quad i = 0, ..., 6 \tag{17}$$

For the Z1 arm, Unitree provides an interface, allowing users to easily test the same code in Gazebo with the Robotic Operating System as middleware, or directly on the real robotic arm. Experiments showed that the behavior of the real arm resembles more the Gazebo simulation than the Isaac Gym one. The Gazebo interface has been used extensively as a safety measure to ensure the robotic arm's safety. Specifically, if a policy does not work as expected in Gazebo, it is not deployed on the real arm.

Transition to RL policy

When deploying the RL policy on the real arm, to mirror the simulation conditions, the arm is first controlled to the initial configuration and then the RL policy is activated. Upon starting the policy, the arm exhibits a strong shaking behavior. Figure 25 illustrates this unstable behavior by superposing 4 frames of a video. In this figure, step 1 represents the initial configuration, steps 2 and 3 show the unstable response, and by step 4, the arm stabilizes under the RL policy.

For a more detailed view, Figure 26 shows the joint angles evolution over time. In this plot, up until the 38th time step, the arm moves to the initial configuration in open loop control mode, and then the RL policy is deployed.

This initial instability can be substantially reduced by incrementally clipping the first actions from 0 to the full action range (i.e 10 [rad]) over a 4-second duration.

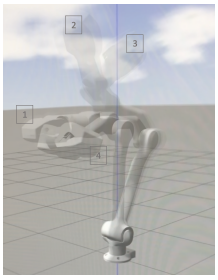


Figure 25: Transition to RL: Gazebo illustration

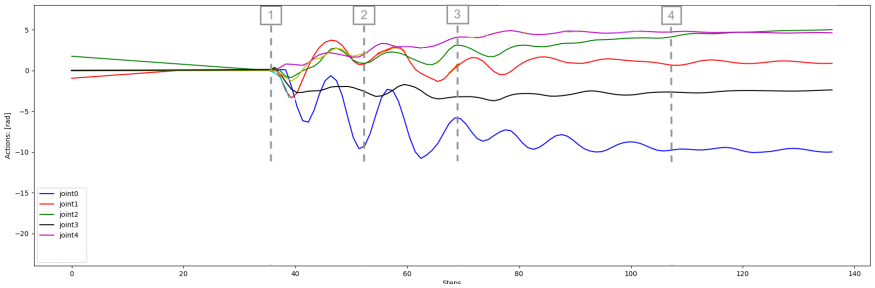


Figure 26: Transition to RL: arm joint angles over time

PD gains tuning

After a hyperparameter tuning step, the end-effector controller performed nicely in Isaac Gym, but when deployed in Gazebo, the arm exhibited shaking. Figure 27 superimposes three frames from a video showcasing the arm trying to maintain a commanded constant end-effector position (illustrated by a red dot). This result has been achieved by sending the following commands to the PD controllers of the robotic arm:

$$q_{des,i} = q_{def,i} + \sigma_a a_{t,i}, \quad \dot{q}_{des,i} = 0, \quad \tau_{ff,i} = 0, \quad K_{p,i} = 40, \quad K_{d,i} = 2 \quad i = 0, ..., 6 \tag{18}$$

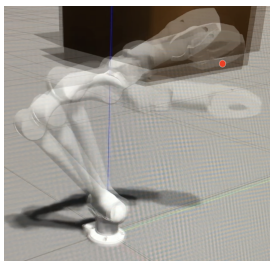


Figure 27: First RL policy deployed: oscillations in Gazebo

After several experimentation rounds, a solution has been found which substantially stabilized the arm in Gazebo and in the real world. The updated method involved using smaller PD gains and send the final torque as a feedforward (FF) torque, as described in Equation 19.

$$\begin{gathered} q_{des,i} = q_i, \ \dot{q}_{des,i} = \dot{q}_i, \ K_{p,i} = 18, \ K_{d,i} = 0.8 \\ \tau_{ff,i} = K_{p,i}(q_{def,i} + \sigma_a a_{t,i} - q_i) + K_{d,i}(0 - \dot{q}_i) \quad i = 0, ..., 6 \end{gathered} \tag{19}$$

First arm teleoperation

The end-effector policy being stabilized, the arm could be teleoperated using the tracking system of the Oculus headset. Figures 28 and 29 illustrate this initial teleoperation, where the gripper follows the commands given by the human's hand holding the Oculus joystick.

After achieving stable teleoperation, the system was guided to attempt the door-opening task. The arm was successfully teleoperated to approach the door, open and close the gripper on the door handle, but the motor for joint 0 powered down during the handle-pulling action. This behavior has been engaged by Unitree's low-level safety controller. While the source code of this safety function is not open source, it is suspected that it was triggered when the cumulative power across joints exceed a maximum value. Figure 30 supports this hypothesis, where the

Figure 28: Teleoperation of the first arm policy: illustration 1

Figure 29: Teleoperation of the first arm policy: illustration 2

torque of joint 0 is at the torque limits for a prolonged period (from timestep 1300 onwards), possibly causing its motor to shut down.

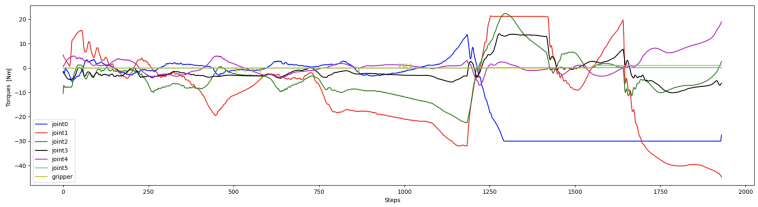


Figure 30: Arm joint torques over time when opening a door.

Gripper force application during training

As previously mentioned in section 4.4.1, applying randomized constant forces to the gripper during training equips the controller with the understanding of efficient versus inefficient postures for resisting force. Using this training schedule, initial controllers could resist external forces applied on the gripper, but they needed a significant period of time to stabilize at the commanded end-effector position. Indeed, upon the external force application, these controllers resulted in large oscillations of the gripper until they could counteract the force and track the end-effector command in a stable manner. Figure 72 in the Appendix section depicts this behavior.

Incorporating the external force applied vertically to the gripper as a privileged information has proven to considerably minimize these oscillations. The resulting controller is able to rapidly and smoothly respond to external forces, as illustarted in Figure 73 in the Appendix sections.

Figure 31 shows an example of the external force being applied to the gripper, alongside its corresponding force estimate from the *Estimator* network. The force estimate follows closely the external force applied to the gripper, demonstrating the accuracy of the force *Estimator* network in simulation.

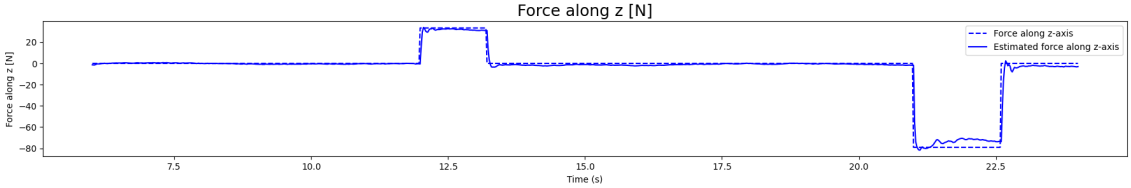


Figure 31: External force applied to the gripper along the z-axis, alongside the force estimation in Isaac Gym.

Bad gripper force estimate on hardware

The aforementioned controller, having demonstrated good performance in simulation, was then used to teleoperate the arm for the door-opening task once more. This time, the motors did not overheat and the task was accomplished successfully. However, despite the force estimate being impressively precise in simulation, it proved to be inaccurate on the actual hardware, as shown in Figure 32. In this example, the arm was commanded to track a predefined trajectory of end-effector position commands, while no force was applied on the gripper. Ideally, the force estimate should be zero, however as observed in Figure 32, the estimated force is very noisy, fluctuating randomly between 0 and 20 N.

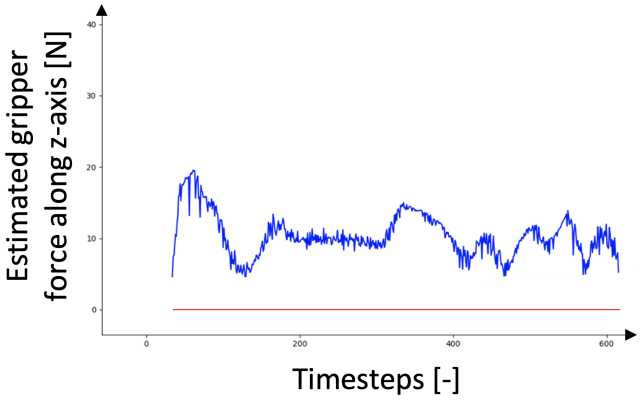


Figure 32: Estimation of vertical force by the *Estimator* network when no force is applied to the gripper: blue represents the estimate and red shows the target.

Transitioning from FeedForward (FF) to Proportional-Derivative (PD) control

The previous subsections showed that, while the end-effector position tracking component of the policy exhibits good performance both in simulation and hardware, the force estimate is accurate only in simulation.

The current control framework is:

- *Simulation*: In Isaac Gym, the agents are commanded using the torque control mode with relatively low gains ($K_{p,i}$=18, $K_{d,i}$ = 0.8) with the friction and damping factors of the arm URDF set to 0.0.
- *Hardware*: The torques are computed manually ($K_{p,i}$=18, $K_{d,i}$ = 0.8) and they are sent using the Feedforward torque interface of the arm.

The core objective is to deploy the policy utilizing the PD control mode of the arm using high gains. The motivation for this is twofold:

1. *Improved tracking performance with PD mode*: When commanding the arm in open-loop, the PD control mode achieves perfect tracking with high control gains. In contrast, the feedforward mode exhibits lower performance, especially when high gains are used, leading to an oscillating behavior of the arm.
2. *High-frequency control*: The PD control mode operates at a higher frequency compared to the FF mode. This characteristic is suspected to impact the force estimate.

To achieve this goal, two steps were undertaken. First, a system identification has been conducted for the Z1 arm. Then, a method has been discovered to train the policies in Isaac Gym using high gains. These two phases are described in details in the following.

Z1 arm: system identification

The Z1 robotic arm underwent a system identification process to first derive an accurate torque model for its joints, and then understand the differences between the different control modes offered by Unitree.

The arm's joints can be controlled in two distinct ways:

1. *FeedForward mode*: This mode can only be used with low gains, otherwise it results in the arm shaking substantially.
2. *Proportional-Derivative mode*: This mode requires high gains to function, otherwise the arm reamins stationary.

The performance of these two control modes has been assessed through open-loop trajectory tests. The PD control mode with high gains ensures perfect tracking, whereas the FF mode does not, as illustrated in Figures 33 and 34. These plots clearly reflect that the PD control mode should be preferred over the FF mode to reach accurate joint position tracking.

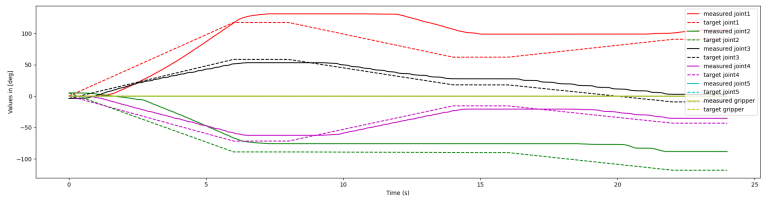


Figure 33: Open loop trajectory tracking of the real robotic arm controlled in FF mode with low gains.

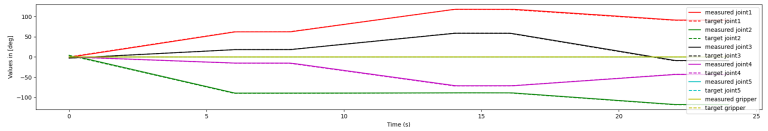


Figure 34: Open loop trajectory tracking of the real robotic arm controlled in PD mode with high gains.

These open-loop experiments revealed that the gains utilized by the motors are implicitly scaled in the source code in PD control mode. The real motor torques are defined as follows:

$$\tau = K_{p,i} * 25.6(q_{d,i} - q_i) + K_{d,i} * 0.0128(\dot{q}_{d,i} - \dot{q}_i) \quad i = 1, ...7 \tag{20}$$

A dive into an example file provided by Unitree developers, revealed surprisingly elevated default gains:

$$\begin{gathered} K_{p,Unitree} = [20.0, 30.0, 30.0, 20.0, 15.0, 10.0, 20.0, 10.0] \\ K_{d,Unitree,i} = 2000.0 \quad i = 1, ...7 \end{gathered} \tag{21}$$

Given the multiplication by constants in the source code, the actual gains are:

$$\begin{gathered} K_{p,real} = K_{p,Unitree} * 25.6 = [512.0, 768.0, 768.0, 512.0, 384.0, 256.0, 512.0] \\ K_{d,real,i} = K_{d,Unitree,i} * 0.0128 = 25.6 \quad i = 1, ...7 \end{gathered} \tag{22}$$

Given the high values of these gains, the goal is to progressively reduce them until the tracking performance drops substantially. Lower gains are typically preferred because they generally provide a more stable control response, especially by reducing the noise's impact on the control action for example.

In open-loop, the arm could still move accurately when dividing $K_{p,real}$ and $K_{d,real}$ by 6, however when divided by 8, the joint1 was not able to track the reference anymore. Hence every arm controller mentioned henceforth is trained in simulation with:

$$\begin{aligned} K_{p,simulation} &= \frac{K_{p,real}}{6} = [85.3, 128.0, 128.0, 85.3, 64.0, 42.6, 85.3] \\ K_{d,simulation} &= \frac{K_{d,real}}{6} = [4.26, 4.26, 4.26, 4.26, 4.26, 4.26, 4.26] \end{aligned} \tag{23}$$

For hardware deployment, these gains are adjusted to $K_{p,hardware} = \frac{1}{25.6}\frac{K_{p,real}}{6}$ and $K_{d,hardware} = \frac{1}{0.0128}\frac{K_{d,real}}{6}$.

Figure 35 and 36 display the computed and measured arm torques for an open-loop trajectory on the Z1 arm. The measured torques are retrieved from Unitree interface, while the computed torques are determined using Equation 20 with gains set to $K_p = K_{p,hardware}$ and $K_d = K_{d,hardware}$. These torques are sent in FF mode in Figure 35 and in PD mode in Figure 36. In both figures, the observed and desired torques are closely aligned and hence the torque model derived in this thesis reflects reality.

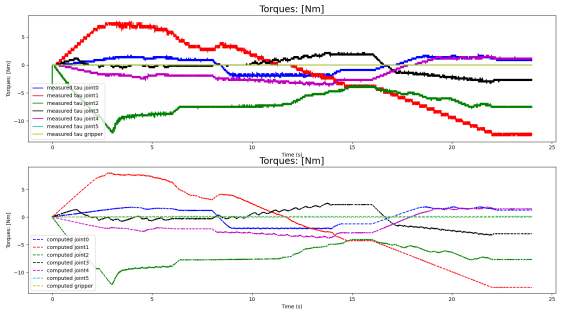


Figure 35: Arm joint torques commanded to follow an open loop trajectory and controlled through Unitree's FeedForward control mode.

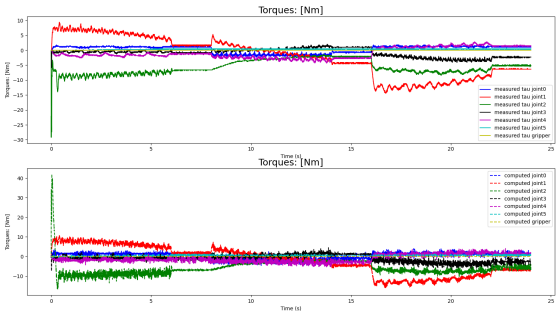


Figure 36: Arm joint torques commanded to follow an open loop trajectory and controlled through Unitree's PD control mode.

Training policies with high gains in Isaac Gym

The last subsection emphasized the need to control the arm with higher gains. However, there is challenge to train policies with such high gains in Isaac Gym. Indeed, when applying these in torque control mode, the joint velocity and torque estimates from the simulator are completely inaccurate. Figure 37 shows this behavior for an open-loop trajectory.

It has been discovered that, within Isaac Gym, there is a built-in PD controller able to track target joint positions and velocities. This controller utilizes the *stiffness* and *damping* coefficients from the agent's URDF file, where the *stiffness* coefficient acts as the p-gain and the *damping* coefficient as the d-gain. This mode resembles the PD controller of Unitree, since they both run at a high frequency. By adopting this control mode, for the same open-loop trajectory and high gains as in the previous example, the joint velocity and torque estimates from the simulator are accurate as illustrated in Figure 38. Training the arm with this mode achieved similar results to the previously used torque control mode.

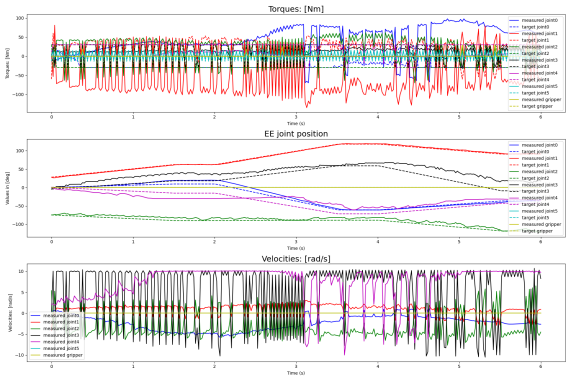


Figure 37: Open-loop trajectory using torque control with high gains in Isaac Gym.

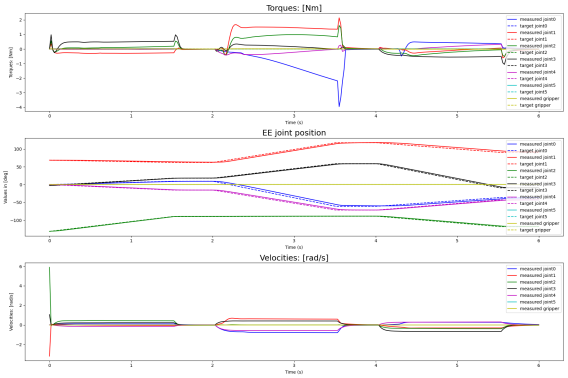


Figure 38: Open-loop trajectory using position control with high gains in Isaac Gym.

A last examination was conducted towards the effect of the *friction* coefficient defined for each joint in the arm's URDF file. In Unitree's default setting, this coefficient is set to 1.0 for every joint. It has been discovered that this coefficient increases the sim-to-real gap.

In order to study this effect, the same closed-loop trajectory test has been conducted in the Isaac Gym and Gazebo simulators and the real arm. For this final system identification process, the controller is a trained policy. Figure 39 shows the trajectory of the end-effector position commands in spherical coordinates synchronized for the 3 environments.

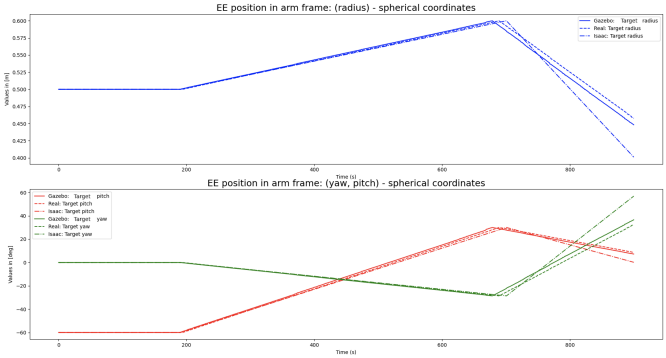


Figure 39: Trajectory of end-effector commands synchronized between the Isaac Gym and Gazebo simulators, and the real robotic arm.

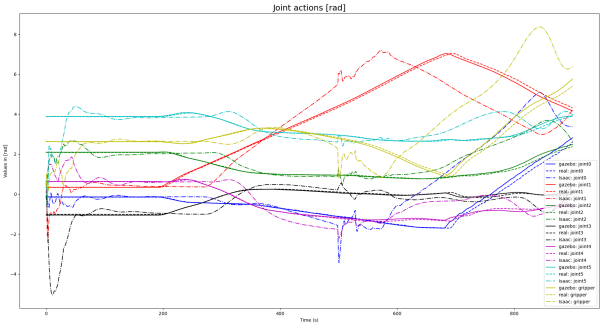


Figure 40: Joint actions from the Isaac Gym simulator do not align with the ones of the Gazebo simulator and the real robotic arm, when joints have non-zero friction coefficient.

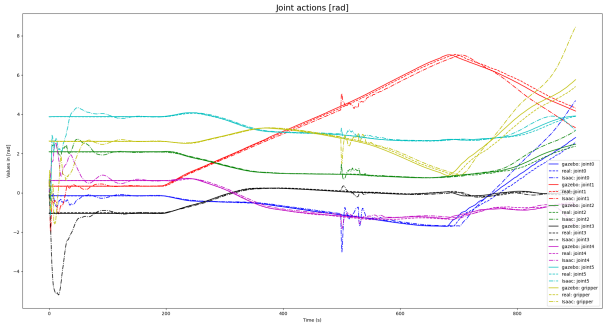


Figure 41: Joint actions from the Isaac Gym simulator align with the ones of the Gazebo simulator and the real robotic arm, when joints have zero friction coefficient.

In Figure 40, when the the friction coefficient is set to 1.0 for every degree of freedom of the arm, the Isaac Gym actions show a lag compared to the actions in Gazebo and the actual arm. Maintaining the friction to 0.0 for every joint solved this misalignment, as illustrated in Figure 41.

After these adjustments, a new policy has been trained. For this new controller, both the end-effector position tracking and the force estimate exhibited good performance in simulation and on the real hardware, as described in more details in the next section.

One noteworthy observation is that these sim-to-real adjustments not only improved the force estimate on hardware, but also increased the controller's robustness and stability. Indeed, before this sim-to-real investigation, if a force was applied to the gripper, the arm would resist effectively. However, upon releasing the force, the arm would rebound. This behavior is no longer observed for the policies trained with the sim-to-real adjustments.

5.1.2 Z1 arm: end-effector position controller

In this section, the tracking performance of the end-effector position controller is assessed. Although the main objective of this work is the development of a whole-body controller for a mobile legged manipulator, this evaluation highlights the effectiveness of the current formulation for end-effector position tracking.

Table 10 presents the final weights and parameters of the reward functions used for training the end-effector position controller for the Z1 arm.

Term	Equation	Weight	Parameters
gripper position tracking	$\exp\{\text{-}\lvert(r,\theta,\phi)-(r,\theta,\phi)_{cmd}\rvert/\sigma\}$	0.5	$\sigma = 0.25$
joint angles violation	$\mathbb{1}_{q_i>p_q*q_{max}\|\|q_i<p_q*q_{min}}$	-3.0	$p_q = 0.9$
joint velocities	$\lvert\dot{\mathbf{q}}\rvert$	-8e-4	-
joint accelerations	$\lvert\ddot{\mathbf{q}}\rvert^2$	-3e-7	-
action smoothing	$\lvert\mathbf{a}_t-\mathbf{a}_{t-1}\rvert^2$	-0.05	-
action smoothing, 2nd order	$\lvert\mathbf{a}_t-2\mathbf{a}_{t-1}+\mathbf{a}_{t-2}\rvert^2$	-0.02	-

Table 10: Reward function and parameters for the Z1 arm end-effector controller.

In what follows, there is a presentation of quantitative end-effector tracking performance in simulation, a teleoperation demonstration of the arm opening a door and an estimation of the gripper force on actual hardware. These results serve to illustrate the potential of the current approach to be extended for the whole-body platform.

Simulation results

Figure 42 displays the front and side view of an end-effector trajectory for the Z1 arm in simulation. This trajectory includes a variety of commands within the training distribution. The associated tracking performance, expressed in spherical coordinates, can be viewed in Figure 43. Given the close alignment between the measured and commanded curves, the reward formulation and task sampling applied to this policy during training are confirmed to be effective.

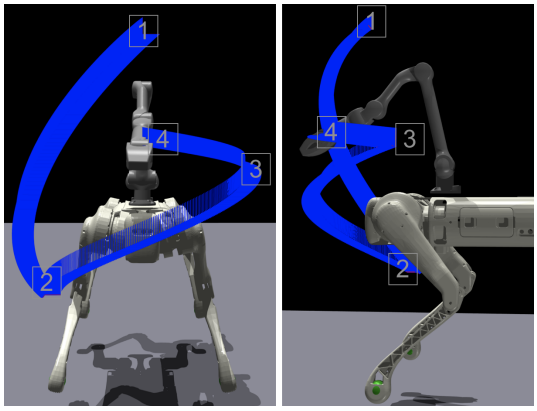


Figure 42: Example of an end-effector trajectory tracking for the Z1 arm in simulation.

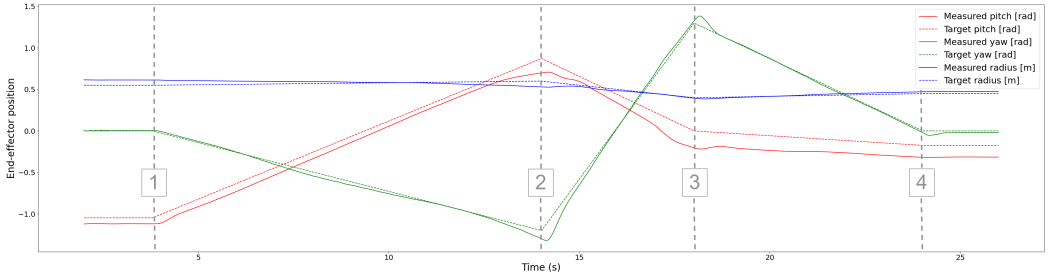


Figure 43: End-effector tracking performance expressed in spherical coordinates for the trajectory illustarted in Figure 42.

Hardware results

This policy has been tested on the hardware. Figure 44 depicts the successful teleoperation of the Z1 arm as it opens a door. On the right side of the figure, the z-axis force estimate from the *Estimator* network is reported. As one can see, the estimate exhibits considerably less noise compared to the policies previously discussed in section 5.1.1. Figure 45 illustrates the response of the arm when commanded to maintain a constant end-effector position (step 1) while subjected to an external force along the z-axis. Upon the external force release, the arm smoothly returns to the commanded position without oscillations. Again, on the right side of this figure, the corresponding estimate of the force applied is provided. Interestingly, this estimate is derived only from action state history, relying on the joint encoders as sensors without the use of torque or force sensors. These results suggest that the same framework, namely the end-effector position tracking formulation and force estimator, has the potential to be effective for the legged manipulator in accomplishing forceful tasks.

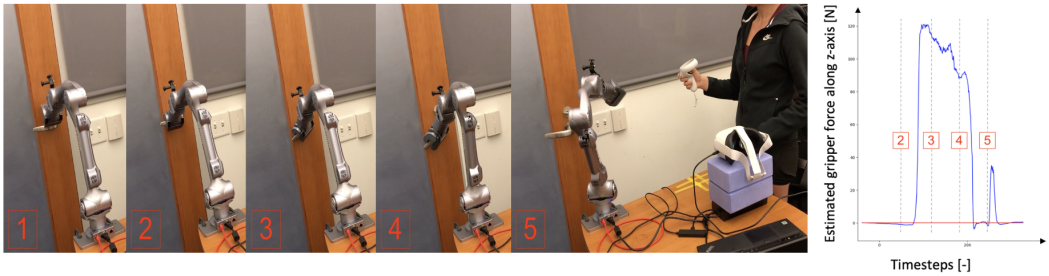


Figure 44: Time-sequence of the Z1 arm teleoperated to open a door using Oculus (left) and corresponding force estimate along the z-axis (right).

Figure 45: Time-sequence of the Z1 arm commanded to a constant end-effector position, while subjected to an external force along the z-axis (left) and corresponding force estimate (right).

5.2 B1 robot

In this section, the steps undertaken to bridge the simulation-to-reality gap for the B1 robot, are first presented. Then, the results of the stand-up policy in simulation and on hardware are described.

5.2.1 B1 robot: sim-to-real

Compared to the Z1 robotic arm, the transition from simulation to real-world deployment for the B1 robot was significantly easier. Two steps have been undertaken to bridge the simulation-to-reality gap:

1. A mismatch was observed between the URDF model of the B1 robot provided by Unitree upon the robot's delivery and the actual robot. For instance, differences in body masses and inaccurate limits in terms of effort, velocity, and position were noticed (e.g the base mass was set to 9kg in the URDF model, while the base weights around 25kg in reality). By investigating the Unitree public repository, a new version of this file, resembling more closely with reality, has been discovered.

2. On the real B1 robot, the low-level joint controller incorporates a power protection function preventing motor damage. To have similar torque limits as in the datasheet, the power safety factor was increased from level 6 to 9.

The PD gains, used in both simulation and hardware, were retrieved from an example script provided by Unitree. The default proportional gains are: $K_{p,hip} = 70, K_{p,thigh} = 180, K_{p,calf} = 300$, while the default derivative gains are $K_{d,hip} = 3, K_{d,thigh} = 8, K_{d,calf} = 15$. These gains appeared to be well calibrated for the tasks of interest in this work in both simulation and reality. Hence, no further tuning was involved.

5.2.2 B1 robot: stand-up controller

Table 11 presents the final weights and parameters of the reward functions used for training the stand-up controller.

The total reward curve during training is depicted in Figure 46. As one can see, it stabilizes after 10000 iterations, marking the end of the training phase. The first training iterations

Term	Equation	Weights	Parameters
target joint positions tracking	$\exp\{-\|\|\mathbf{q} - \mathbf{q}_{target}\|\|_2^2/\sigma_q\}$	2.0	$\sigma_q = 0.01$
body height tracking	$\exp\{-(h - h_{target})^2/\sigma_h\}$	3.0	$\sigma_h = 1.0$
feet contact forces	r_c	0.001	$\sigma_f = 0.01$, $F_{min} = 30$, $F_{max} = 250$
thigh/calf collision	$\mathbb{1}_{collision}$	-5.0	-
joint angle limits violation	$\mathbb{1}_{\mathbf{q}>\mathbf{q}_{max}\|\|\mathbf{q}<\mathbf{q}_{min}}$	-3.0	-
joint torque limits violation	$\mathbb{1}_{\|\boldsymbol{\tau}\|>p*\boldsymbol{\tau}_{max}}$	-0.1	$p = 0.8$
joint velocities	$\|\dot{\mathbf{q}}\|$	-0.0008	-
joint accelerations	$\|\ddot{\mathbf{q}}\|^2$	-3.0e-07	-
action smoothing	$\|\mathbf{a}_t - \mathbf{a}_{t-1}\|^2$	-0.05	-
action smoothing, 2nd order	$\|\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\|^2$	-0.02	-

Table 11: Reward function and parameters for the B1 stand-up controller

observed lower rewards, due the high number of environments terminated on body orientation, further detailed in Figure 79 of the Appendix. Additional task reward training curves from Table 11, can be found in Figure 80 in the Appendix.

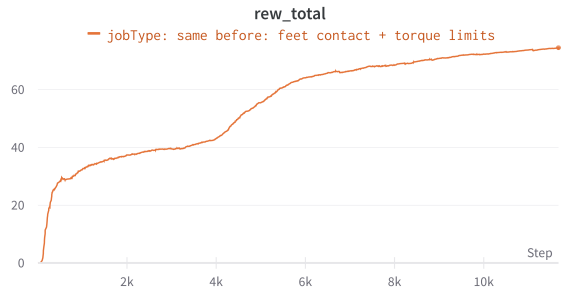


Figure 46: Stand-up policy training: total reward

Figure 48 shows the successful stand-up of the robot in simulation and the corresponding time evolution of the leg torques are depicted in Figure 47. As one can see, these torques do not exceed the torque limits and hence this policy has been deployed on hardware, as illustrated in Figure 49. For the hardware deployment, the robot's leg joint angles are first controlled to reach to the initial positions utilized during training using open-loop control. Then, the stand-up policy is initiated with exponential clipping of the first actions for a smoother response. The seated transition is achieved through interpolation between the standing and the seated joint configurations.

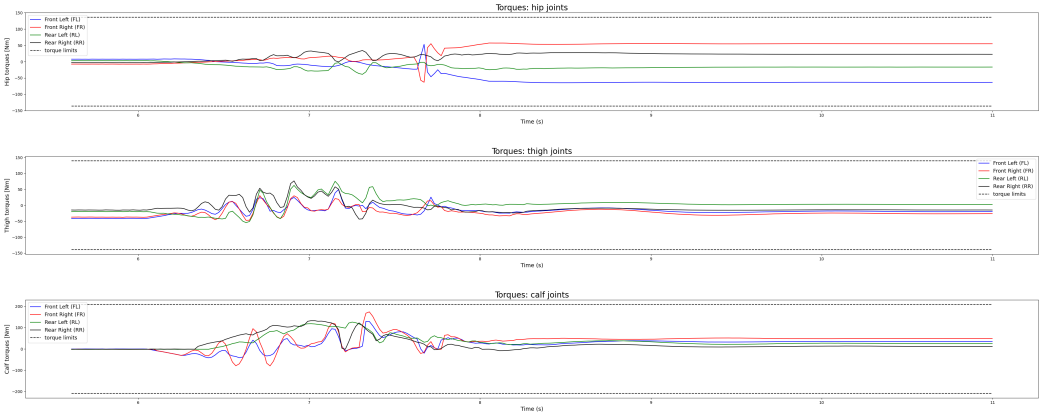


Figure 47: Leg torques when standing-up in Isaac Gym.

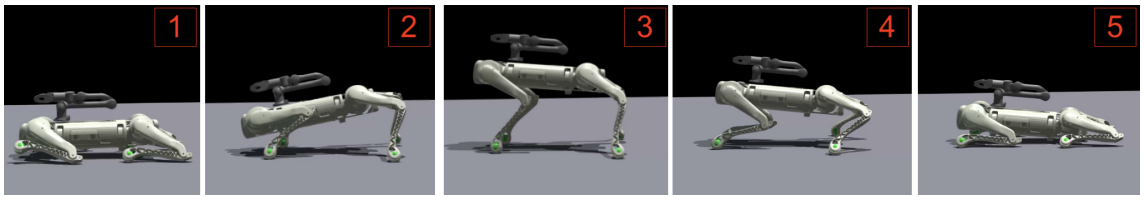


Figure 48: Isaac Gym simulation: stand-up policy and open-loop seating.

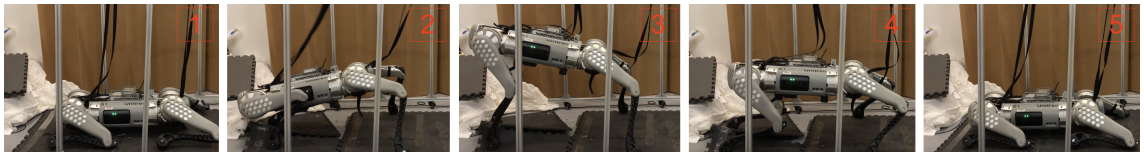


Figure 49: Hardware: stand-up policy and open-loop seating.

5.3 Whole-body position controller

In this chapter, the final parameters employed to train the whole-body position controller are first presented. Then, the controller's performance is assessed in simulation and on hardware.

Table 12 displays the final weights and parameters of the reward functions used to train the end-effector position and locomotion controller.

Term	Equation	Weight	Parameters
gripper position tracking	$\exp\{-\lvert(r,\theta,\phi)-(r,\theta,\phi)^{cmd}\rvert/\sigma_p\}$	5.0	$\sigma_p = 0.5$
x velocity tracking	$\exp\{-\lvert v_x - v_x^{cmd}\rvert^2/\sigma_{vx}\}$	1.0	$\sigma_{vx} = 0.25$
y velocity tracking	$\exp\{-\lvert v_y - v_y^{cmd}\rvert^2/\sigma_{vy}\}$	1.0	$\sigma_{vy} = 0.25$
yaw velocity tracking	$\exp\{-\lvert\omega_z - \omega_z^{cmd}\rvert^2/\sigma_{\omega z}\}$	1.0	$\sigma_{\omega z} = 0.25$
swing phase tracking (force)	$\sum_{foot}\left[1 - C_{foot}^{cmd}(t)\right]\exp\{-\lvert\mathbf{f}^{foot}\rvert^2/\sigma_{cf}\}$	0.9	$\sigma_{cf} = 4.0$
stance phase tracking (velocity)	$\sum_{foot}\left[C_{foot}^{cmd}(t)\right]\exp\{-\lvert\mathbf{v}_{xy}^{foot}\rvert^2/\sigma_{cv}\}$	4.0	$\sigma_{cv} = 4.0$
collision penalty	$\mathbb{1}_{collision}$	-5.0	-
arm joint limit violation	$\mathbb{1}_{\mathbf{q}_a > p_a * \mathbf{q}_a^{max} \lvert\lvert \mathbf{q}_a < p_a * \mathbf{q}_a^{min}}$	-3.0	$p_a = 0.9$
leg joint limit violation	$\mathbb{1}_{\mathbf{q}_l > p_l * \mathbf{q}_l^{max} \lvert\lvert \mathbf{q}_l < p_l * \mathbf{q}_l^{min}}$	-1.0	$p_l = 0.9$
joint velocities	$\lvert\dot{\mathbf{q}}\rvert$	-0.0008	-
joint accelerations	$\lvert\ddot{\mathbf{q}}\rvert^2$	-3.0e-07	-
action smoothing	$\lvert\mathbf{a}_t - \mathbf{a}_{t-1}\rvert^2$	-0.05	-
action smoothing, 2nd order	$\lvert\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\rvert^2$	-0.02	-

Table 12: Weights and parameters of the reward function for the whole-body controller.

The total reward curve during training is depicted in Figure 50. As one can see, it stabilizes after 3000 iterations, marking the end of the training phase. Additional task reward training curves from Table 12, can be found in Figure 89 in the Appendix.

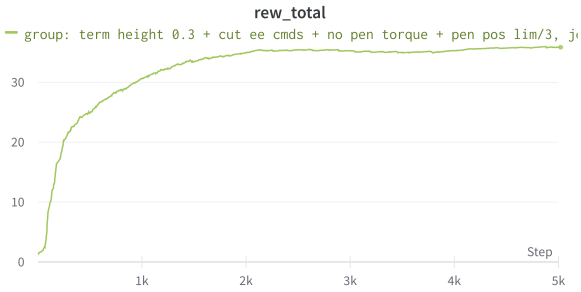


Figure 50: Whole-body position controller: total reward

5.3.1 Simulation results

In this section, an example of a trajectory that tracks both base velocity and end-effector commands simultaneously is provided. Subsequently, a thorough assessment of the end-effector tracking precision is conducted when the base remains static, a scenario frequently observed in most teleoperations.

Whole-body tracking performance

Figure 51 shows the trajectory tracking for the base and end-effector commands. Notably, while the tracking is generally accurate, when the velocity commands are non zero, the gripper

experiences minor oscillations at the stepping frequency, as seen in the small fluctuations of the radius, pitch and yaw curves.

It's worth noting that the yaw angle tracking performs exceptionally well, primarily because the first joint (i.e J1 in figure 6) can handle this task independently. In contrast, the other two spherical coordinates, namely radius and pitch angle, do not display as precise behavior since they require the coordination of all platform's joints, including legs and arm.

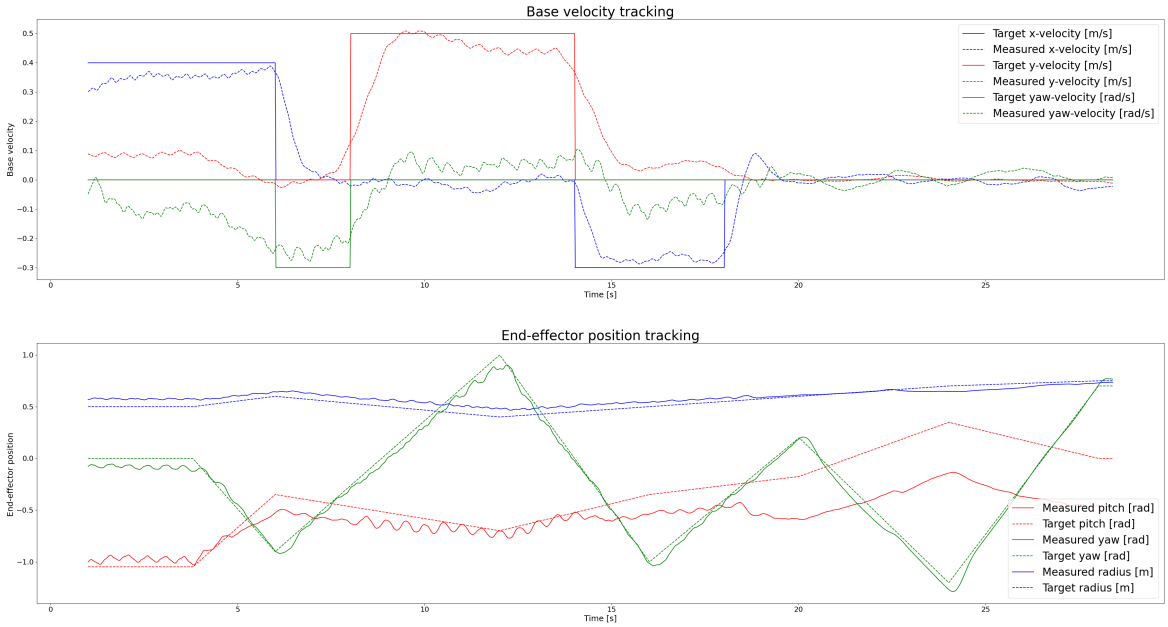


Figure 51: Example of a trajectory that tracks both base velocity and end-effector commands simultaneously for the whole-body position controller.

Figure 90 in the Appendix illustrates the time evolution of the leg and arm torques alongside the torque limits for each joint. Observably, the torques never exceed their respective limits and appear to be considerably below them. This distance from the limits in torque space serves as the primary safety measure used in this thesis: if the torques remain well within the limits, the controller is deployed on hardware, otherwise it is not. The results of this controller's hardware deployment are given in section 5.3.2.

End-effector tracking performance

To grasp objects for manipulation, the policy for end-effector position control should be accurate. In this subsection, the end-effector tracking performance with a static base has been thoroughly assessed in simulation. A set of 2200 end-effector position commands, uniformly sampled from the training distribution, have been tested simultaneously. In this parallel evaluation, illustrated in Figure 52, each agent tracks one of these constant commands for 5 seconds. During this period, the end-effector position error in spherical coordinates is collected and averaged. From this dataset, the error distribution for the radius, pitch and yaw errors were derived, as depicted in Figure 53.

In assessing the accuracy of the static end-effector position controller, it is instructive to identify the range in which 90% of the errors occur:

- For the radius, 90% of the errors fall within a range of 0.5 to 11cm.
- For the pitch angle, 90% of the observed errors lie between 0.55 and 10.82 degrees.
- For the yaw angle, 90% of the observed errors are between 0.79 and 13.96 degrees.

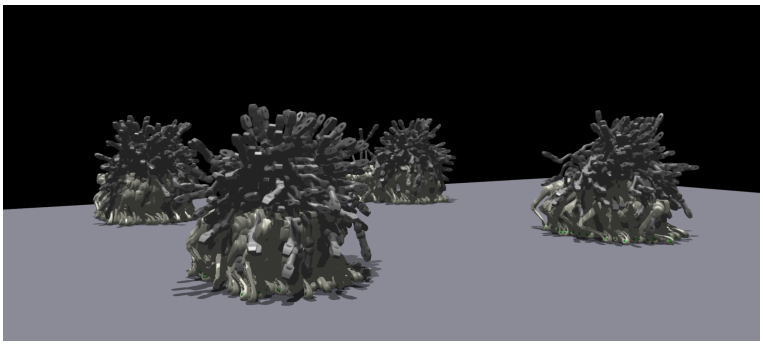


Figure 52: Parallel evaluation of the end-effector tracking for static base in Isaac Gym. Each agent tracks one of the 2200 end-effector position commands uniformly sampled from the training distribution.

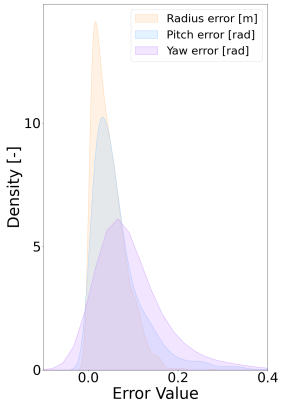


Figure 53: Distribution of the radius, pitch and yaw position error across 2200 end-effector position commands.

Workspace in simulation

To manipulate objects that are higher or lower than the robot's body, there is a desire to achieve a large manipulation workspace through coordination of the body and arm. Prior work has shown this for a much smaller robot [7].

In this subsection, the expanded workspace achieved by the whole-body controller in comparison to the arm-only configuration is evaluated. Firstly, the workspace of the arm alone is estimated and then an estimate of the whole-body controller is computed. The objective of this comparison is to demonstrate that leg actions increase the arm's reachability.

Z1 arm workspace

An estimate of the arm's workspace is derived by randomly sampling 3000 arm joint configurations for the first five joint angles. The final estimate is constituted of the end-effector positions resulting in no arm collision. Figure 54 illustrate the front and side views of this arm workspace. In the visualization, green points denote feasible end-effector positions, whereas red indicates end-effector positions resulting in arm collision. A representation of the arm's workspace, composed of solely feasible points, is shown in Figure 55.

In order to compare the workspace of the trained policy with the arm's workspace, the latter is truncated to align with the former's yaw dimension training distribution. Figure 56 shows the front and top perspectives of the arm workspace segmented by the yaw angle training distribution. In this figure, the green dots denote those falling within the training distribution, whereas the red ones highlight those outside.

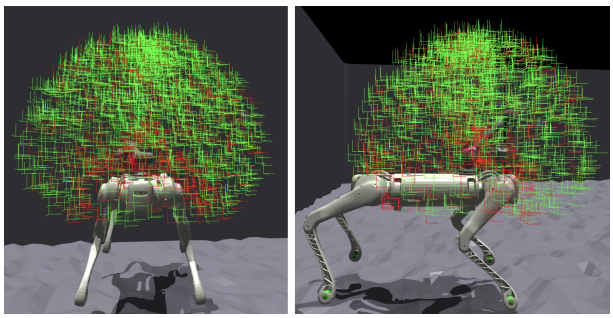


Figure 54: Front and side views of the arm workspace: green for feasible and red for collision from 3000 sampled arm configurations.

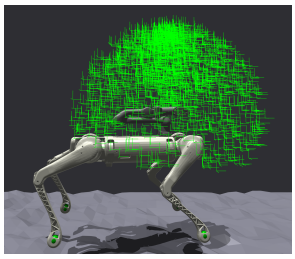


Figure 55: Side view of the arm workspace: all feasible configurations in green.

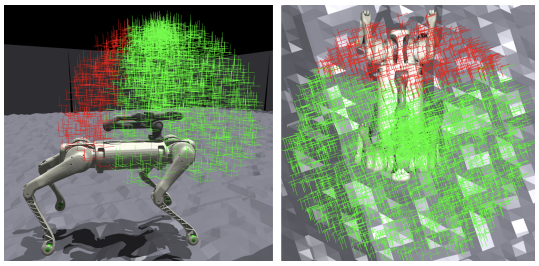


Figure 56: Front and top views of the arm workspace segmented by the yaw angle training distribution: green for inside and red for outside.

Whole-body controller workspace

To estimate the workspace of the whole-body controller, a trajectory of end-effector commands from the training set is tracked. To maximize this estimate, the radius command is fixed to the maximum radius value of the training distribution, specifically 0.9 m. Figure 57 shows a chronological representation of this trajectory. This trajectory clearly demonstrates how the leg movements enhance the arm's reachability. Indeed, it is evident that the legs bend to accommodate lower end-effector commands, and they extend to facilitate higher end-effector commands.

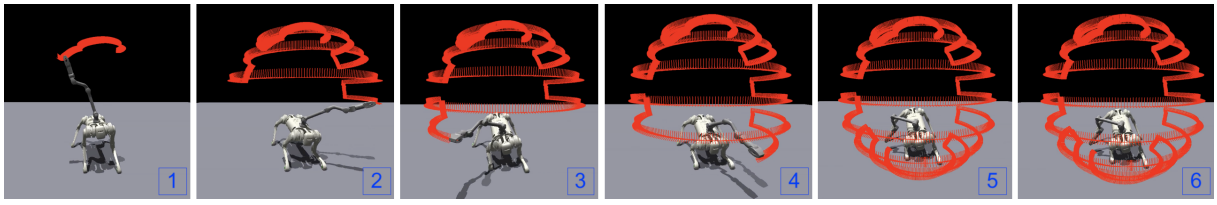


Figure 57: Chronological representation of the trajectory used to estimate the maximum workspace achieved by the whole-body controller in simulation.

To numerically compare the workspace gains of the whole-body controller against the arm-only configuration, a convex hull is computed for each case. Figure 58 illustrates the convex hull of the whole-body controller in red and the one of the arm-only setup in green, including the specific points that defined each hull. For clearer visualization the two-dimensional projections of these hulls are shown in Figure 59.

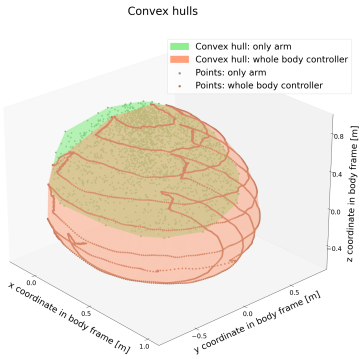


Figure 58: 3D visualization

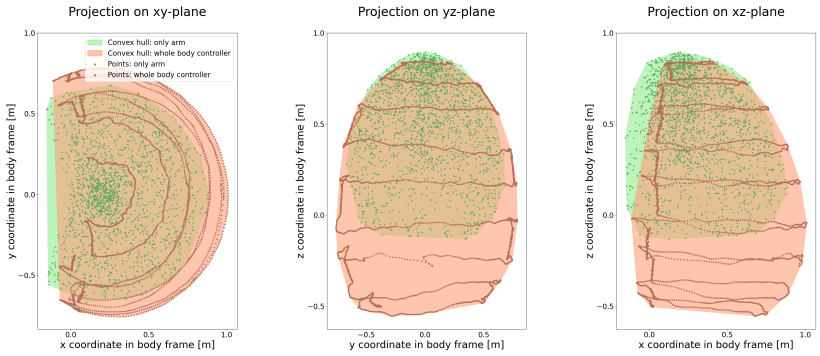
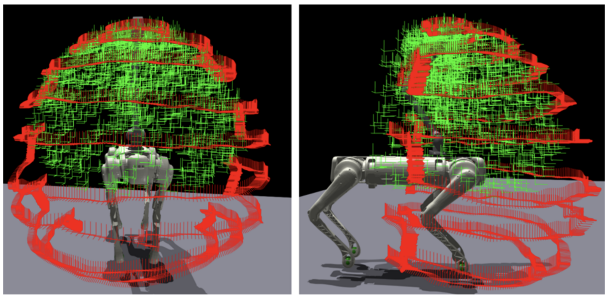


Figure 59: 2D projections

Figure 60: Whole-body controller's convex hull (red) vs. arm-only configuration's convex hull (green)

Finally, by computing and comparing the volume of each convex hull, the workspace enhancement attributed to the whole-body controller is found to be +71.5%. This considerable increase is illustrated in Figure 61, where both workspaces are illustrated on the same graph.



Controller type	Convex hull volume [m^3]	
Arm controller	0.81	+ 71.5%
Whole body controller	1.39	

Figure 61: Whole-body controller's maximum workspace (red) and arm-only configuration's workspace (green) with their respective convex hull volumes.

5.3.2 Hardware deployment

In this section, first, the end-effector tracking accuracy is quantified using a motion capture system. Subsequently, various tasks accomplished by teleoperating the legged manipulator with Oculus are presented. These showcase the capability of this controller to support walking to objects and grasping them, to then apply a force utilizing the force controller.

End-effector tracking performance

The end-effector tracking performance is evaluated by teleoperating the arm across a variety of end-effector position commands within the training distribution for a static base velocity. During this process, the trajectory of commands is recorded, as well as the trajectory of end-effector positions, captured via a motion tracking system.

The trajectory covers a wide spectrum of the training range, encompassing commands with both small and large radii, pitch and yaw angles. The path, expressed in cartesian coordinates,

can be seen in Figure 62. As shown, the commanded and actual end-effector positions align closely. Note that the irregularities observed in the measured curves are due to the motion capture system.

For this trajectory the average positional errors on the x, y and z axes are 4.42 cm, 5.37 cm, and 6.86 cm respectively.

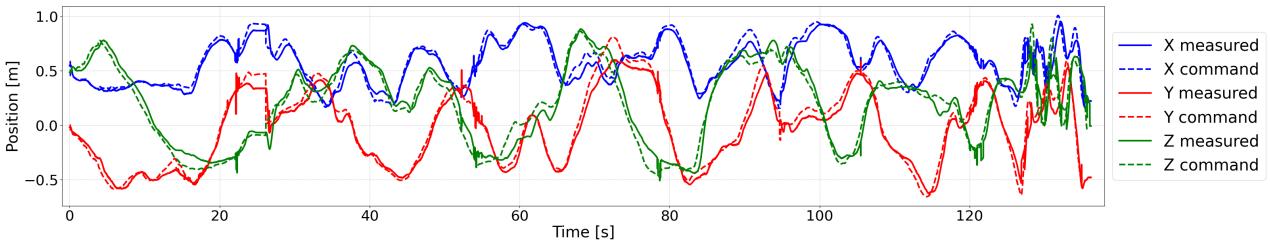


Figure 62: End-effector position tracking performance on hardware.

The same trajectory can also be represented in spherical coordinates, as shown in Figure 91 in the Appendix. For this representation, the average errors in radius, pitch and yaw coordinates are 5.38 cm , 0.11 rad , 0.1 rad. These errors closely align with the error distribution calculated in the simulation (refer to Figure 53 for details), indicating a minimal gap between simulation and real-world performance.

Teleoperated tasks execution with Oculus

Figure 63 and 64 display the successful execution of several tasks, including opening drawers, cabinets and doors, as well as pouring water, all achieved through teleoperation using Oculus.

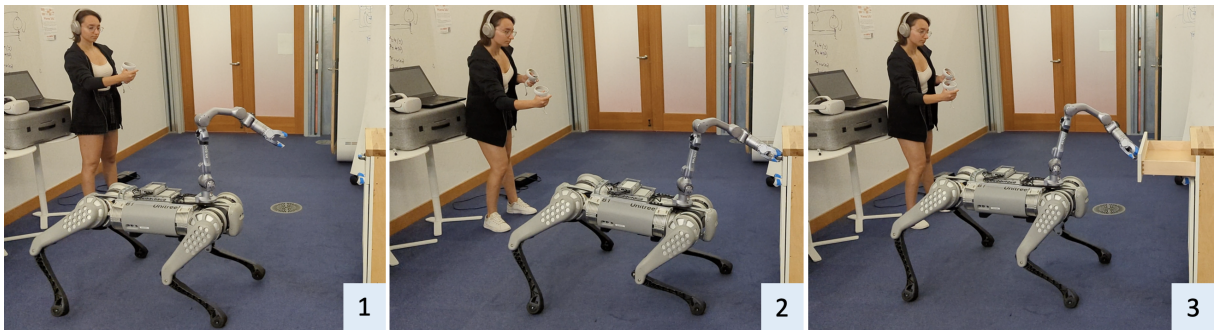


Figure 63: Teleoperation of the legged manipulator in action: an individual directing the platform to approach a drawer, grasp its handle, and open it by walking backwards, all while using the Oculus headset and joysticks.

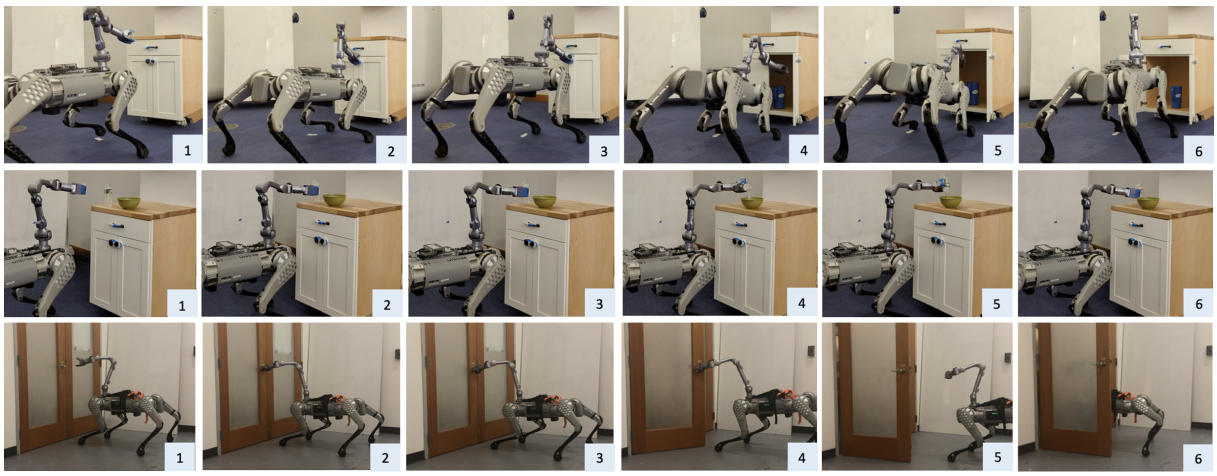


Figure 64: Successful tasks achieved through teleoperation of the legged manipulator. (Top) A time-sequence of the robotic system bending down to open a cabinet, illustrating the advantages of whole-body control. (Middle) Another time-sequence display, showcasing the precision of the system as it pours water from a bottle into a bowl. (Bottom) The robot successfully opens a spring-loaded door, indicating its capability to undertake forceful tasks due to its inherent strength.

5.4 Whole-body force controller

In this section, the force tracking and estimation performance of the whole-body force controller is assessed in simulation. Then, the applications of force control, including compliant tasks and force application supported by data from deployment on the real robot, are described. Two separate learned force controllers are evaluated: one to regulate force in the z-axis and another to regulate force in the xy-plane. The respective training curves for these controllers can be found in Figure 98 and 105 in the Appendix section.

Table 13 displays the final weights and parameters of the reward functions used to train the whole-body force controller.

Term	Equation	Weight	Parameters
force tracking	$\exp\{-\lvert\mathbf{F}_g^{cmd} - \mathbf{F}_g\rvert/\sigma_F\}$	5.0	$\sigma_F = 20.0$
arm torque limit violation	$\mathbb{1}_{\tau_a > p_a * \lvert\tau_a^{max}\rvert}$	-0.0015	$p_a = 0.8$
stance phase tracking	$\sum_i^4 \left[C_i^{cmd}(t)\right] \exp\{-\lvert\mathbf{v}_{xy}^i\rvert^2/\sigma_{cv}\}$	4.0	$\sigma_{cv} = 4.0$
collision penalty	$\mathbb{1}_{collision}$	-5.0	-
arm joint limit violation	$\mathbb{1}_{\mathbf{q}_a > p_a * \mathbf{q}_a^{max} \lvert\lvert \mathbf{q}_a < p_a * \mathbf{q}_a^{min}}$	-3.0	$p_a = 0.9$
leg joint limit violation	$\mathbb{1}_{\mathbf{q}_l > p_l * \mathbf{q}_l^{max} \lvert\lvert \mathbf{q}_l < p_l * \mathbf{q}_l^{min}}$	-1.0	$p_l = 0.9$
joint velocities	$\lvert\dot{\mathbf{q}}\rvert$	-8e-4	-
joint accelerations	$\lvert\ddot{\mathbf{q}}\rvert^2$	-3e-7	-
action smoothing	$\lvert\mathbf{a}_t - \mathbf{a}_{t-1}\rvert^2$	-0.05	-
action smoothing, 2nd order	$\lvert\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\rvert^2$	-0.02	-

Table 13: Weights and parameters of the reward function for the whole-body force controller.

5.4.1 Simulation results

When lifting or pulling objects, an effective controller should realize a large applied force without undesired transients or inefficient postures that would result in the motor exceeding its safety limits. To evaluate this characteristic, the average force tracking and estimation errors for each target force across all training setpoints are reported in Figure 65. For z-axis force control, the mean absolute tracking error is below 5 N for low force targets and increases at high forces, while the estimation error constantly falls below 5 N.

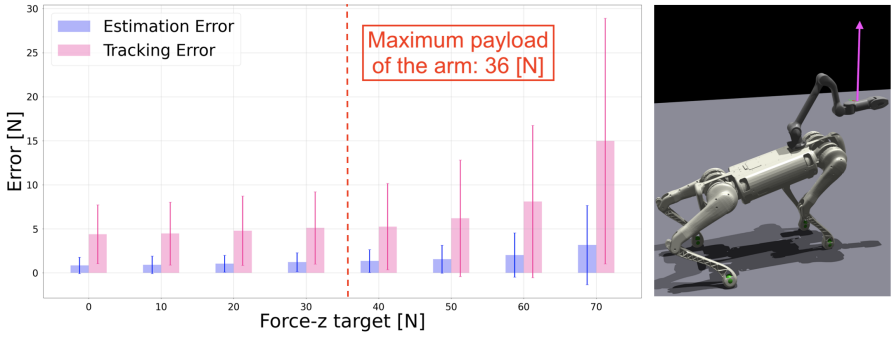


Figure 65: Bar plot illustrating the average force tracking and estimation errors for each target force across all training setpoints. The accompanying figure presents a robot with an external force applied at the gripper along the vertical axis, exemplifying the practical implications of these tracking errors.

Figure 66 illustrates the force tracking discrepancies in simulation across all training setpoints for three target vertical forces: 20, 40, and 60 N. In the heatmap, elevated tracking errors are depicted in red, while optimal tracking is shown in green. Notably, as the desired force increases, the tracking accuracy decreases, especially for points close to the ground.

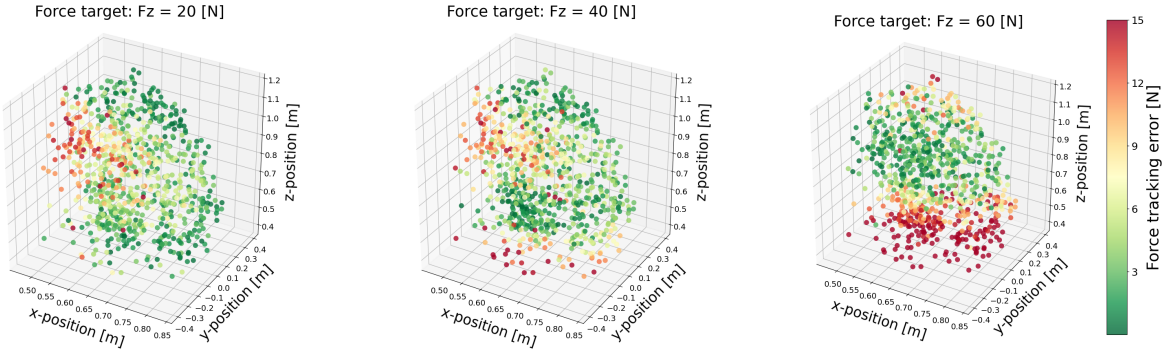


Figure 66: Visualization of force tracking errors in simulation across all training setpoints for three target vertical forces: 20, 40, and 60 N. In the heatmap, elevated tracking errors are depicted in red, while optimal tracking is shown in green.

5.4.2 Hardware deployment

To evaluate this policy on the real platform, the robot's end effector is attached to a rope and the downward force command is gradually increased through the entire training range (0-70 N). A dynamometer is used to record the applied force across five trials with the gripper in high, low, left, right, and middle positions. Figure 67 shows the setup for the gripper in the middle position, as well as the tracking performance of the vertical force controller. Notably,

the applied force is within one standard deviation of the simulated error distribution (Figure 65). The estimated force tends to overshoot, suggesting a moderate sim-to-real gap.

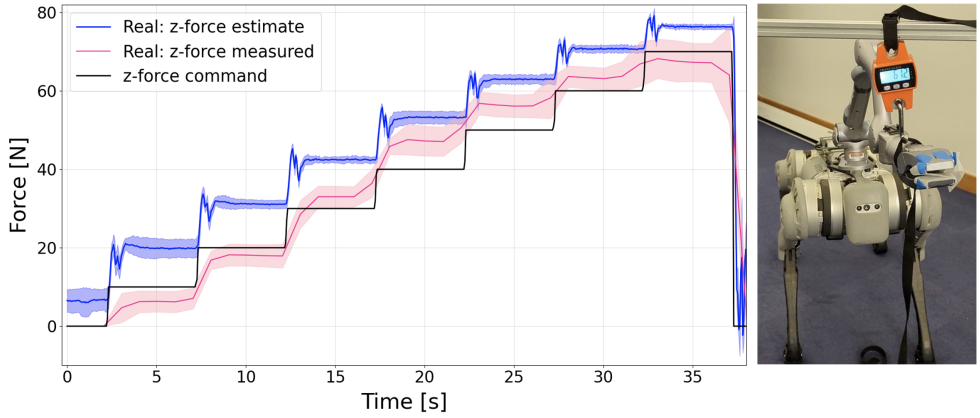


Figure 67: The effectiveness of the end-effector force controller is captured in its interaction with a dynamometer, with a view of the actual setup (right) and a graphical representation of the measured, commanded and estimated force over time (left), where the solid lines represent the means across five set points and the shaded regions indicate one standard deviation from the means.

To evaluate whether the controller can coordinate the body with the legs to increase the applied force, the arm has been initialized directly in front of the robot, the force command in the x-axis has been ramped up, and the highest applied pulling force has been recorded. Figure 68 illustrates this experiment, where the observed maximum force reached 90N. This value is a significant 150 % increase over the arm's rated payload of 36 N. As one can see, when pulling a heavy object, the robot does not only rely on its arms, which can only exert a limited force by itself. Instead, it optimally make use of its entire body to maximize the applied force. In the scenario presented in Figure 68, the robot adopts a pose where the arm is straight, allowing the legs to generate the majority of the force required for pulling. This approach not only increase the maximum force the system can apply, but also ensures the stability and balance of the robot during the force application.

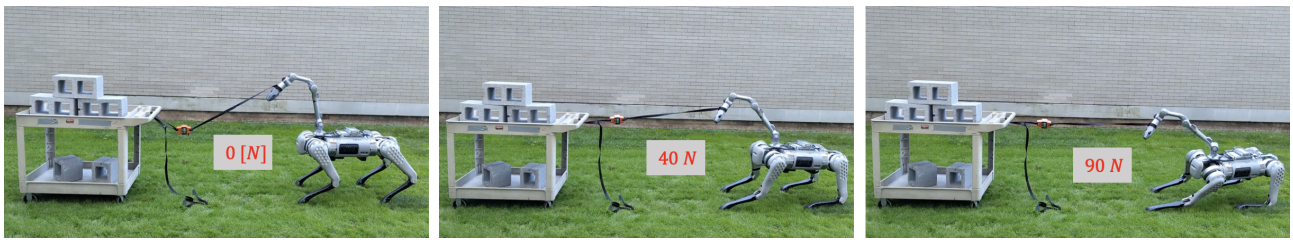


Figure 68: Robot pulling a heavy cart, sequentially exerting 0N (left), 40N (center), and peaking at 90N (right), surpassing its arm's payload by 150%.

Whole-body compliance

When the force controller is commanded to apply zero force, this corresponds to a fully compliant mode where the posture of the body and arm coordinate to drive the gripper force application to zero in all three axes. When released, the gripper remains suspended in place, with the system exhibiting gravity compensation, as illustrated in Figure 69.

Figure 69: Robot in compliant mode, with coordinated posture ensuring the gripper's steady position.

Compliant manipulation of heavy objects

By modulating the force application command in the z-axis, the compliant mode can be extended to the scenario where the robot is lifting an object with its arm. The compliant mode has been tested with payloads of 0kg, 2kg, 4kg, 6kg. With a command force of zero, nonzero payloads cause the gripper to sink to the ground, which is expected since it should not apply a resistive force, as shown in Figure 70 (A-1 to A-3). With a zero force command, it takes substantial force for the human to lift the gripper with the object in grasp because the gripper will not apply any lifting force to the grasped object. Next, the vertical force command is increased to match the payload weight. For payloads up to 4kg, this resulted in restored gravity compensation against the payload as illustrated in Figure 70-B.

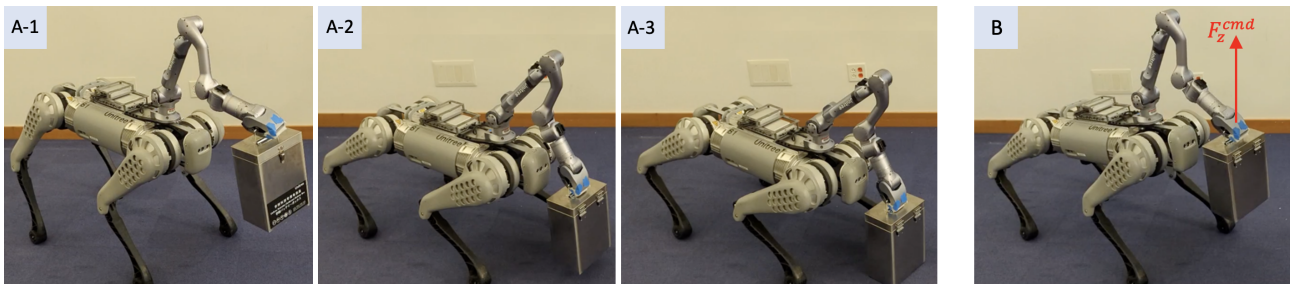


Figure 70: Response of the gripper to different force commands under a payload of 4kg. With zero force command, the gripper sinks to the ground under the payload (A-1 to A-3). When the force command is adjusted to match the payload weight, gravity compensation is evident (B).

With the force controller compensating for gravity, the gripper can then be effortlessly moved around, displaying compliance in all axes, as shown in Figure 71. With the highest payload of 6kg, the gripper is less compliant and drifts to the center of the robot when released to the side.

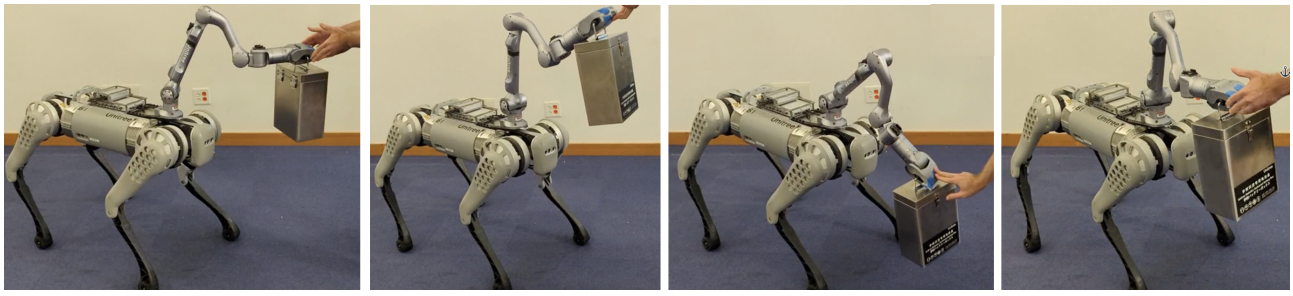


Figure 71: Demonstration of the force controller's gravity compensation capability when the force command is set to 40N and the payload weight is 4kg. The gripper exhibits compliance in all directions: above, left, right, and below the robot's base.

6 Discussion

This research has demonstrated that learned whole-body manipulation policies can perform force control at the end-effector using only the minimal sensor configuration of joint encoders and body IMU. The results also support that fully compliant behavior is possible using the standard reinforcement learning architecture, which has not been previously shown. Such compliance in learned motor policies is likely to improve the safety of robots around humans and robustness against unexpected disturbances. A practical implication being a running quadruped when it collides its gripper with an obstacle, could greatly reduce damage to itself or the environment by using this compliant mode. This compliant behavior also enables kinesthetic teaching, in which a human operator directly manipulates a robot's limbs in order to demonstrate a task without writing code or learning to use a teleoperation interface.

Furthermore, it has been demonstrated that force tracking performance is sufficient for some force-controlled tasks, including kinesthetic teaching with a weight and whole-body pulling. In quantitative experiments, good accuracy of a learned force estimator and the force application command tracking have been shown, with tracking performance degrading when the applied force reaches double the arm's rated payload. The sim-to-real gap of the legged manipulator has also been characterized. In combination with the end-effector positioning and locomotion controller, this work provides a teleoperation framework that is suitable for teleoperation and kinesthetic teaching of forceful loco-manipulation tasks.

6.1 Future work

In this final chapter, limitations of the proposed framework are discussed, along with recommendations for future research directions.

One major limitation of the end-effector position and locomotion controller is its inability to control the end-effector's orientation, except for roll. This constraint significantly reduces the capability of the controller. For instance, it cannot grasp objects positioned at higher or lower levels since the gripper would be oriented upwards or downwards, respectively. An interesting direction for future research could involve incorporating gripper orientation into the reward function.

The existing framework supports both force and position control, yet they operate independently. In the future, it will be promising to explore hybrid control modes where the robot performs compliant trajectory tracking for improved teleoperation. An illustrative example would be directing force on a whiteboard while using a position controller to guide the gripper along the tangential plane.

While the current capabilities of the controllers developed in this project could technically be addressed by model-based methods like MPC, the current state of the proposed framework lays the groundwork for more advanced behaviors. These include navigating rough terrains or applying force on uneven surfaces through the gripper, where reinforcement learning is known to outperform model-based approaches.

Another interesting direction for future work is to not only apply forces through the gripper, but leverage the robot's entire body for force application using various contact points. Drawing inspiration from [14], which offers a method for selecting contact points from whole-body regions and controlling the corresponding force exerted, there's potential to adapt and integrate these techniques into the current framework.

While this research highlighted promising results, the tracking performance for both the end-effector position and locomotion controller, and the force controller is not perfect. These

imperfections may arise from the multi-objective nature of the tasks, the challenge in crafting suitable reward functions, or inadequate exploration. If the policy is sub-optimal with respect to the reward function, it indicates insufficient exploration, which could potentially be improved using methods leveraging intrinsic reward mechanisms. These might include strategies based on prediction errors between anticipated and actual states [53] or by utilizing random network distillation techniques that reward agents for exploring unfamiliar parts of the environment [54]. However, if the policy happens to be optimal with respect to the reward function, then the task specification should be improved. The methodology introduced by [55], proposes a novel learning framework for training robotic controllers using both rewards and constraints. This approach not only simplifies the traditional reward engineering process by emphasizing suitable constraint types tailored to an engineer's intent, but also demonstrates enhanced performance. Applying this technique to train the controllers of this research could yield promising results.

Finally, while the current system is teleoperated, there is potential to integrate it into imitation learning pipelines. This would enable the robot to autonomously accomplish compliant and forceful behaviors by learning directly from demonstrations, transitioning from manual control to autonomous operations.

7 Appendix

7.1 Z1 arm: position controller

Figure 72 shows the performance of the end-effector position tracking policy trained without privileged information. In this scenario, a constant end-effector position command is given in spherical coordinates (radius, pitch, yaw), while a constant vertical force is applied to the gripper vertically. Upon the force application, the measured radius, pitch and yaw, describing the position of the arm in the base frame, exhibit large oscillations.

On other other hand, Figure 73 shows the performance of the end-effector position tracking policy trained with the vertical force applied to the gripper as privileged information. In this example, there are clearly no oscillations when the force is applied.

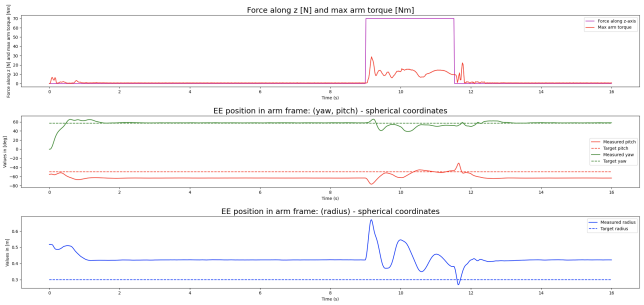


Figure 72: End-effector tracking policy without privileged information: external force applied, force estimate and end-effector tracking performance in spherical coordinates (radius, pitch, yaw)

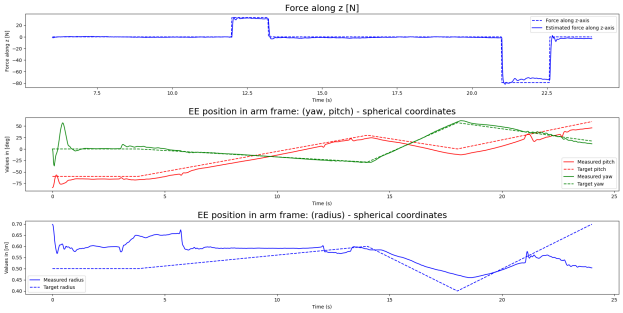


Figure 73: End-effector tracking policy with the vertical force applied to the gripper as privileged information: external force applied, force estimate and end-effector tracking performance in spherical coordinates (radius, pitch, yaw)

7.2 B1 robot: stand-up controller

Figure 80 reports some of the training curves of the task rewards defined in Table 11.

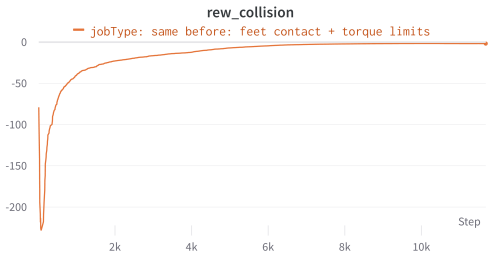


Figure 74: Thigh/calf collisions

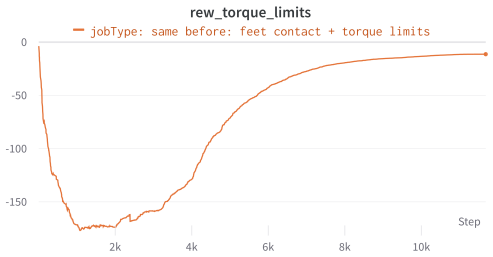


Figure 75: Leg torque limits violation

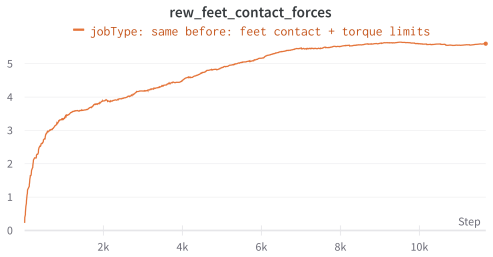


Figure 76: Feet contact forces

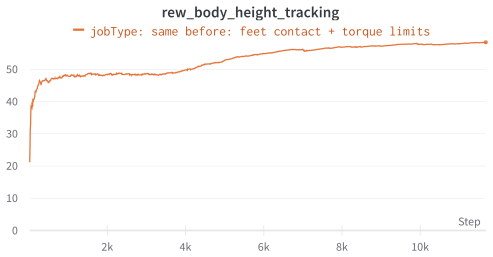


Figure 77: Body height tracking

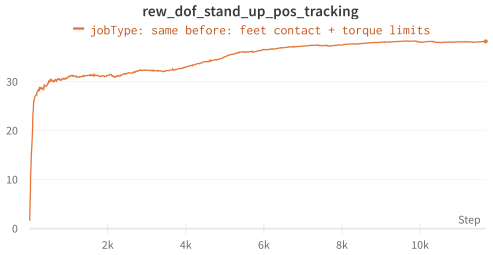


Figure 78: Joint positions tracking

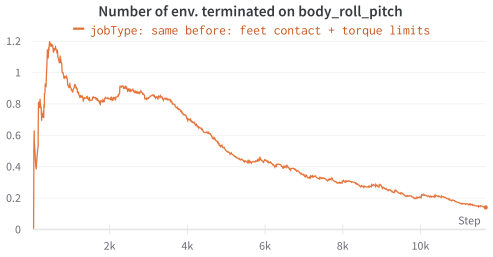


Figure 79: Number of environments terminated on body orientation

Figure 80: Stand-up policy training curves.

7.3 Whole body position controller

7.3.1 Training curves

Figure 89 reports some of the training curves of the task rewards defined in Table 12.

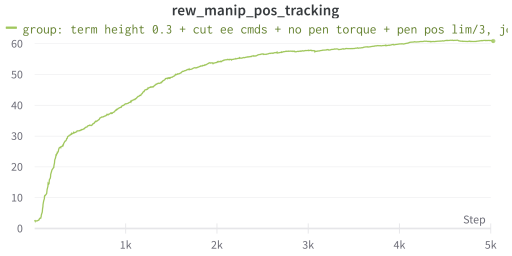


Figure 81: Gripper position tracking reward

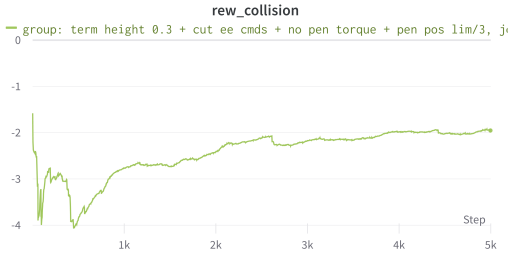


Figure 82: Thigh/-calf/link02/link03/link06 collision penalty

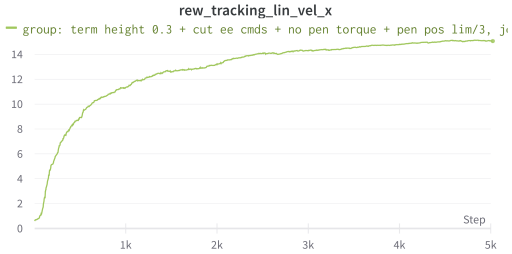


Figure 83: Linear velocity tracking reward along x-axis

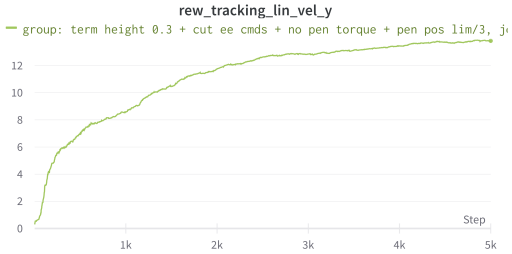


Figure 84: Linear velocity tracking reward along y-axis

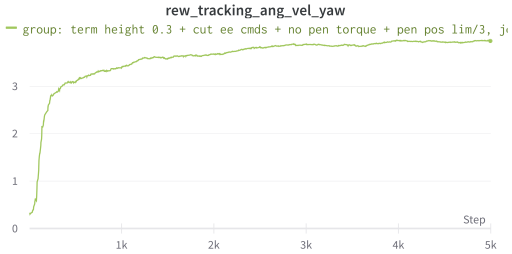


Figure 85: Angular yaw velocity tracking reward

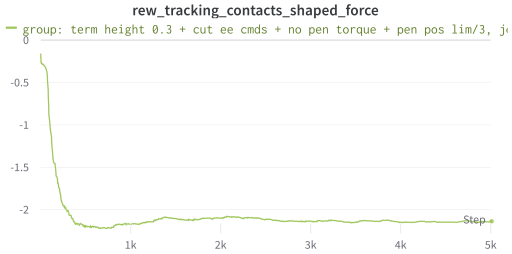


Figure 86: Swing phase tracking (force) penalty

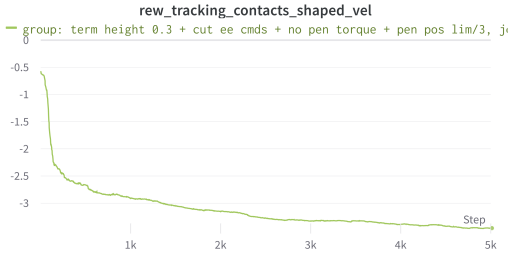


Figure 87: Stance phase tracking (velocity) penalty

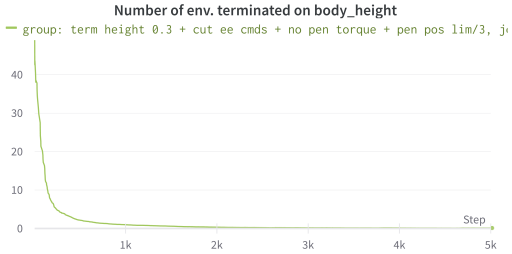


Figure 88: Number of environments terminated on body height

Figure 89: Whole-body position controller training curves.

7.3.2 Tracking performance in simulation

Figure 90 shows the leg and arm torques experienced by the legged manipulator during the trajectory of base velocity and end-effector commands illustrated in Figure 51.

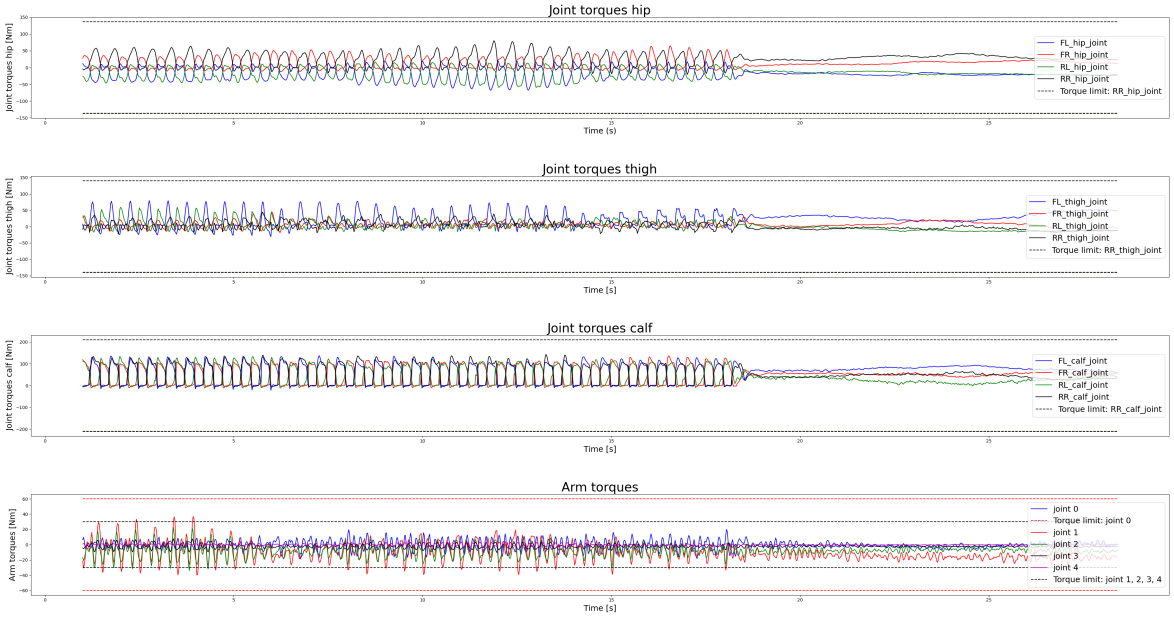


Figure 90: Leg and arm torques during the example of trajectory tracking presented in Figure 51.

7.3.3 Tracking performance on hardware

Figure 91 displays the end-effector trajectory, which is identical to the one shown in Cartesian coordinates in Figure 62, but represented in spherical coordinates.

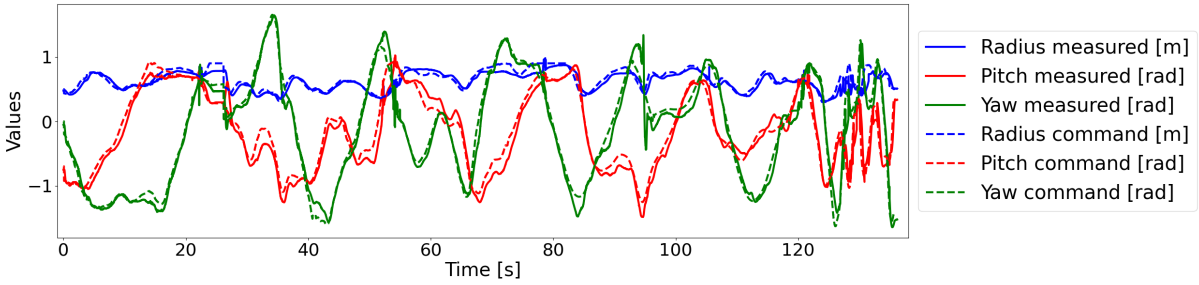


Figure 91: End-effector position tracking performance in spherical coordinates on hardware.

7.4 Whole-body force controller

Figure 98 shows the training curves for the whole-body force controller trained for a gripper force along the z-axis only.

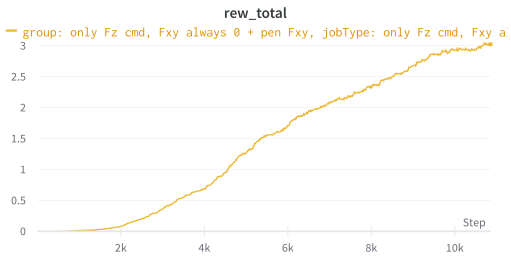


Figure 92: Total reward

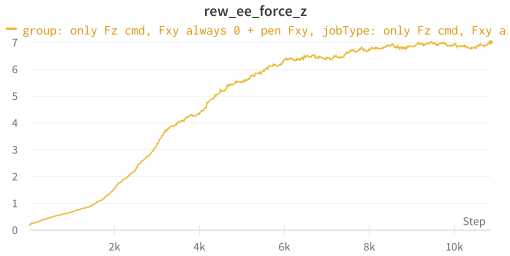


Figure 93: Force tracking reward along z-axis

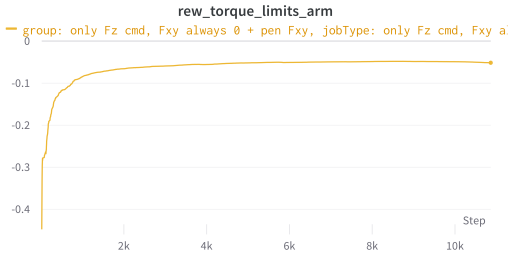


Figure 94: Arm torque limit penalty

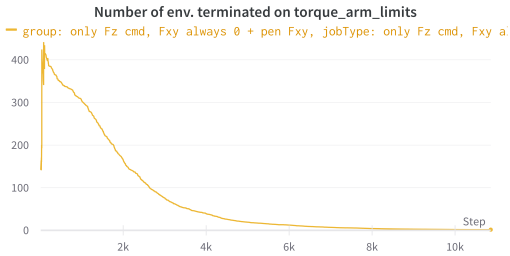


Figure 95: Number of environment terminated on arm torque limits

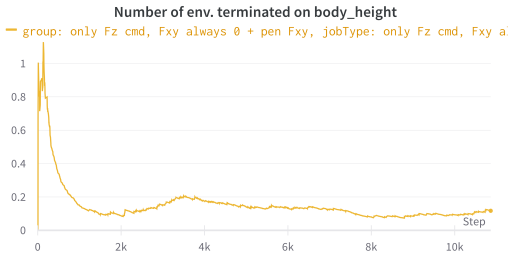


Figure 96: Number of environment terminated on body height

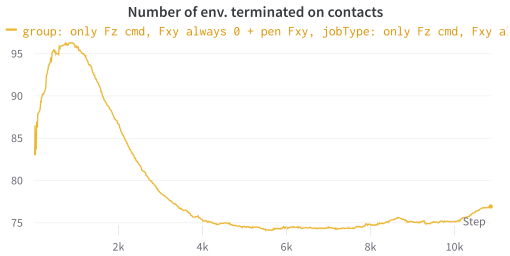


Figure 97: Number of environment terminated on contacts

Figure 98: Whole-body force controller training curves trained for a gripper force along the z-axis only.

Figure 105 shows the training curves for the whole-body force controller trained for a gripper force along the x- and y-axes only.

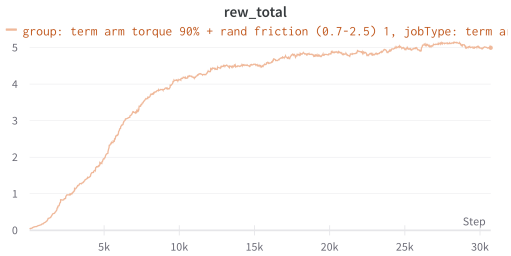


Figure 99: Total reward

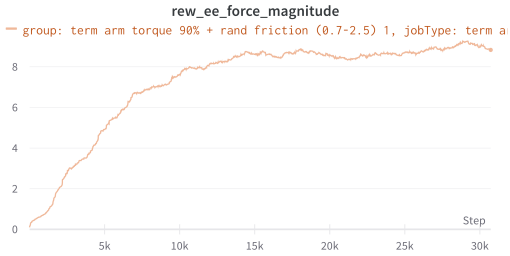


Figure 100: Force tracking reward along x- and y-axes

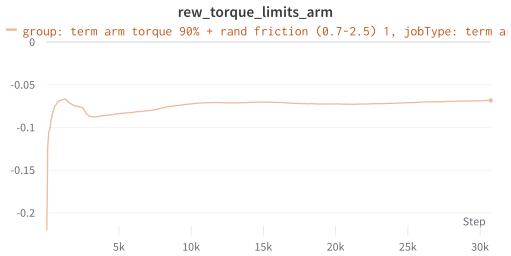


Figure 101: Arm torque limit penalty

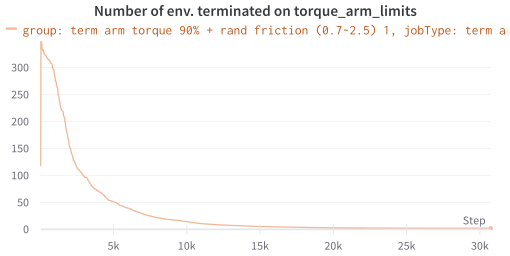


Figure 102: Number of environment terminated on arm torque limits

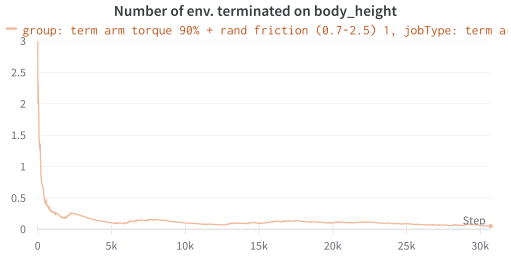


Figure 103: Number of environment terminated on body height

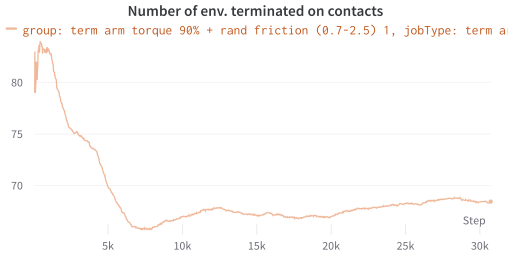


Figure 104: Number of environment terminated on contacts

Figure 105: Whole-body force controller training curves trained for a gripper force along the x- and y-axes only.

8 Acknowledgements

I deeply thank Gabriel Margolis (from the Improbable AI) for his continuous supervision, support, and guidance, which have been instrumental both for the success of this project and for my personal growth. He has developed the code base upon which I built for this project, thereby facilitating my work. I also thank specifically Assistant Prof. Pulkit Agrawal for his supervision and for promoting my scientific curiosity. I'm grateful to the Hasler Foundation for their financial support, which made this incredible experience possible. Finally, my appreciation also goes to Prof. Auke Ijspeert (from EPFL) for allowing this research project between EFPL and MIT and for his invaluable insights during the midterm presentation.

References

[1] M. company, "The impact and opportunities of automation in construction." https://www.mckinsey.com/capabilities/operations/our-insights/the-impact-and-opportunities-of-automation-in-construction. Accessed: 2023-08-23.

[2] M. company, "Imagining construction's digital future." https://www.mckinsey.com/capabilities/operations/our-insights/imagining-constructions-digital-future, 2016. Accessed: 2023-08-22.

[3] W. B. Brown, Samantha Harris, "Fatal injury trends in the construction industry." https://stacks.cdc.gov/view/cdc/115527, 2021. Accessed: 2023-08-23.

[4] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," 2022.

[5] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5761–5768, 2018.

[6] N.Bernstein, "The co-ordination and regulation of movements," 1966.

[7] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: Learning a unified policy for manipulation and locomotion," 2022.

[8] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, jan 2019.

[9] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," *CoRR*, vol. abs/2111.03043, 2021.

[10] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand dexterous manipulation from depth," 2022.

[11] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[12] Y. Ji, G. B. Margolis, and P. Agrawal, "Dribblebot: Dynamic legged manipulation in the wild," 2023.

[13] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators," 2023.

[14] M. Murooka, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "Whole-body pushing manipulation with contact posture planning of large and heavy object for humanoid robot," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5682–5689, IEEE, 2015.

[15] C. D. Bellicoso, K. Krämer, M. Stäuble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter, "Alma - articulated locomotion and manipulation for a torque-controllable robot," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8477–8483, 2019.

[16] B. Ur rehman, M. Focchi, J. Lee, H. Dallali, D. Caldwell, and C. Semini, "Towards a multi-legged mobile manipulator," 05 2016.

[17] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4630–4637, apr 2022.

[18] P. Hacksel and S. Salcudean, "Estimation of environment forces and rigid-body velocities using observers," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 931–936 vol.2, 1994.

[19] S. Kruzic, J. Music, R. Kamnik, and V. Papić, "End-effector force and joint torque estimation of a 7-dof robotic manipulator using deep learning," *Electronics*, vol. 10, 11 2021.

[20] S. Lee, S. Jeon, and J. Hwangbo, "Learning legged mobile manipulation using reinforcement learning," in *Robot Intelligence Technology and Applications 7* (J. Jo, H.-L. Choi, M. Helbig, H. Oh, J. Hwangbo, C.-H. Lee, and B. Stantic, eds.), (Cham), pp. 310–317, Springer International Publishing, 2023.

[21] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," 2023.

[22] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo, "Learning whole-body manipulation for quadrupedal robot," 08 2023.

[23] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 418–432, 1981.

[24] M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," 1981.

[25] T. Yoshikawa, "Dynamic hybrid position/force control of robot manipulators–description of hand constraints and calculation of joint driving force," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 386–392, 1987.

[26] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American control conference*, pp. 304–313, IEEE, 1984.

[27] S. Chiaverini, B. Siciliano, and L. Villani, "A survey of robot interaction control schemes with experimental comparison," *IEEE/ASME Transactions on mechatronics*, vol. 4, no. 3, pp. 273–285, 1999.

[28] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[29] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[30] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[31] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 295–302, IEEE, 2014.

[32] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 505–518, 2005.

[33] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[34] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[35] M. P. Polverini, A. Laurenzi, E. M. Hoffman, F. Ruscelli, and N. G. Tsagarakis, "Multi-contact heavy object pushing with a centaur-type humanoid robot: Planning and control for a real demonstrator," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 859–866, 2020.

[36] M. P. Murphy, B. Stephens, Y. Abe, and A. A. Rizzi, "High degree-of-freedom dynamic manipulation," in *Unmanned Systems Technology XIV*, vol. 8387, pp. 339–348, SPIE, 2012.

[37] B. U. Rehman, M. Focchi, J. Lee, H. Dallali, D. G. Caldwell, and C. Semini, "Towards a multi-legged mobile manipulator," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3618–3624, IEEE, 2016.

[38] C. D. Bellicoso, K. Krämer, M. Stäuble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter, "Alma-articulated locomotion and manipulation for a torque-controllable robot," in *2019 International conference on robotics and automation (ICRA)*, pp. 8477–8483, IEEE, 2019.

[39] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Versatile multicontact planning and control for legged loco-manipulation," *Science Robotics*, vol. 8, no. 81, p. eadg5014, 2023.

[40] U. Robotics, "B1 software and user manual," 2022.

[41] U. Robotics, "Z1 datasheet and operation manual," 2022.

[42] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[43] H. Durrant-Whyte, N. Roy, and P. Abbeel, *Learning to Control a Low-Cost Manipulator Using Data-Efficient Reinforcement Learning*, pp. 57–64. 2012.

[44] M. Deisenroth and C. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search.," pp. 465–472, 01 2011.

[45] A. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent Robotic Systems*, vol. 86, pp. 153–, 05 2017.

[46] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," 2020.

[47] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," 2018.

[48] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[49] P. Wenzel, T. Schön, L. Leal-Taixé, and D. Cremers, "Vision-based mobile robotics obstacle avoidance with deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14360–14366, 2021.

[50] S. Kataoka, S. K. S. Ghasemipour, D. Freeman, and I. Mordatch, "Bi-manual manipulation and attachment via sim-to-real reinforcement learning," 2022.

[51] P. A. Gabriel Margolis, "Learning physically grounded robot vision with active sensing motor policies," 2023.

[52] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.

[53] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," 2017.

[54] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," 2018.

[55] Y. Kim, H. Oh, J. Lee, J. Choi, G. Ji, M. Jung, D. Youm, and J. Hwangbo, "Not only rewards but also constraints: Applications on legged robot locomotion," 2023.