IEEE Open Journal of the Industrial Electronics Society

Received 7 April 2025; accepted 25 April 2025. Date of publication 5 May 2025;
date of current version 23 May 2025. The review of this article was arranged by Associate Editor Weiliang Xu.

Digital Object Identifier 10.1109/OJIES.2025.3567112

Otto—Design and Control of an 8-DoF SEA-Driven Quadrupedal Robot

ANTONELLO SCALDAFERRI, SIMONE TOLOMEI, FRANCESCO IOTTI (Student Member, IEEE), PAOLO GAMBINO, MICHELE PIERALLINI (Member, IEEE), FRANCO ANGELINI (Member, IEEE), AND MANOLO GARABINI (Member, IEEE)

Centro di Ricerca "Enrico Piaggio" and Dipartimento di Ingegneria dell'Informazione, Università di Pisa, 56126 Pisa, Italy

CORRESPONDING AUTHOR: ANTONELLO SCALDAFERRI (e-mail: antonello.scaldaferri@phd.unipi.it).

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme under Grant 101016970 (Natural Intelligence), in part by Horizon Research and Innovation Programme under Grant 101070596 (euROBIN), in part by the European Union's under Grant HORIZON-MSCA-2023-SE-01-01 - MSCA Staff Exchanges 2023 Program under Grant 101182891(NEUTRAWEED), in part by the European Union by the Next Generation EU project ECS00000017 'Ecosistema dell'Innovazione' Tuscany Health Ecosystem (THE, PNRR, Spoke 9: Robotics and Automation for Health), in part by the Italian Ministry of University and Research (MUR) as part of the PON 2014–2020 "Research and Innovation" resources—Green Action—DM MUR 1061/2021, in part by the Italian Ministry of Education and Research (MUR) in the framework of the "FoReLab" (Future-oriented Research Lab) Project (Departments of Excellence), and in part by the ARCADIT project, funded by the National Center of Research on High Performance Computing, Big Data and Quantum Computing, through an Innovation Grant assigned to UNIPISA and Italferr (CUP: I53C22000690001).

(Antonello Scaldaferri and Simone Tolomei contributed equally to this work.)

This article has supplementary downloadable material available at https://doi.org/10.1109/OJIES.2025.3567112, provided by the authors.

ABSTRACT This article presents the mechanical design of Otto, a lightweight 8-degrees-of-freedom (8-DoF) quadrupedal robot employing series elastic actuators, and a training framework for learning locomotion control policies in simulation using reinforcement learning (RL). Otto's design differs from typical 12-DoF quadrupeds by lacking hip adduction–abduction DoF. This reduces the robot's cost and weight and increases complexity for tasks involving base rotation and angular twist following. The elastic elements at the joints improve compliance, energy efficiency, safety, and stability, increase robustness, and reduce damage to robot hardware components. Our locomotion control approach leverages RL to optimize policies in simulation, allowing stable and efficient movement despite mechanical constraints, i.e., an 8-DoF quadrupedal robot. Through extensive simulation training, leveraging highly parallel Graphics Processing Unit (GPU)-accelerated robotic simulators, we ensure the policy is well-suited for deployment in real-world scenarios, where accurate motion control is critical for performance. The trained policy is then transferred to the physical robot platform. We demonstrate its effectiveness in various tasks and real-life scenarios with varying payloads and terrains, and compare it with a state-of-the-art model-based method. The results show that Otto, equipped with our RL-based locomotion control, achieves robust performance, compensating for the reality gap and managing the reduced DoF available in Otto.

INDEX TERMS Articulated soft robots, locomotion control, quadrupedal robots, reinforcement learning (RL), sim2real transfer.

I. INTRODUCTION

Quadrupedal robots have gained significant attention in the robotics community due to their potential applications in unstructured environments, such as search and rescue [1], exploration [2], agronomy [3], environmental monitoring [4], and industrial inspection [5].

Taking inspiration from nature [6], quadruped robots are designed to have 12 degrees of freedom (12-DoF), [7], [8], [9], [10], with each leg having three actuated joints: hip adduction–abduction (HAA), hip flexion–extension (HFE), and knee flexion–extension (KFE). This configuration allows for a wide range of motions, including dynamic gaits, agile maneuvers, and precise control of the robot base orientation. The adoption of elastic elements into these joints can further enhance the robot's performance [11]. For example, they enable highly dynamic motions, including higher and longer jumps [12], which are essential for successfully dealing with hostile environments.

To enable the robot to accomplish movements in unstructured and harsh environments, not only do the design aspects need to be considered, but also the development of robust planning and control strategies [1], [13], [14] should be taken into account. While the adoption of 12 actuators is an engineering-safe solution, the shoulder actuators are often underutilized in most motions. So, what if we try to use only eight (elastic) actuators to drastically reduce the robot's weight?

Reducing weight leads to several advantages. First, it extends battery life by reducing electrical power consumption [15], which enables long-lasting operations such as monitoring [16] and search and rescue [1]. Second, it enhances the robot's dynamic performances, requiring less torque to move in unstructured terrains, thus making it more efficient and agile [17]. This torque reduction not only cuts power usage but also allows the implementation of lower control gains, improving compliance, and consequently, enabling the robot to better absorb impacts [18]. Moreover, a lighter robot is often easier to deploy and transport, simplifying field operations or in scenarios where portability is fundamental, e.g., disaster response or space exploration [19].

Several methods exist for weight reduction, including the use of lighter materials [20], removing unnecessary structural components [21], or reducing the degrees of actuation, for instance, eliminating certain actuators. The HAA joints are underutilized in many tasks [22]. Removing this type of actuator reduces the robot's weight. Especially in small-sized quadruped robots, reducing weight can significantly improve the overall performance [23].

However, the planning and control approach has to compensate for this design choice, particularly during turning maneuvers, to generate robust and dynamic motions [24], [25].

Recently, reinforcement learning (RL)-based approaches have emerged as a powerful tool for locomotion control; see, e.g., [26] and[27]. RL controllers optimize locomotion control policies for agile, dynamic, and robust robots' behaviors in unseen and harsh terrains [1], [14], [25]. However, most of this literature deals with full 12-DoF rigid quadrupedal robots.

Overall, this article answers the research question: *Can an RL-based locomotion control policy effectively handle the challenges posed by an 8-DoF quadrupedal robot?*

The main contributions of this work are as follows: 1) the design of the Otto robot and 2) a state-of-the-art model-free policy based on an RL paradigm to solve locomotion tasks.

1) Otto is a lightweight 8-DoFs quadrupedal robot that features a simplified leg design with 2-DoFs actuators, omitting the HAA joints. The design of the robot draws inspiration from the natural leg structure of mammalian animals, with its legs constructed to replicate their biomechanics for efficient and agile locomotion [28]. For this, we equipped the Otto robot with series elastic actuators (SEA) [11], which enable compliant motions, improving safety, energy efficiency, and robustness in shock absorption.
2) We develop and implement a state-of-the-art RL-based locomotion control policy to enable robust motion across diverse terrains, e.g., asphalt streets, grass, etc. Our approach leverages neural-network-driven policies trained in simulation [1], [14], [26], [27] and successfully transfers them to the real robot. We also compare the policy with [29] state-of-the-art model-based method. For the first time, this approach has successfully solved such tasks dealing with an 8-DoF robot, i.e., Otto. Furthermore, in contrast to the state-of-the-art, our approach does not use estimated velocities in the observation space, which are often included [1], [14].

The rest of this article is organized as follows. In Section II, we present the state-of-the-art of quadrupedal robot design and locomotion control. In Section III, we present the mechanical and electrical design of the 8-DoF quadrupedal robot—Otto—together with the description of the software interface. In Section IV, we present the locomotion control approach for Otto and the simulation performance of the trained policy. In Section V, we demonstrate real-world deployment of the policy, argue Sim2Real transfer, and discuss the obtained results. Finally, Section VI concludes this article.

II. RELATED WORKS

The problems of design and locomotion control in quadrupedal robots are deeply connected and present several key challenges, such as maintaining stability, achieving smooth transitions between gaits, and handling external disturbances. Design and locomotion control have been extensively studied for quadrupedal robots with 12-DoF, leading to significant advancements in intelligent machines and model-based and learning-based control methods.

A. QUADRUPEDAL ROBOT DESIGN

Quadruped robots mimic animal behavior, using bioinspired designs to navigate complex terrains [30], [31]. The structural design of these robots is crucial to their performance as it influences stability, adaptability, and durability across various surfaces. The leg topology structures classify quadruped robots into three bioinspired types: insectlike, reptilelike, and mammallike structures [28]. Typically, quadruped robots have 12-DoF with three in each leg, namely: HAA, HFE, and KFE.

In recent years, soft articulated robots are becoming more diffuse [32]. These robots integrate SEAs as joints [33], which include a carefully designed compliance element, such as a spring, in series between the motor and the load, which enhances performance by reducing impact forces and introducing flexibility [34]. This compliance also improves safety and allows better navigation on uneven surfaces. As the name suggests, the spring is in series with the motor; when instead the spring is in parallel with the motor, there are the so-called parallel elastic actuators [35], [36].

Another crucial part of the stability of the robot is the design of the foot. Generally, feet can be cylindrical or spherical, with spherical feet being the most common. This shape allows the

foot to make contact with the ground from all directions [37]. However, in certain scenarios, such as when the robot encounters obstacles, slopes, or uneven terrain, traditional foot designs may struggle during locomotion tasks. In [38], they developed an articulated adaptive foot to adjust to various surfaces and improve the locomotion of the robot on challenging terrains. Whereas, Ranjan et al. [39] introduce a novel bioinspired adaptive foot modeled on the anatomy of the goat hoof. This design enables the foot to adapt dynamically across various terrains to achieve a stable configuration.

B. CONTROL OF QUADRUPEDAL ROBOTS

Literature shows two main approaches for locomotion control of quadrupedal robots: model-based [29] and learning-based [14].

1) MODEL-BASED CONTROL

The earliest solutions to quadrupedal locomotion control were rooted in model-based methods, where the dynamics of the robot was explicitly modeled using mathematical formulations of the robot's kinematics and dynamics [29]. One of the pioneering works in this area was Raibert's dynamic balance control in legged systems [40], which introduced the concept of controlling the center of mass and foot placements to maintain stability during dynamic gaits. Techniques such as inverse dynamics control, model predictive control (MPC) [41], [42], and optimization-based approaches [13], [43], [44], [45] are used to design controllers that could compute the necessary joint torques to achieve stable locomotion. These methods relied on detailed robot models, accurate information about the environment, and the robot's state.

Despite their success in enabling early quadrupedal robots to perform stable walking and running, model-based approaches faced limitations in highly dynamic or unpredictable environments, where modeling the entire system with high fidelity became infeasible. Furthermore, these methods were computationally expensive and difficult to generalize to different terrains or unexpected perturbations, such as sudden changes in ground conditions.

2) LEARNING-BASED CONTROL

In recent years, learning-based approaches, particularly RL, have gained attention in the field of quadrupedal locomotion control; see, e.g., [1], [14], [17], [25], and [26]. RL has the advantage of learning policies directly from interaction with the environment, enabling robots to autonomously discover complex behaviors without relying on predefined models of their dynamics [27]. This approach has been particularly effective in dealing with high-dimensional systems such as 12-DoF quadrupeds, where learning-based methods can adapt to diverse terrains and dynamic conditions in a way that model-based methods struggle to achieve.

One of the most notable advancements in this area was the work by Hwangbo et al. [26], where a model-free RL framework was used to train a neural-network-based control policy for a 12-DoF quadrupedal robot in simulation. The trained policy was then successfully transferred to a real robot. This work set the foundation for using RL to address dynamic locomotion challenges, bypassing the need for precise modeling of the robot dynamics and environment.

Further works, see, e.g., [1], [14], [17], [27], and [46], demonstrated the potential of RL in learning versatile locomotion behaviors, such as high-speed running and jumping, using 12-DoF robots. These advancements have paved the way for the widespread adoption of RL in quadrupedal robot control.

Subsequent studies expanded on these techniques, combining model-based and learning-based methods to improve quadrupedal locomotion stability and efficiency. For example, hybrid approaches that leverage MPC for high-level trajectory planning and RL for low-level control have been explored to enhance the robustness of gait transitions and improve energy efficiency [25].

In contrast to the substantial body of work on RL-based locomotion control of 12-DoF quadrupeds, relatively little research has been conducted on quadrupedal robots with reduced DoF, such as 8-DoF systems [47], [48], [49]. The absence of HAA DoF in these robots significantly complicates the control of dynamic locomotion, particularly for maneuvers that require rotation around the robot base or following a complex base twist trajectory. The lack of independent lateral movement in each leg limits the robot's ability to stabilize itself during sharp turns or uneven terrains, making it more challenging to design control strategies.

To the best of our knowledge, there are no comprehensive studies with extensive real-world testing in different rough environments that focus on dynamic locomotion control for 8-DoF quadrupedal robots using either model-based or learning-based approaches. Most existing research assumes the availability of all three DoFs per leg, thus neglecting the unique challenges these systems pose. This gap in the literature underscores the novelty of our work, as it is one of the first to address the dynamic locomotion control of an 8-DoF quadrupedal robot using RL in the real world.

By building on the success of RL in 12-DoF actuated quadrupedal systems, our research aims to demonstrate that learning-based approaches can overcome the limitations imposed by reduced DoF. This work contributes to the state of the art of legged systems by presenting the design of an 8-DoF quadruped and exploring control strategies specifically tailored for it (Fig. 1). This work offers new insights into how simplified and lightweight hardware can still achieve complex locomotion tasks when paired with advanced control policies.

III. ROBOT DESIGN

This section describes the design of our 8-DoF quadrupedal robot Otto, starting with the robot's mechanical design, then the electrical scheme, and finally, the low-level hardware interface to control the whole system.

A. MECHANICAL DESIGN

The Otto robot, shown in Fig. 2, is a quadrupedal mammallike robot with 8-DoF compared with canonical quadruped robots;

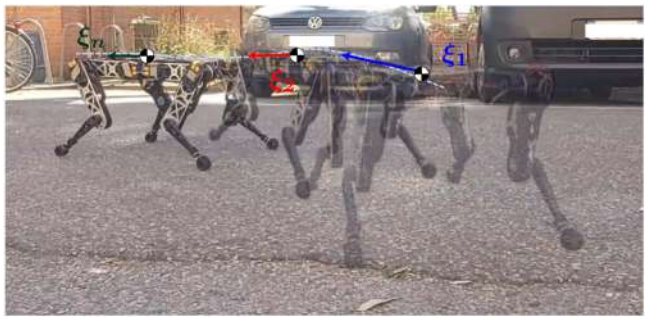


FIGURE 1. Otto walking on an asphalt terrain via an RL-based controller. The user sends command velocities, i.e., ξ_i for $i = 1, \ldots, n$, to the base and Otto follows them. Otto's figures decrease in transparency toward the final position, while the cyan dotted plot represents the center of mass trajectory.

it lacks the HAA joint, and each leg presents just two revolute joints representing the HFE and the KFE joints. Removing the HAA actuator decreases the number of actuated DoF and may potentially lead to unstable robot motion, especially in harsh terrains. However, recalling that each actuator weighs 0.5 kg, Otto would weigh more than 8 kg using 12 Elastic Modular-Actuators (EM-Act). This corresponds to a 30% increase of the current weight, i.e., 5.5 kg.

Two EM-Act actuators [11], shown in Fig. 3 without their cover plastic, are connected to create a single leg, thanks to their modular design that allows for easy coupling and the formation of a continuous chain of actuators. The actuator features a two-stage reduction ratio, with each stage offering a reduction of 3, resulting in an overall reduction ratio of 9. These elastic actuators feature a specially designed elastic component, a torsional spring, which decreases the overall stiffness of the system, enhancing compliance and reducing impact forces. The feet of the robot present a cylindrical form. To increase the contact area, they are equipped with pads made of rubber, providing improved grip and shock absorption. After that, the four legs are attached to the robot's body, which is made of aluminum and has the form of a cross-linked box containing all the electronic devices to aliment and control the system. In particular, as shown in Fig. 4, the first EM-Act actuator of each leg is coupled with the lateral layers of the body. Based on the mechanical aspects of the robot, a Unified Robot Description Format (URDF) model[1] is implemented in the Gazebo simulator as shown in Fig. 5.

B. ELECTRICAL DESIGN

The Otto robot has eight motors and uses a Raspberry Pi 4 as its onboard computer. Each actuator is controlled by a Moteus R4.11 controller, which utilizes a magnetic encoder to set the motor position and regulate current, ensuring precise control. It is powered by a DJI 47D battery with a capacity of 4500 mAh. Each leg has a dedicated MJBots power distributor R4.5B to ensure safe electrical power distribution; in addition, four individual switches are incorporated, allowing each leg to be turned ON or OFF independently. This feature proves particularly useful for identifying and isolating hardware issues. As depicted in Fig. 4, each power distributor is connected to the lateral layers of the body to minimize the distance from each leg. Directly onto the Raspberry Pi 4 is mounted the mjbots Pi3hat r4.5 module, which enables simultaneous control of each actuator drive. This module facilitates seamless communication over the Flexible Data-rate Controller Area Network (FD-CAN) bus, ensuring rapid and reliable data exchange with sensors for efficient data collection and real-time feedback.

C. CONTROL INTERFACE

A dedicated hardware interface is implemented to facilitate robot control. This interface includes multiple ROS2 packages that support modular communication between the robot joints and a laptop, utilizing either an Ethernet cable or a Wi-Fi module. Specifically, the ROS2 hardware interface enables joint control through FD-CAN using a Raspberry Pi 4 and Pi3Hat. Through a flexible configuration file, it is possible to fine-tune various motor parameters, including position, velocity, effort, current limits, and motor gains. The interface also allows for adjustable parameters to control the update rate of the system, ensuring optimal performance for different applications. In Table 1, the mechanical, electrical, and control features of the Otto robot are reported.

To allow seamless teleoperation of the robot, we developed a web-based graphical interface to send reference velocities to the locomotion controller. This interface does not require any specialized software to run, allowing us to easily and reliably control the robot from a wide range of devices. Ultimately, the whole software stack that we developed offers a highly adaptable solution, allowing for seamless integration of motors and sensors with customizable settings tailored to meet the specific needs of various robotic applications.

D. COMPARISON WITH EXISTING ROBOTIC PLATFORMS

TABLE 1. Datasheet of the Otto Robot

Otto Datasheet	
Property	**Value**
Nominal length	430 mm
Width	440 mm
Maximum height	450 mm
Minimum height	73 mm
Weight	5.5 kg
Number of DOF	8
Peak leg joint torque	19.6 Nm
Battery capacity	4500 mAh
Battery Voltage	24 V
Battery Life	1.5 h
Robot communication	Ethernet, Wi-Fi 5 GHz
Software	ROS2 compatible
Low-level motor controller	PID antiwindup
Low-level controller frequency	30 kHz

[1] [Online]. Available: https://github.com/CentroEPiaggio/mulinex_description

FIGURE 2. (a) CAD design of the Otto robot. (b) Otto robot.

FIGURE 3. EM-Act actuator without plastic skin.

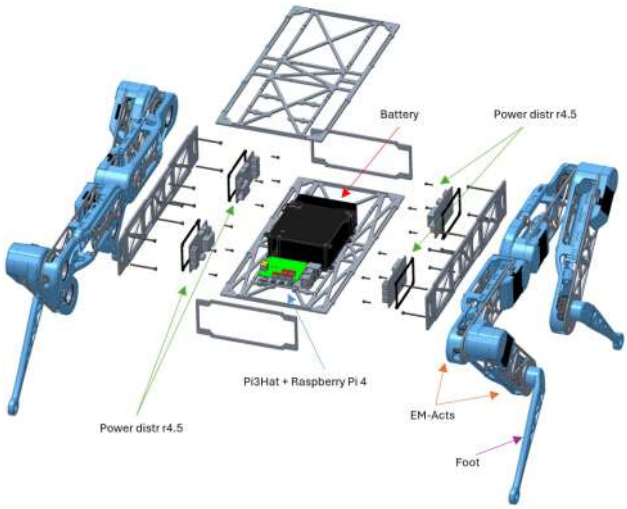


FIGURE 4. Exploded view of the Otto robot.

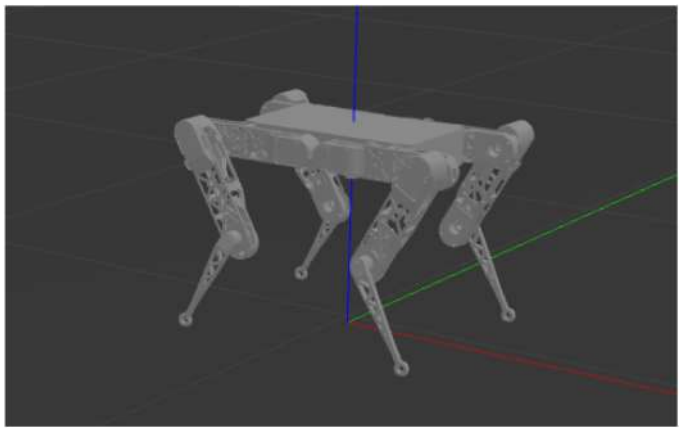


FIGURE 5. Simulated model of the Otto robot.

IV. LOCOMOTION CONTROL

In this section, we propose the RL-based locomotion controller for Otto. We start introducing background notation. Then, we describe the training environment and the control architecture. Next, we focus on the Sim2Real transfer problem, and finally, we report the training results.

A. BACKGROUND

The locomotion control problem of an 8-DoF quadrupedal robot has been modeled as a partially observable Markov decision process (POMDP), which is a generalization of a Markov decision process (MDP), a mathematical framework for formulating discrete-time decision-making problems. A POMDP models an agent decision process, in which the system dynamics are assumed to be determined by an MDP, but the agent cannot directly observe and measure the real environment state. Unlike the policy function in MDP, which maps the full environment states to the actions, a POMDP policy is a mapping from partial observations of the real environment state to the actions.

A POMDP is described as a tuple of seven elements $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \Omega, \mathcal{O}, \rho_0 \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}(s_t, a_t, s_{t+1}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the per-step reward function, $\mathcal{T}(s_{t+1}|s_t, a_t)$ is the conditional transition probability density function, Ω is the observation space, $\mathcal{O}(o_t|s_t, a_{t-1})$ is the conditional observation probability density function, and ρ_0 is the starting state distribution.

At time step t, the environment is in the state $s_t \in \mathcal{S}$. The learning agent receives an observation $o_t \in \Omega$, which depends on the current full environment state s_t and on the previously taken action a_{t-1}, with probability $\mathcal{O}(o_t|s_t, a_{t-1})$, and takes an action $a_t \in \mathcal{A}$ drawn from a parameterized policy $a_t \sim \pi_\theta(\cdot|o_t)$, with parameters θ. This causes the environment to transition to state $s_{t+1} \in \mathcal{S}$, with probability $\mathcal{T}(s_{t+1}|s_t, a_t)$.

At the same time, the agent also receives a reward signal $r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1})$, and then, the process repeats.

These reward signals are used to compute the episode return $\mathrm{R}(\tau) = \sum_{t=1}^{\infty} \gamma^t r_t$, i.e., the cumulative discounted reward, where $\tau = (s_0, a_0, s_1, a_1, \ldots)$ are trajectories (sequences of states and actions), also called episodes, drawn from the policy and the environment. The discount factor $\gamma \in (0, 1)$ determines how much immediate rewards are favored over more distant rewards. When $\gamma \to 1$, the agent cares about maximizing the expected sum of future rewards.

The agent's goal is to choose actions that maximize the expected return $\mathcal{J}(\pi_\theta) = \int_\tau \mathcal{T}(\tau|\pi_\theta)\mathrm{R}(\tau) = \mathbb{E}_{\tau\sim\pi_\theta}[\mathrm{R}(\tau)]$.

The central optimization problem to solve, for policy optimization algorithms, in RL is to find the optimal set of policy parameters θ^*, which maximizes the policy performance (the expected return), i.e. $\theta^* = \arg\max_\theta \mathcal{J}(\pi_\theta)$.

The policy $\pi_\theta(\cdot|o_t)$ is modeled as a Gaussian distribution, i.e., $a_t \sim \mathcal{N}(\mu_\theta(o_t), \sigma_\theta)$, using a multilayer perceptron (MLP) for the mean value $\mu_\theta(o_t)$ and a state-independent standard deviation σ_θ. We use the RL-Games [50] implementation of the model-free proximal policy optimization (PPO) algorithm [51] to optimize the policy parameters.

PPO is an on-policy RL algorithm that trains an agent's decision-making function to fulfill complex tasks. It belongs to the *policy gradient* methods family, which applies the policy parameters update rule $\theta_{k+1} = \theta_k + \alpha\nabla_\theta\mathcal{J}(\pi_\theta)$, where α is the learning rate and $\nabla_\theta\mathcal{J}(\pi_\theta)$ is the policy gradient.

In particular, PPO maximizes the *surrogate advantage* objective function $\mathcal{L}(\theta_k, \theta)$, instead of directly maximizing the policy performance, using stochastic gradient ascent $\theta_{k+1} = \theta_k + \alpha\nabla_\theta\mathcal{L}(\theta_k, \theta)$, where $\mathcal{L}(\theta_k, \theta)$ is a measure of how policy π_θ performs with respect to the old policy π_{θ_k} using data from the old policy.

We employ the PPO-Clip variant, which relies on explicit clipping in the objective function $\mathcal{L}(\theta_k, \theta)$ to remove incentives for the new policy to get far from the old policy

$$\mathcal{L}(\theta_k, \theta) = \mathbb{E}_{\substack{s,a\sim\pi_{\theta_k} \\ o\sim\mathcal{O}}}\left[\min\left(\frac{\pi_\theta(a|o)}{\pi_{\theta_k}(a|o)}A^{\pi_{\theta_k}}(s,a), \mathrm{clip}\left(\frac{\pi_\theta(a|o)}{\pi_{\theta_k}(a|o)}, 1-\epsilon, 1+\epsilon\right)A^{\pi_{\theta_k}}(s,a)\right)\right]$$

with *clip value* ϵ and *advantage function* $A^{\pi_{\theta_k}}(s, a)$. The latter describes how much better it is to take a specific action a in state s and is a function of the full environment state s.

PPO exploits an actor-critic structure. The actor is responsible for learning the policy π_θ, which takes the environment observation o_t as input and outputs the corresponding action a_t. The critic estimates the value function $V_\phi(\cdot)$, a function of the full environment state s with parameters ϕ, which evaluates how good it is to be in the environment state s_t, and it is used for advantage estimation. $V_\phi(\cdot)$ is an auxiliary function approximator that acts exclusively in the training phase and predicts an estimate of the expected cumulative reward (i.e., the return) starting from the full environment state s_t. In POMDP, for asymmetric training, the input of the two neural networks is different: the actor is given the environment observation o_t provided by robot onboard sensors, while the critic is given the full environment state s_t available in simulation for privileged learning. The critic has the goal of learning the reward function and is used to update the Actor neural network parameters.

B. TRAINING ENVIRONMENT

The environment[2] is the rough terrain locomotion environment from [52], based on the Isaac Gym framework [53], which leverages the Isaac Sim [54] realistic physics-based GPU-accelerated robotics simulator, allowing for highly parallel simulations. In particular, our environment employs 2048 simultaneously simulated Otto robots.

The goal of this task is to make the robot follow a user-provided desired robot base twist command $\xi_\mathrm{d} = [\mathrm{v}_\mathrm{d}^\mathrm{x}, \omega_\mathrm{d}^\mathrm{z}]$, consisting of a desired linear velocity $\mathrm{v}_\mathrm{d}^\mathrm{x}$ along the robot base sagittal x-axis and a desired angular velocity $\omega_\mathrm{d}^\mathrm{z}$ around the robot base longitudinal z-axis. During the training, the desired robot base twist commands ξ_d are generated randomly in the ranges of $[-1 \div 1]$ m/s and $[-1 \div 1]$ rad/s for $\mathrm{v}_\mathrm{d}^\mathrm{x}$ and $\omega_\mathrm{d}^\mathrm{z}$, respectively.

The policy has been trained using a game-inspired curriculum learning [27], [55], [56]. The environment terrain is divided into different terrain types (smooth sloping terrain, rough sloping terrain, stairs up and down, discrete obstacles, and wavy terrain) and levels of increasing difficulty. The higher the level, the greater the slope, step height, surface roughness, obstacle height, and wave height, and the shorter the wavelength. Further information on the increase in the difficulty of the various terrains is given in Table 2. Fig. 6 shows snapshots of the training phase in simulation on the different terrains.

All robots start at level zero, and if, at the end of the episode, the robot has traveled a distance greater than half, or less than a quarter, of the current terrain length, the robot moves to the next, or previous, level in the new episode, respectively. This allows for learning a more robust walking strategy that helps to succeed in the cases where it failed, and also not to forget the behaviors already learned, avoiding catastrophic forgetting [57].

Episodes may incur an early termination if the robot falls, i.e., more than one knee or the base comes in contact with the ground. In any case, the maximum episode duration is 20 s, dealing appropriately with time-out termination conditions [58] if the robot does not fall for the entire episode.

C. TRAINING AND CONTROL ARCHITECTURE

The locomotion controller, i.e., the policy mean value network μ_θ, is an MLP with three hidden layers of 512, 256, and 128 units, respectively, with interleaved exponential linear unit

[2] [Online]. Available: https://github.com/CentroEPiaggio/OmniIsaacGymEnvs/tree/Otto-training

TABLE 2. Initial Values of the Different Parameters for the Different Terrains and Their Corresponding Increment Value for Each Level

Terrain type	Parameter	Initial value	Level Increment	Final value
Smooth/rough sloping terrain	slope	0°	1°	20°
Stairs up/down	step height	1 cm	3 mm	7 cm
Discrete obstacles	obstacle maximum height	5 mm	2 mm	4.5 cm
Wavy terrain	wave amplitude - wave frequency	1 cm – 2.5 m^{-1}	1.5 mm – 0 m^{-1}	4 cm – 2.5 m^{-1}

The number of levels considered in the training is 20.

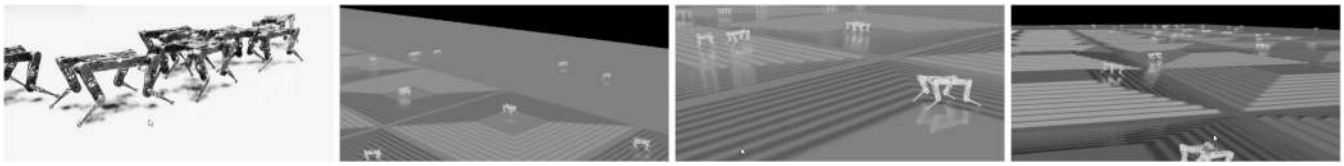


FIGURE 6. Multiple snapshots of Otto training in different terrains.

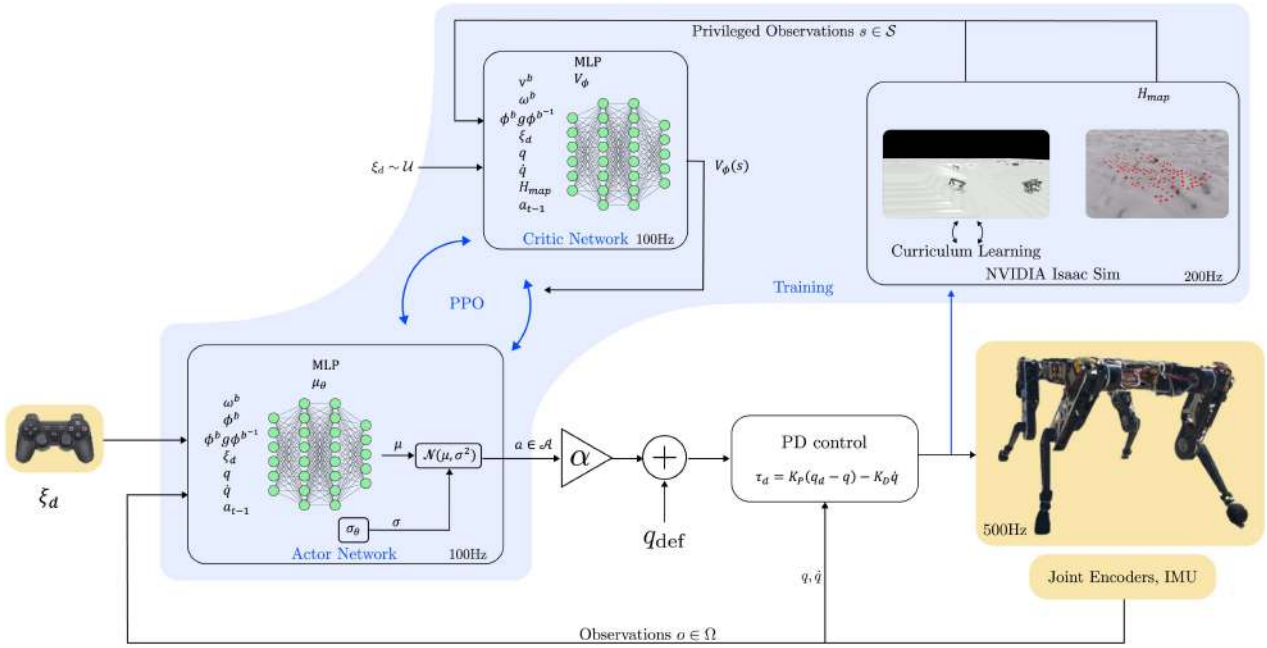


FIGURE 7. Overview of our asymmetric actor-critic training pipeline using PPO. During training, the critic (top right) has access to privileged observations (e.g., terrain heightmaps) and provides value estimates $V_\phi(s)$. The actor (bottom left) only receives nonprivileged observations such as joint states $(q, \dot{q})$, IMU readings (ω^b, ϕ^b), and user joystick commands ξ_d. The actor network outputs mean and variance parameters μ, σ that define a stochastic action distribution $\mathcal{N}(\mu, \sigma^2)$, from which actions $a \in \mathcal{A}$ are sampled at 100 Hz. These actions are then mapped through a scaling factor α and converted into torque commands via PD control at 500 Hz. After curriculum learning in simulation, only the actor-network, PD controller, and onboard sensors are deployed on the physical quadruped robot.

(ELU) activation functions. The locomotion control policy is non-perceptive, i.e., it does not take any information coming from exteroceptive sensors (e.g., LiDARs or depth cameras) as input, and it runs at 100 Hz, while, for the training, the simulation runs at 200 Hz.

The controller input is the 36-D environment observation o_t containing only the information that can be directly measured on the real robot with onboard sensors, i.e., joint encoders and IMU (see Fig. 4). Notably, in contrast to the state-of-the-art, our approach relies only on proprioceptive sensors without velocity estimates. This contrasts with the literature, where locomotion policies often include outputs from state estimators [1], [14]. As shown in Fig. 7, it contains the robot base angular velocity ω^{b} (in robot base frame), robot base orientation ϕ^{b} (in world frame as a scalar-first unit quaternion), projected gravity $\phi^{\text{b}} g \phi^{\text{b}^{-1}}$ (the gravity vector $g = [0, 0, -1]$ projected onto the robot base frame axes), desired robot base twist command $\xi_{\text{d}} = [\text{v}_{\text{d}}^{\text{x}}, \omega_{\text{d}}^{\text{z}}]$ (in body frame), joint positions q and velocities $\dot{q}$, and previous action a_{t-1}.

The controller output is the 8-D action a_t that is used to compute the desired joint positions, i.e., q_{d}, as shown in Fig. 7.

The action is first scaled by a factor $\alpha = 0.5$ and the result is added to a constant default configuration q_{def} that allows the robot to stand when the action equals zero. This default configuration q_{def} is $\pm 120^\circ$ for the hip joints and $\pm 60^\circ$ for the knee joints (see Fig. 2).

The full privileged state s_t of the environment used for asymmetric training is 266-D. As shown in Fig. 7, it contains robot base linear v^{b} and angular velocity ω^{b} (in robot base frame), projected gravity $\phi^{\text{b}} g \phi^{\text{b}-1}$ (in body frame as in the observation), desired robot base twist command $\xi_{\text{d}} = [\text{v}_{\text{d}}^{\text{x}},\ \omega_{\text{d}}^{\text{z}}]$ (same as in the observation), joint positions q and velocities $\dot{q}$, previous action a_{t-1}, and the robot-centric height-map H_{map} of the surrounding environment taken on a patch of ± 1 and ± 0.5 m along the robot base frame x- and y-axes, respectively, with a resolution of 0.1 m, for a total of 231 sampled points.

The inputs of both the actor and the critic networks are normalized using a running-mean-standard-deviation approach, by first subtracting the mean values from the input components, and then, dividing the result by their standard deviations.

The use of this asymmetric actor-critic training architecture avoids the need for more complex learning paradigms such as teacher–student policy distillation, as done in other works [1], [14], to successfully learn the RL task.

D. SIM-TO-REAL

We employ different techniques to close the Sim2Real gap [59], [60], [61], [62] and successfully deploy the trained policy on the real robot.

First, we perform a simplistic identification for the values of the actuators' dynamic parameters, i.e., the stiffness K_{P} and damping K_{D} parameters of the low-level PD controllers.

To this end, we designed a set of open-loop joint position trajectories: trotting in place, rolling around the robot base frame x-axis (similar to slowly pacing in place), pitching around the robot base frame y-axis (similar to slowly bounding in place), doing push-ups, doing stand-ups, moving the base back and forth while keeping the feet still on the ground, and performing a circular trajectory with the robot base on the robot sagittal plane.

These trajectories were executed on the real robot, for different frequencies and amplitudes, gathering the actual joint positions of the physical system. The same trajectories were then executed on the simulated robot for different values of the K_{P} and K_{D} parameters, of the simulated drives, to solve the following optimization problem:

$$K_{\text{P}}^*, K_{\text{D}}^* = \arg\min_{K_{\text{P}}, K_{\text{D}}} \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left\| q_i^{\text{real}}(t) - q_i^{\text{sim}}(t) \right\|^2$$

where N is the number of trajectories, T is the total number of trajectory samples, and q^{real} and q^{sim} are the actual joint positions of the real and simulated robot, respectively.

This procedure allowed us to identify the values for the simulated drives' parameters to achieve the simulated robot behavior that most closely replicates the real robot behavior ($K_{\text{P}} = 70.9$ Nm/rad and $K_{\text{P}} = 0.4$ Nms/rad). Indeed, using on the real robot the same values used in simulation may lead to lower performance due to the unmodeled belts, gears, delays, and other phenomena.

In particular, we started with the gains used in the simulation and incrementally increased the proportional gain until the root-mean-squared-error on the joint position was within an acceptable threshold. The derivative term was then tuned to mitigate the vibrations introduced by the increased proportional gain.

We also apply domain randomization techniques [63] at different levels during training by applying random disturbances to the robots [64], [65] in the form of random pushes applied to the bases of the robots, and by randomizing some robot parameters and other physics properties of the simulation [66], as shown in Table 3, to make the policy robust against modeling mismatches between the simulated and real robot, as well as against unmodeled dynamics.

The gravity g magnitude on the world frame z-axis is randomized to simulate different possible external loads. The randomization applied to the joint torques τ is used to simulate motor faults. To deal with different terrain conditions, we randomize terrain properties, such as restitution e and friction coefficients μ_{s} and μ_{d}, as shown in Table 3.

The initial positions and orientations of the robots are randomized too, as well as the initial joint positions and velocities, to explore more different cases and make the policy robust to various scenarios. We add random noise to the observation and actions to simulate sensor noise and actuation inaccuracies, respectively.

Furthermore, considering actuators compliance and dynamics modeling simplifications, we also perform randomization for the stiffness K_{P} and damping K_{D} actuator model parameters, and joint friction F_{V} and armature J_{a} coefficients.

The randomization, together with the actuators' dynamic parameters identification discussed previously, allowed us to successfully transfer the policy to the real robot, avoiding the use of more complex approaches, such as actuator networks training; see, e.g., [1], [14], [26], and [27].

E. REWARD SHAPING

We build upon our reward function from one of the classic end-to-end policies for quadrupedal robot locomotion; see, e.g., [52]. All the reward terms are listed in Table 4, and the new terms, aimed to improve the sim2real transfer, are presented as follows.

Due to Otto's lightweight body, we reshape the reward to reduce vibration and impulsive movements of the joints by penalizing the joints' acceleration and jerk while generating smooth joint motions [67]. To this end, we considered the norm of the 8-D vector associated with all the DOFs. Using the norm allows us to balance the convergence of the different components, ensuring that no joint value is over- or underoptimized. These terms are listed in Table 4 as *action_jerk* and *action_accel*.

TABLE 3. Randomization Applied During the Training to the Physics Parameters of the Simulation and the Robots

Entity	Parameter	Description	Frequency	Operation	Distribution	Δ
Simulation	g	Gravity	On reset	Additive	Uniform	$[0 \div 2.943]$ (world frame z-axis)
Robot	K_P	Joint drives stiffness	On reset + 20 s	Scaling	Uniform	$\pm 5\%$ (on reset) $\pm 10\%$ (on interval)
	K_D	Joint drives damping	On reset + 17.5 s	Scaling	Uniform	$\pm 10\%$ (on reset) $\pm 5\%$ (on interval)
	F_V	Joint frictions	On reset + 13 s	Direct (on reset) Scaling (on interval)	Uniform	$\pm 1.5 \times 10^{-3}$ (on reset) $\pm 9\%$ (on interval)
	J_a	Joint armatures	On reset	Direct	Uniform	$\pm 2 \times 10^{-3}$
	τ	Joint torques	25 s	Scaling	Uniform	$\pm 15\%$
Terrain	μ_s	Static friction	10 s	Direct	Uniform	$[0.3 \div 1.3]$
	μ_d	Dynamic friction	10 s	Direct	Uniform	$[0.15 \div 1.3]$
	e	Restitution	10 s	Direct	Uniform	$[0.2 \div 1.0]$

Here, "on reset" means every time an episode terminates, while the "on interval" randomization (shown in seconds) refers to the simulation time. Δ is the distribution interval. The random value is applied considering the initial value for the "on reset" randomization, While, for the "on interval" randomization, it is applied considering the last value set by the "on reset" randomization.

TABLE 4. Reward Terms With Their Corresponding Weighting Factors

Reward Terms		
Component	**Description**	**Value**
lin_vel_xy	Linear velocity, XY plane	1.5
ang_vel_z	Angular velocity, Z- axis	0.75
lin_vel_z	Linear velocity, Z- axis	-4.0
ang_vel_xy	Angular velocity in the XY plane	-0.05
orient	Orientation	-0.75
torques	Torques	-0.0
joint_acc	Joint accelerations	-1.5e-3
action_rate	Rate of actions	-7.5e-3
base_height	Height of the base	-2.5
hip	Hip position	-0.25
feet_pos_z	Feet positions, Z- axis	-0.1
action_accel	Action accelerations	-0.07
action_jerk	Action jerks	-5e-4
joint_jerk	Joint jerks	-1e-4
falcata	Step's length	-10.0
action_jerk_max	Maximum action jerk	-0.012
joint_jerk_max	Maximum joint jerk	-1.8e-3
action_accel_max	Maximum action acceleration	0.1
joint_accel_max	Maximum joint acceleration	0.05
feet_air_time	Feet air time	0.25

Some of these terms have a particular evolution w.r.t. the observations; see Section IV-E.

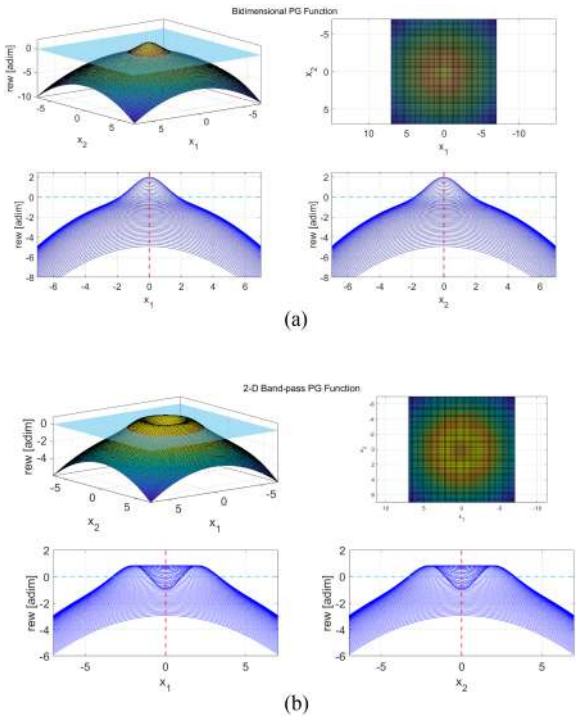


FIGURE 8. Example of function in (3). On the top left, we report the 3-D representation of the surface, on the top right, we show its upper section. In the bottom line, the surface is depicted in the function of one component.

In addition, we added maximum torque and jerk penalty terms, i.e., *joint_accel_max* and *joint_jerk_max* in Table 4, which reduce instantaneous peaks that can lead to high currents, actuator stress, and premature breakage [11].

To regularize the gait, we used two dedicated reward terms. The first one enforces the desired stride, while the second one enforces the desired feet air time, i.e., *falcata* and *feet_air_time* in Table 4, respectively.

To guarantee a gait without jolting and with the base sufficiently high, we introduced two terms regarding the orientation and the base height, i.e., *orient* and *base_height*, respectively; see Table 4. We reshape these two reward contributions as follows. We blend a quadratic penalty and a Gaussian-like gain term depending on how distant the base is from the desired pose, i.e., $x_{\text{act}} - x_{\text{des}}$. The function designed is the named PG function, from the penalty and gain terms, i.e.,

$$r(x) = \alpha_{s-} x^\top x + \alpha_{s+} \mathrm{e}^{-\frac{x^\top x}{2\sigma^2}} \tag{1}$$

where α_{s-} and α_{s+} are scale factors to tune, $x = x_{\text{act}} - x_{\text{des}}$ is the error, and σ^2 is the variance of the Gaussian-like bell. The α_{s-} penalizes for a large deviation from the desired value. The α_{s+} is the maximum value of the Gaussian-like function part, it rewards if the variations from the desired value are small. We defined the variance to blend the parabola and the bell

$$\sigma^2 = \frac{-1}{\kappa \alpha_{s-}} \tag{2}$$

with the tunable parameter $\kappa \geq 1$. If $\kappa = 1$, the bell joins the parabolic curve smoothly. Conversely, if $\kappa > 1$ the bell squeezes Fig. 8(a). This function can be compared with a low-pass filter, but there may be scenarios in which x_{act} needs to be reduced. Next, we require a bandpass filter. To this end, we add another Gaussian-like function with a negative scale to

TABLE 5. Values of the Parameters Used for the Training With the PPO Algorithm

RL Parameters	
Property	**Value**
γ	0.99
ϵ	0.2
GAE-λ	0.95
KL threshold	0.008
Learning rate	3e-4 (Adaptive)
Horizon length	48
Critic coefficient	2
Entropy coefficient	0.001

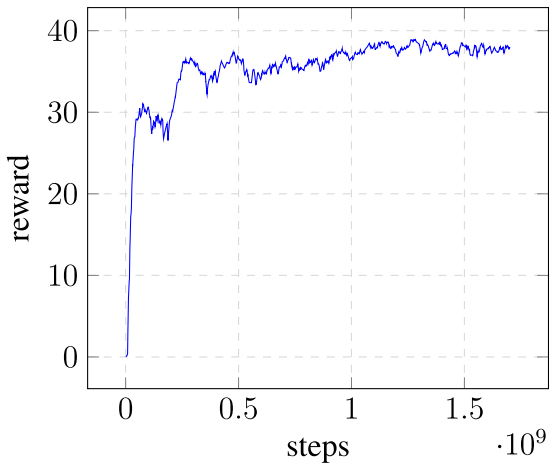


FIGURE 9. Mean cumulative reward over policy steps during the training process.

"dig" into the bonus range Fig. 8(b). Recalling (1), the overall reward follows:

$$r_{\text{bp}}(x) = r(x) + \alpha_{\text{GN}} e^{-\frac{x^\top x}{2\sigma_{\text{GN}}^2}} \tag{3}$$

with the new tunable parameter α_{GN} and σ_{GN}^2 is a new variance, i.e.,

$$\kappa_{\text{GN}} = -\kappa_{\text{sc}} \frac{\alpha_{\text{GN}}}{\alpha_{s+}}, \quad \sigma_{\text{GN}}^2 = \frac{-1}{\kappa_{\text{GN}} \alpha_{s-}} \tag{4}$$

with $\kappa_{\text{sc}} \geq 1$, $\alpha_{\text{GN}} < 0$, and $|\alpha_{\text{GN}}| > \alpha_{s+}$. Examples of the low-pass and bandpass versions are displayed in Fig. 8

F. TRAINING RESULTS

The values of the main PPO algorithm hyperparameters, used for training the policy, are shown in Table 5. The training reward trend is shown in Fig. 9, and how the curriculum learning affects the training process can be seen, i.e., the decreasing reward means the advancement to the next level.

Table 6 shows the policy performance in simulation. It was evaluated considering 2048 episodes of 20 s. In these episodes, the robots are given all different random desired twist command references, over the various terrains considered during the training phase.

TABLE 6. Summary of the Training Results

Training Results	
Property	**Value**
v_x	86%
ω_z	87%
Fallen	15%
Episode length	97%
Reward	44.4

These results show that the policy can achieve good twist command tracking performance, despite the limitations posed by this specific mechanical design, successfully crossing the different terrains considered in most cases.

V. EXPERIMENTAL RESULTS

This section reports the experiments to verify the robustness of the 8-DoF robot—Otto—and the performance of the trained-in-simulation policy. We perform three types of experiments: velocity tests, indoor obstacle tests, and outdoor tests; see Fig. 10. These tests, similar to the different terrains during the training process on the simulator, act as a real-world disturbance, introducing different forms of challenge and unpredictability. By assessing performance on these diverse surfaces and with varying obstacles, we demonstrate the system's capability to maintain control in dynamic environments.

A. VELOCITY TRACKING

The following experiments have twofold objectives. First, we want to establish the robustness of Otto's design, and second, we stress the capability of the policy to track linear velocity commands.

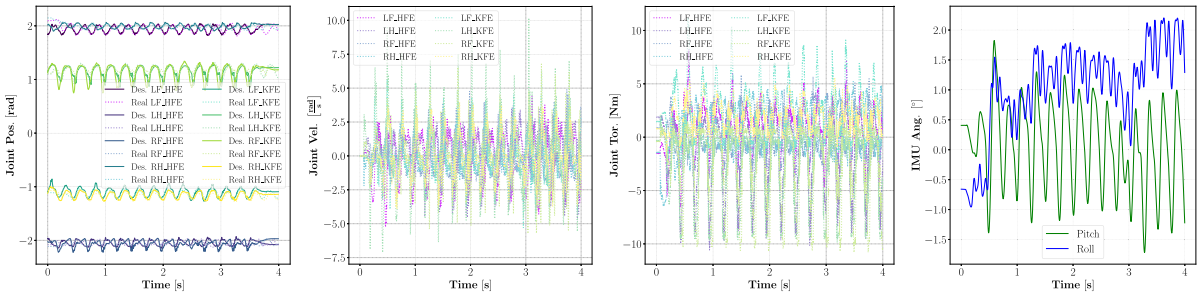
We test the linear velocity and not the angular one. Three different absolute values of velocity are chosen to test: ± 0.5, ± 0.7, and ± 1 m/s. We repeat three times each of the three velocity signals, leading to 18 experimental tests. The velocity is given as policy input as a square wave. To quantify the velocity tracking, we use the following metrics:

$$\begin{aligned} v_{x_{\text{mean}}} &\triangleq \frac{1}{n} \sum_{i=1}^{n} \frac{x_i}{\Delta_{\text{s}}} \\ \eta_x &= \frac{v_{x_{\text{mean}}}}{v_x} \\ \eta_y &= -\frac{h_{\text{s}}}{l_{\text{des}}} \\ \eta_{\text{tot}} &= \eta_x + \eta_y \end{aligned} \tag{5}$$

where x_i is the real displacement on the longitudinal axis, v_x is the command velocity, n is the number of repetitions, i.e., 3, Δ_{s} is the step duration, l_{des} is the desired displacement onto the longitudinal axis, and h_{s} is the real motion onto the lateral axis. We define η_x for the longitudinal displacement and η_y for the horizontal shift. The latter is always negative to penalize the lateral displacement. In the best case, $\eta_x = 1$ and

(a) (b) (c)

FIGURE 10. Photos of different scenarios to test Otto's locomotion abilities. (a) Indoor terrain with obstacles. (b) Outdoor asphalt terrain. (c) Outdoor grass terrain.



(a) (b) (c) (d)

FIGURE 11. Plots of the experimental results for velocity tracking testing with $v_x = 0.5$ m/s. (a) Joint positions. (b) Joint velocities. (c) Joint torques. (d) Roll and pitch.

TABLE 7. Results for the Velocity Tracking Tests

Velocity Tracking Results	
Property	**Value**
$E[e_{l_d}]$ [m]	0.56
$\bar{\sigma}(l_d)$ [m]	0.06
$E[hs]$ [m]	0.28
$\bar{\sigma}(h_s)$ [m]	0.07
η_x [%]	72.1
η_y [%]	-14.1
η_{tot} [%]	58

$\eta_y = 0$, hence $\eta_{tot} \in [-1, 1]$. After each experimental test, we measure the real displacement along the longitudinal and lateral axes.

Table 7 displays the mean results obtained: the mean of and the standard deviation of the error on the longitudinal displacement, i.e., $E[e_{l_d}]$ and $\bar{\sigma}(l_d)$, the mean and standard deviation of the lateral shift, i.e., $E[h_s]$ and $\bar{\sigma}(h_s)$, and the three return indexes, i.e., η_x, η_y, and η_{tot}.

Figs. 11–13 report the experimental results for testing $v_x = 0.5, 0.7$, and 1 m/s, respectively. We report here just the positive command for the sake of space. Each experiment lasts approximately 4 s to standardize the tests. Refer to the Video attachment for further details.

Discussion: These tests assess the robustness and durability of both the mechanical and electrical components; see Figs. 11(b)–13(b). We successfully conducted multiple tests repeatedly. Figs. 11(a), 12(a), and 13(a) present the joint positions, demonstrating that the controller achieves accurate tracking performance with a slight residual error introduced to maintain system compliance. Moreover, Figs. 11(b), 12(b), and 13(b) illustrate the joint velocities, showing transient peaks reaching approximately 15 rad/s and an average operating range of around ±5 rad/s. Since the EM-Act datasheet specifies a maximum joint velocity of 69 rad/s [11], these results confirm that the system operates well within its design limits, whereas Figs. 11(c), 12(c), and 13(c) illustrate the joint torques, which remain within the specified limit of 15 Nm. Notably, torque peaks do not exceed 10 Nm, confirming that the control policy operates without imposing excessive stress on the system. Finally, Figs. 11(d), 12(d), and 13(d) illustrate the roll and pitch angles for the three tests, with the maximum base oscillation remaining within 5°, demonstrating the effectiveness of the control policy.

These results demonstrate the robustness of the entire hardware system. Mechanically, the system performed the tasks without any noticeable structural damage, and electrically, it consistently delivered the required torque and velocity to each joint.

B. MODEL-BASED CONTROL COMPARISON

We compare our model-free learning method with a state-of-the-art solution, namely differential dynamic programming (DDP) using the CROCODDYL library presented in [29].

DDP is a model-based optimal control routine that can generate feasible motions, feedforward terms, and feedback policy; see e.g., [29], [68], and[69].

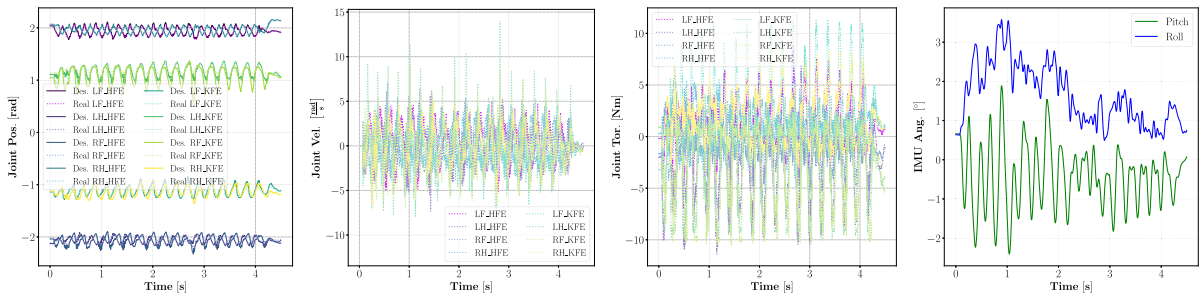


FIGURE 12. Plots of the experimental results for velocity tracking testing with $v_x = 0.7$ m/s. (a) Joint positions. (b) Joint velocities. (c) Joint torques. (d) Roll and pitch.

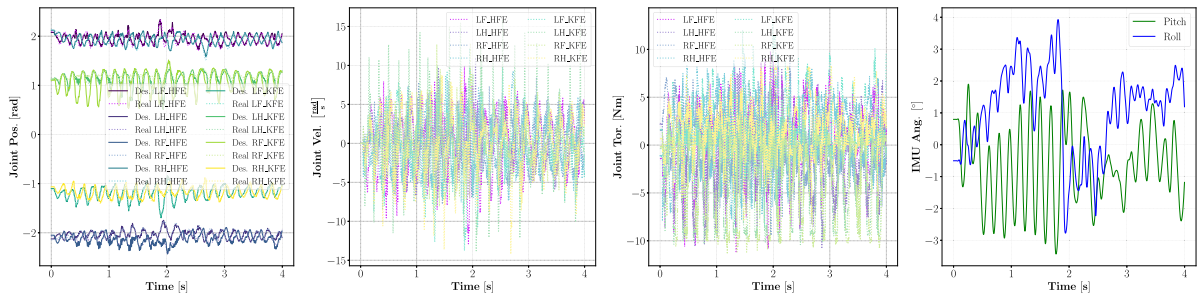


FIGURE 13. Plots of the experimental results for velocity tracking testing with $v_x = 1.0$ m/s. (a) Joint positions. (b) Joint velocities. (c) Joint torques. (d) Roll and pitch.

We perform a walking task setting the following parameters: `stepLength`: 0.1 [m], `stepHeight`: 0.05 [m], `timeStep`: 0.01 [s], `stepKnots`: 100, and `supportKnots`: 50.

In Fig. 14 are shown the results of this experiment.

Fig. 15 reports a photo sequence of the walking motion. The algorithm can execute the walking. However, DDP needs approximately 30 s to find the optimal trajectory.

Discussion: The model used for planning with DDP and training with RL is the same. While DDP produces the most elegant gait, it sacrifices the adaptability and therefore the robustness characteristic of learning-based approaches. In contrast, the walk learned through RL proves to be the most robust. Both methods show comparable performances in terms of tracking [see Fig. 14(a)], speed [see Fig. 14(b)], torque [see Fig. 14(c)], and motion.

Furthermore, DDP walking causes more tilting of the base [see Fig. 14(e)], even at relatively slow speeds and on flat terrain, compared to RL walking, as shown in the velocity tracking task. This indicates that RL walking has a generalized behavior when dealing with multiple terrains, making it more robust than DDP walking.

C. PAYLOAD

In this experiment, we test and stress the effectiveness of Otto's design and the robustness of the control policy via a payload test. We load Otto with a payload of 5 kg, which is almost its weight. The experiment lasts 20 s indoors, where the user controls the robot with a joystick web interface, sending velocity commands to the robot's base. The overall results are shown in Fig. 16. It is important to note that the same policy used in the velocity tracking experiments was employed here as well.

Furthermore, in Fig. 17, a sequence of the walking of the Otto robot with a payload of 5 kg is displayed.

Discussion: During the test, the robot successfully walked with a payload of 5 kg, showing the robustness of both design and control policy.

Fig. 16(a) presents the roll and pitch angles. During the experiment, the base of the robot exhibited significantly more tilt compared to earlier experiments, showing the robustness of the overall hardware system.

Fig. 16(b) and (c) reports the linear and angular velocity commands, respectively, provided by the user via the joystick web interface.

The temperature in Fig. 16(d) rises steadily throughout the experiments. It is worth noting that the knee joints exhibit a higher temperature due to the greater load they support.

Despite the payload, torque values remain below the 15-Nm limit, even under these demanding conditions; see Fig. 16(e). Overall, the system demonstrates accurate reference tracking while maintaining compliant behavior; see Fig. 16(f).

D. TERRAIN WITH OBSTACLES

In this experiment, the robot walks indoors on a terrain with randomly placed obstacles. These range from simple

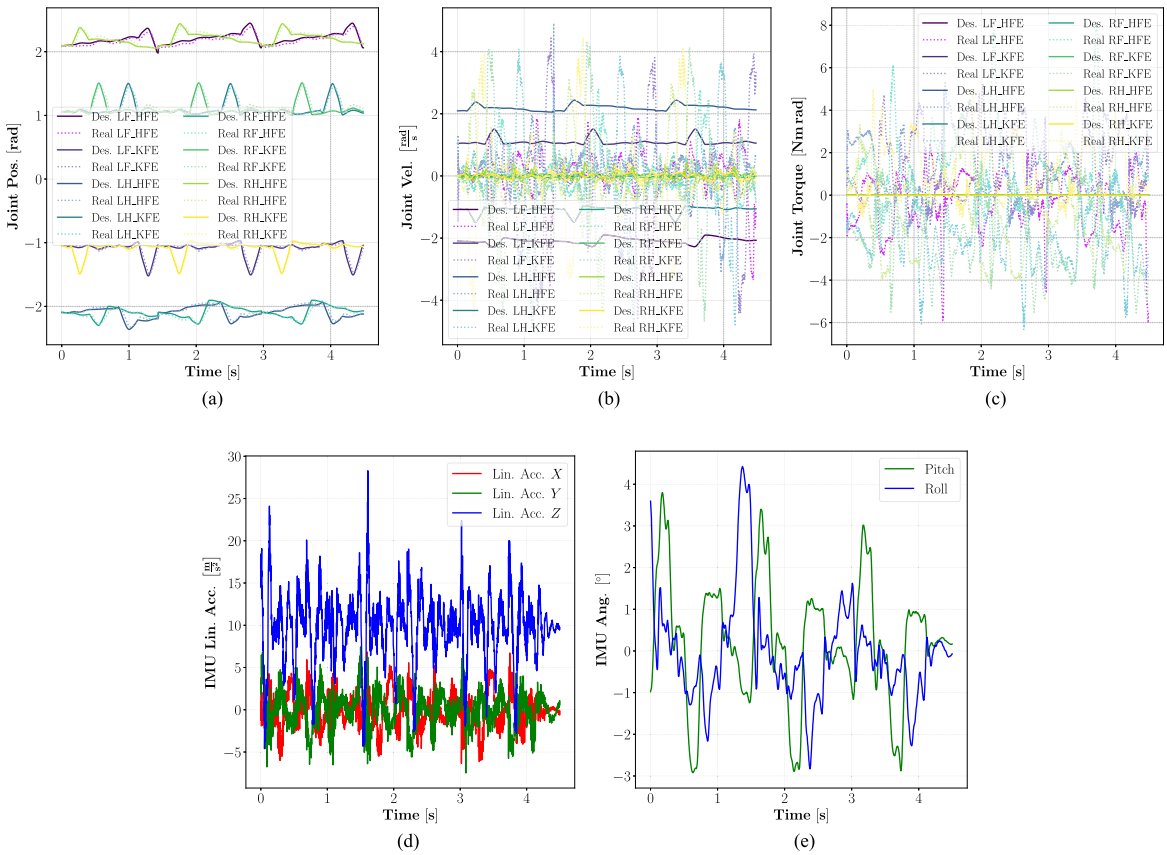


FIGURE 14. Results for the DDP walking task implemented on Otto. (a) Joint positions. (b) Joint velocities. (c) Joint torque. (d) IMU orient. (e) IMU lin. acc.

FIGURE 15. Photo sequence of Otto walking while performing two steps via DDP.

barriers on the floor, e.g., wooden bricks and disk-shaped weights, to slope surfaces of inclination between 4° and 20°, see Fig. 10(a). The experiment lasts approximately 110 s, where the user controls the robot with a joystick web interface, sending velocity commands to the robot's base. Note that different terrain friction coefficients are present in this scenario.

Fig. 18 reports the experimental results for the indoor test where the robot walks into an unstructured indoor terrain and is controlled via a joystick web interface. Fig. 19 shows a photo sequence of the robot walking into the unstructured indoor terrain with obstacles.

Discussion: The behavior of the policy shows good performance in terms of maneuverability and stability while showing a smooth gait. In detail, Fig. 18(b) and (c) reports the command provided by the user via the joystick web interface. Moreover, Fig. 18(a) presents the yaw and pitch angles, revealing that the base of the robot exhibits significantly more tilt compared to earlier experiments, demonstrating the robustness of the overall hardware system. Fig. 18(f) and (e) illustrates the joint positions and torques, respectively, confirming that torque values remain below the 15-Nm limit, even under these demanding conditions. Overall, the system demonstrates accurate reference tracking while maintaining compliant behavior. Finally, Fig. 18(d) depicts the joint temperature. The temperature rises steadily throughout the experiments. It is worth noting that the knee joints exhibit a higher temperature due to the greater load they support. In some cases, the robot partially falls from the footplate, but the policy was capable of raising the feet and making the robot rise completely above, showing robustness to disturbances of the terrain. Refer to the Video attachment for further details.

E. OUTDOOR

In the last experiments, the robot crosses common everyday life environments, i.e., asphalt streets and a garden, as

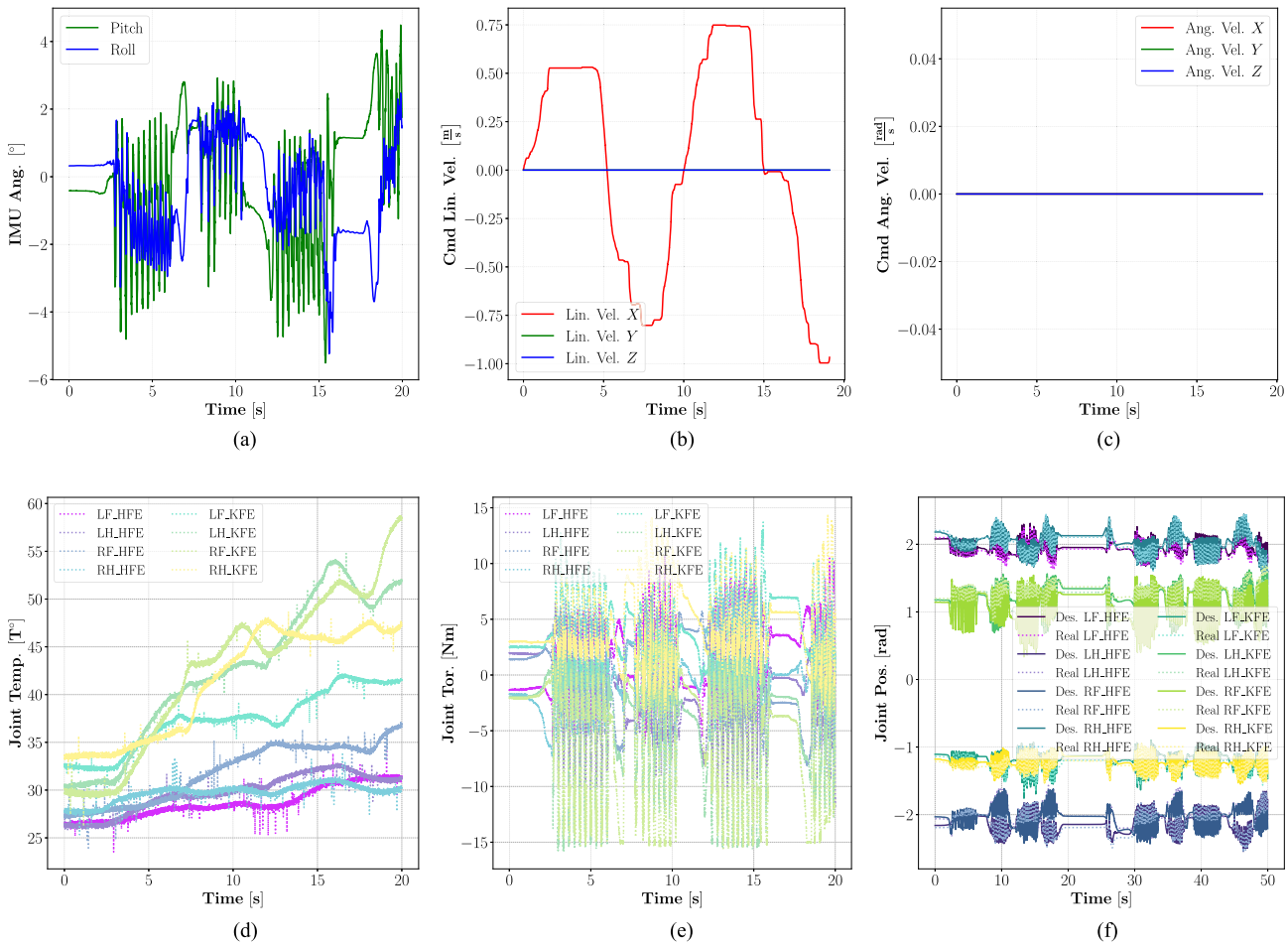


FIGURE 16. Results for the *EGR-EET3* policy dealing indoor test with a payload of 5-kg controlled via the joystick web interface. (a) Roll and pitch. (b) Command vel. lin. (c) Command vel. ang. (d) Joint temperature. (e) Joint torque. (f) Joint positions.

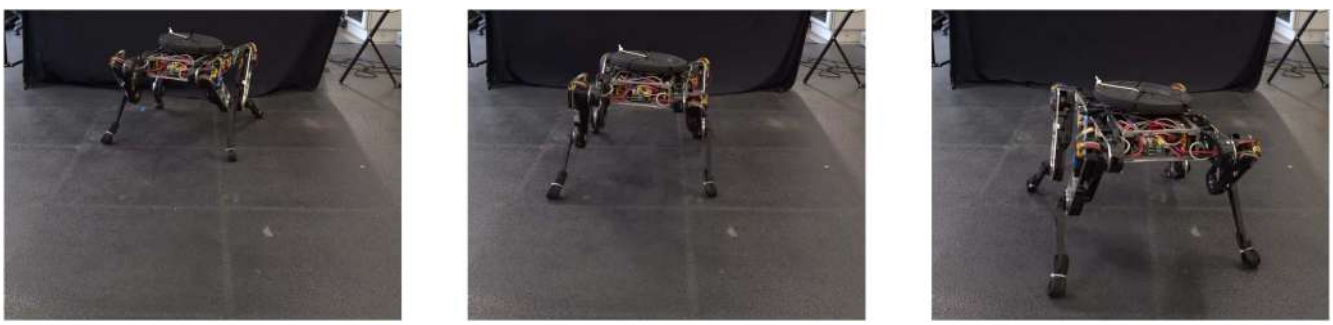


FIGURE 17. Photo sequence of the motion of the Otto robot while walking with a 5-kg payload.

in Figs. 21 and 23, respectively. The street tests the walk on an irregular, rigid ground with many inhomogeneities for 220 s. Conversely, the grass terrain is challenging due to the high ground compliance, bushes that work as obstacles with very high compliance, and the different ground friction coefficients. The robot moves in the grass terrain for about 95 s. In both cases, the user moves the robot freely using the web interface.

Figs. 20 and 22 report the experimental results on the grass scenario.

Refer to the Video attachment for further details.

Discussion: The policy maintains stable walking, leading to smooth and regular robot behavior. The robot was successfully able to work in outdoor conditions. At the same time, the trained policy showed enough robustness to allow the robot to walk, despite not being trained explicitly for terrains with high compliance such as grass(see Fig. 23), or extremely high friction, such as asphalt terrain (see Fig. 21). Figs. 20(b), and 22(b) and Figs. 20(c) and 22(c) report the command provided by the user via the joystick web interface. As

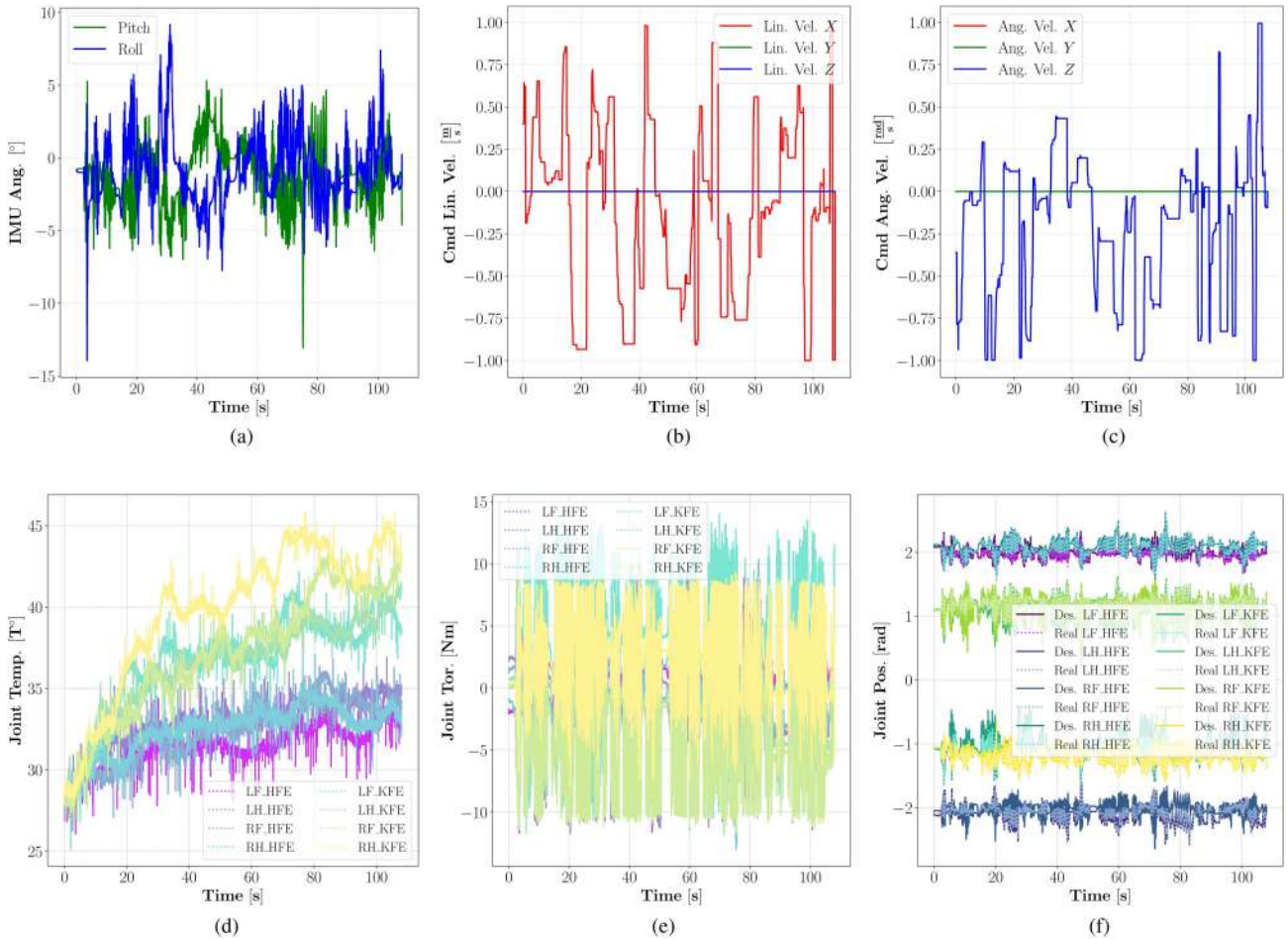


FIGURE 18. **Results for the *EGR-EET3* policy dealing with indoor obstacles terrain controlled via the joystick web interface. (a) Roll and pitch. (b) Command vel. lin. (c) Command vel. ang. (d) Joint temperature. (e) Joint torque. (f) Joint positions.**

FIGURE 19. **Photo sequence of Otto walking indoors with obstacles. Snapshots are taken within a 2-s window.**

shown in Figs. 20(d) and 22(d), the joint temperatures display varying patterns depending on the environment. Specifically, during the street experiment, the temperature rises linearly throughout the task. In contrast, the behavior observed in the grass experiment differs. The softness of the terrain reduces the torque during the experiment, resulting in less overheating of the joints. Furthermore, Figs. 20(a) and 22(a) display the roll and pitch angles, it is clear to see the difference in angles from the two plots, in asphalt the terrain is more regular respect the grass and this is depicted in the figures showing that in grass terrain the base of the robot is more tilted. Finally, Figs. 20(f) and 22(f) display the joint positions, as previously discussed, the control objective is to achieve a compliant behavior while keeping the tracking error quite low, and Figs. 20(e) and 22(e) report the joint torques for tests conducted on asphalt and grass, respectively. Notably, torque levels remain relatively high throughout the asphalt experiment, while they vary more significantly on the softer grass terrain.

It is important to note that both tests were conducted on the same day, so the weather conditions were virtually identical.

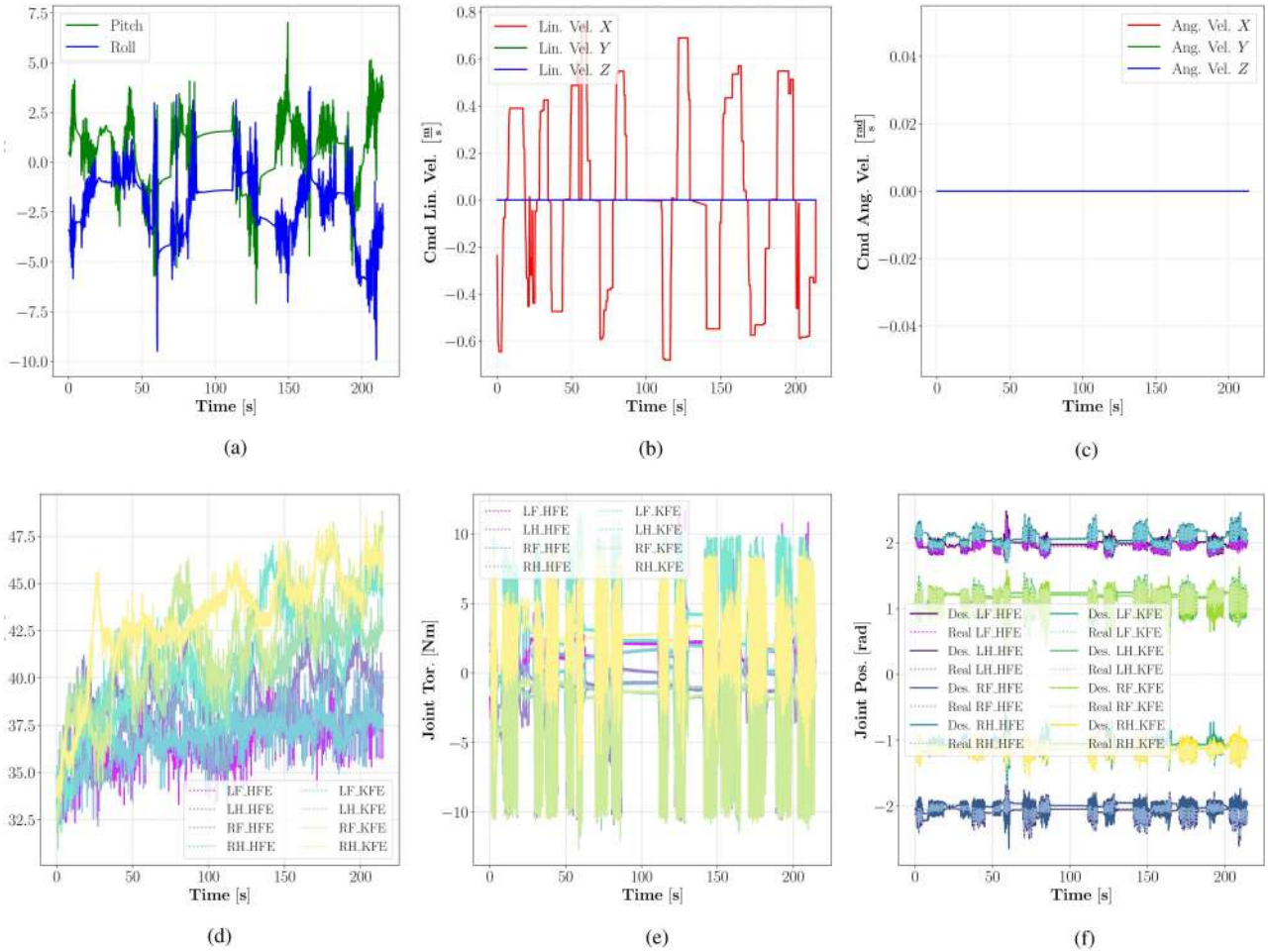


FIGURE 20. Results for the *EGR-EET3* policy dealing outdoors in asphalt terrain controlled via the joystick web interface. (a) Roll and pitch. (b) Command vel. lin. (c) Command vel. ang. (d) Joint temperature. (e) Joint torque. (f) Joint positions.

FIGURE 21. Photo sequence of Otto walking in the asphalt terrain. Snapshots are taken within a 4-s window during the task.

F. COMPARISON WITH OTHER 12-DOF QUADRUPED ROBOTS

Unlike any 12-DoF quadrupedal robots, Otto cannot perform lateral motion but can only move forward, backward, and turn right or left. However, compared to other state-of-the-art quadrupedal robots, Otto presents similar performances in terms of forward velocities, payload, etc. For example, the maximum forward velocity of Otto is 1 m/s, while the ANYmal D can move up to 1.3 m/s, and the Spot up to 1.6 m/s.

In [70], Solo12 is controlled using RL to navigate various terrains, yielding results similar to ours. For instance, the maximum forward velocity is 1.5 m/s, while the maximum angular velocity command is 1 rad/s. Thus, our findings demonstrate that even with 8 actuators, the robot can perform comparably to other 12-DoF robots.

Regarding the design aspects, Table 8 shows the difference between Unitree Go1 and Otto. The different number of DoFs is reflected in the weights of the two robots. For this reason, the battery capacity of the Otto robot is smaller, even if the two robots show a similar work period time. The payload capacity of the Otto, concerning the overall mass, is greater than that of the Unitree Go1, showing that the EM-Act actuators can effectively be adopted in the prototyping of novel robotic platforms. Thanks to their modular design, they are well-suited

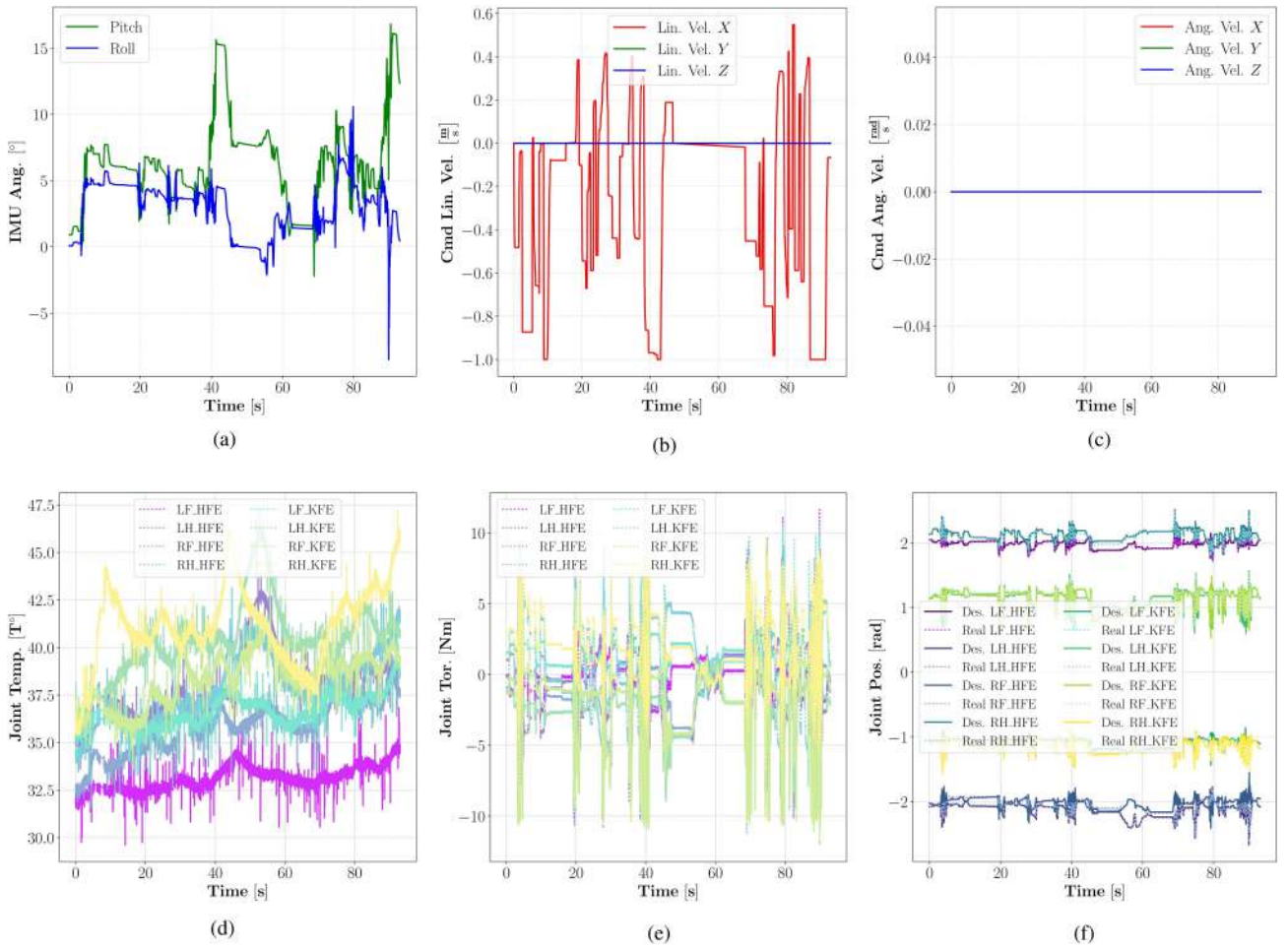


FIGURE 22. **Results for the *EGR-EET3* policy dealing outdoors in grass terrain controlled via the joystick web interface. (a) Roll and Pitch. (b) Command vel. lin. (c) Command vel. ang. (d) Joint temperature. (e) Joint torque. (f) Joint positions.**

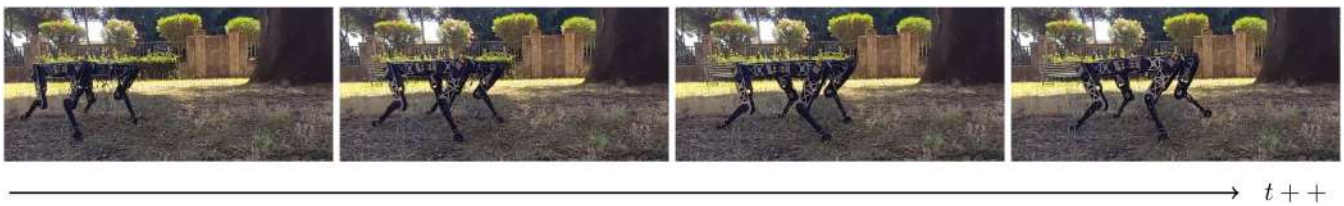


FIGURE 23. **Photo sequence of Otto walking into the grass terrain. Snapshots are taken within a 3-s window during the task.**

TABLE 8. **Comparison Between Unitree Go1 and Otto Robots**

Specification	Unitree Go1	Otto
Degrees of freedom (DoF)	12	8
Actuator type	Electric	Series Elastic
Weight (kg)	12	5.5
Payload capacity (kg)	3–5	5
Battery capacity (mAh)	6000 or 9000	4500
Operability duration (h)	1–2.5	1.5

for assembling robots with multiple DoFs. Moreover, Otto features a unique open-source ROS2 hardware interface that facilitates the easy adjustment of low-level controller parameters.

VI. CONCLUSION

In this article, we proposed the design of an SEA-equipped quadrupedal robot with 8-DoF, namely Otto, and a learning framework for training a policy for its control. The design solution shows high performance in terms of tracking joints, temperature values, and high power-to-weight ratio. On the other hand, the policy is robust and enables the robot to traverse unstructured and unknown real-life environments. Overall, 8-DoF is enough for walking gaits, showing good maneuverability and agility. To validate our framework, we

conducted a series of experiments in both indoor and outdoor environments, along with comparisons to model-based control approaches. Furthermore, we demonstrated the robustness not only of the learned policy but also of the robot's design, which was capable of walking while carrying a payload of up to 5 kg.

Future work will try to incorporate optimal control algorithms into this paradigm and equip the robot with exteroceptive sensors.

ACKNOWLEDGMENT

The authors would like to thank L. Martignetti, J. Cioni, and G. Di Lorenzo from the University of Pisa for their support and help.

REFERENCES

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

[2] S. Gilroy et al., "Autonomous navigation for quadrupedal robots with optimized jumping through constrained obstacles," in *Proc. IEEE 17th Int. Conf. Automat. Sci. Eng.*, 2021, pp. 2132–2139, doi: 10.1109/CASE49439.2021.9551524.

[3] L. Milburn, J. Gamba, M. Fernandes, and C. Semini, "Computer-vision based real time waypoint generation for autonomous vineyard navigation with quadruped robots," in *Proc. IEEE Int. Conf. Auton. Robot Syst. Competitions*, 2023, pp. 239–244, doi: 10.1109/ICARSC58346.2023.10129563.

[4] F. Angelini et al., "Robotic monitoring of habitats: The natural intelligence approach," *IEEE Access*, vol. 11, pp. 72575–72591, 2023, doi: 10.1109/ACCESS.2023.3294276.

[5] C. Gehring et al., "Anymal in the field: Solving industrial inspection of an offshore HVDC platform with a quadrupedal robot," in *Proc. Conf. Field Serv. Robot.*, Eds., 2021, pp. 247–260.

[6] Y. Sun, C. Zong, F. Pancheri, T. Chen, and T. C. Lueth, "Design of topology optimized compliant legs for bio-inspired quadruped robots," *Sci. Rep.*, vol. 13, no. 1, 2023, Art. no. 4875, doi: 10.1038/s41598-023-32106-5.

[7] M. Hutter et al., "Anymal—A highly mobile and dynamic quadrupedal robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 38–44, doi: 10.1109/IROS.2016.7758092.

[8] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proc. Vols.*, vol. 41, no. 2, pp. 10822–10825, 2008, doi: 10.3182/20080706-5-KR-1001.01833. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016407020

[9] B. Dynamics, "Spot—The agile mobile robot," [Online]. Available: https://bostondynamics.com/products/spot/. [Accessed: May 18, 2025].

[10] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HYQ—A hydraulically and electrically actuated quadruped robot," *Proc. Inst. Mech. Eng., I, J. Syst. Control Eng.*, vol. 225, no. 6, pp. 831–849, 2011, doi: 10.1177/0959651811402275.

[11] R. k. M. Gopanunni, L. Martignetti, F. Iotti, A. Ranjan, F. Angelini, and M. Garabini, "EM-ACT: A modular series elastic actuator for dynamic robots," *IEEE Open J. Ind. Electron. Soc.*, vol. 5, pp. 468–480, May 2024.

[12] J. Ding, V. Atanassov, E. Panichi, J. Kober, and C. Della Santina, "Robust quadrupedal jumping with impact-aware landing: Exploiting parallel elasticity," *IEEE Trans. Robot.*, vol. 40, pp. 3212–3231, Jun. 2024.

[13] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3359–3365, doi: 10.1109/IROS.2017.8206174.

[14] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Robot.*, vol. 5, no. 47, 2020, Art. no. eabc5986. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abc5986

[15] T. Verstraten, M. S. Hosen, M. Berecibar, and B. Vanderborght, "Selecting suitable battery technologies for untethered robot," *Energies*, vol. 16, no. 13, 2023, Art. no. 4904, doi: 10.3390/en16134904.

[16] M. Pierallini, R. K. M. Gopanunni, F. Angelini, A. Bicchi, and M. Garabini, "Fishing for data: Modeling, optimal planning, and iterative learning control for flexible link robots," *IEEE Trans. Control Syst. Technol.*, 2025, early access, doi: 10.1109/TCST.2025.3550030.

[17] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal Parkour: Learning agile navigation for quadrupedal robots," *Sci. Robot.*, vol. 9, no. 88, 2024, Art. no. eadi7566, doi: 10.1126/scirobotics.adi7566.

[18] A. Spiridonov et al., "SpaceHopper: A small-scale legged robot for exploring low-gravity celestial bodies," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 3464–3470.

[19] H. Kolvenbach, P. Arm, G. Valsecchi, N. Rudin, and M. Hutter, "Legged systems for exploration," in *Space Robotics: The State of the Art and Future Trends*. Berlin, Germany: Springer, 2024, pp. 135–156.

[20] L. Sha, A. Lin, Q. Xi, and S. Kuang, "A topology optimization method for robot light-weight design under multi-working conditions and its application on upper-limb powered exoskeleton," in *Proc. Int. Conf. Artif. Intell. Electromech. Automat.*, 2020, pp. 17–22, doi: 10.1109/AIEA51086.2020.00011.

[21] M. Bugday and M. Karali, "Design optimization of industrial robot arm to minimize redundant weight," *Eng. Sci. Technol., Int. J.*, vol. 22, pp. 346–352, 2018, doi: 10.1016/j.jestch.2018.11.009.

[22] P. de Santos, E. Garcia, and J. Estremera, *Quadrupedal Locomotion: An Introduction to the Control of Four-Legged Robots*. London, U.K.: Springer, 2007.

[23] F. Grimminger et al., "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3650–3657, Apr. 2020, doi: 10.1109/LRA.2020.2976639.

[24] S. P. Chhatoi, M. Pierallini, F. Angelini, C. Mastalli, and M. Garabini, "Optimal control for articulated soft robots," *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3671–3685, Oct. 2023.

[25] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "DTC: Deep tracking control," *Sci. Robot.*, vol. 9, no. 86, 2024, Art. no. eadh5401, doi: 10.1126/scirobotics.adh5401.

[26] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau5872, doi: 10.1126/scirobotics.aau5872.

[27] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proc. 5th Conf. Robot. Learn.*, A. Faust, D. Hsu, and G. Neumann, Eds., Nov. 2022, vol. 164, pp. 91–100. [Online]. Available: https://proceedings.mlr.press/v164/rudin22a.html

[28] D. Todd, *Walking Machines: An Introduction to Legged Robots, Ser. Chapman and Hall Advanced Industrial Technology Series*. Berlin, Germany: Springer, 2013.

[29] C. Mastalli et al., "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2536–2542.

[30] H. Chai et al., "A survey of the development of quadruped robots: Joint configuration, dynamic locomotion control method and mobile manipulation approach," *Biomimetic Intell. Robot.*, vol. 2, no. 1, 2022, Art. no. 100029, doi: 10.1016/j.birob.2021.100029.

[31] A. Fukuhara, M. Gunji, and Y. Masuda, "Comparative anatomy of quadruped robots and animals: A review," *Adv. Robot.*, vol. 36, pp. 1–19, 2022.

[32] M. Hutter, C. Gehring, M. Bloesch, M. Hoepflinger, C. Remy, and R. Siegwart, "Starleth: A compliant quadrupedal robot for fast, efficient, and versatile locomotion," in *Adaptive Mobile Robotics*. Singapore: World Scientific, Sep. 2012, pp. 483–490.

[33] J. Pratt and B. Krupp, "Series elastic actuators for legged robots," *Proc. SPIE*, pp. 135–144, 2004, doi: 10.1117/12.548000.

[34] S. Kim and P. Wensing, "Design of dynamic legged robots," *Found. Trends Robot.*, vol. 5, pp. 117–190, Jan. 2017.

[35] D. Häufle, M. Taylor, S. Schmitt, and H. Geyer, "A clutched parallel elastic actuator concept: Towards energy efficient powered legs in prosthetics and robotics," in *Proc. IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatronics*, 2012, pp. 1614–1619, doi: 10.1109/BioRob.2012.6290722.

[36] S. Liu, J. Ding, C. Lu, Z. Wang, B. Su, and Z. Guo, "A novel optimization design of dual-slide parallel elastic actuator for legged robots," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 4, pp. 2886–2894, Aug. 2024, doi: 10.1109/TMECH.2024.3401546.

[37] Y. Fan, Z. Pei, C. Wang, M. Li, Z. Tang, and Q. Liu, "A review of quadruped robots: Structure, control, and autonomous motion," *Adv. Intell. Syst.*, vol. 6, no. 6, 2024, Art. no. 2300783.
[38] M. G. Catalano et al., "Adaptive feet for quadrupedal walkers," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 302–316, Feb. 2022.
[39] A. Ranjan, F. Angelini, T. Nanayakkara, and M. Garabini, "Design guidelines for bioinspired adaptive foot for stable interaction with the environment," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 2, pp. 843–855, Apr. 2024.
[40] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: MIT Press, 1986.
[41] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for torque-controlled legged robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4730–4737.
[42] D. De Benedittis, F. Angelini, and M. Garabini, "Soft bilinear inverted pendulum: A model to enable locomotion with soft contacts," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 55, no. 2, pp. 1478–1491, Feb. 2025.
[43] C. Gehring, C. Dario Bellicoso, P. Fankhauser, S. Coros, and M. Hutter, "Quadrupedal locomotion using trajectory optimization and hierarchical whole body control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 4788–4794, doi: 10.1109/ICRA.2017.7989557.
[44] M. Neunert et al., "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018, doi: 10.1109/LRA.2018.2800124.
[45] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 2261–2268, Jul. 2018, doi: 10.1109/LRA.2018.2794620.
[46] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 2497–2503, doi: 10.1109/IROS47612.2022.9981198.
[47] Y. Kim, B. Son, and D. Lee, "Learning multiple gaits of quadruped robot using hierarchical reinforcement learning," 2021. [Online]. Available: https://arxiv.org/abs/2112.04741
[48] G. Patil, G. Lakhekar, and J. Karajagikar, "Quadruped robot locomotion control using reinforcement learning," in *Proc. IEEE 3rd Int. Conf. Power Electron., Intell. Control Energy Syst.*, 2024, pp. 424–429, doi: 10.1109/ICPEICES62430.2024.10719228.
[49] J. Jabbour et al., "Closing the sim-to-real gap for ultra-low-cost resource-constrained quadruped robot platforms," in *Proc. 3rd Workshop Closing Reality Gap Sim2Real Transfer Robot.*, Robotics: Science and Systems, Jun. 2022.
[50] D. Makoviichuk and V. Makoviychuk, "RL-Games: A high-performance framework for reinforcement learning," May 2021. [Online]. Available: https://github.com/Denys88/rl_games
[51] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[52] A. Scaldaferri, F. Angelini, and M. Garabini, "Learning to walk with adaptive feet," *Robotics*, vol. 13, no. 8, 2024, Art. no. 113, doi: 10.3390/robotics13080113. [Online]. Available: https://www.mdpi.com/2218-6581/13/8/113
[53] V. Makoviychuk et al., "ISAAC gym: High performance GPU based physics simulation for robot learning," in *Proc. 35th Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=fgFBtYgJQX_
[54] NVIDIA, "Isaac sim—Robotics simulation and synthetic data generation," Mar. 2022. [Online]. Available: https://developer.nvidia.com/isaac-sim
[55] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, New York, NY, USA, 2009, pp. 41–48, doi: 10.1145/1553374.1553380.
[56] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "POET: open-ended coevolution of environments and their optimized solutions," *Proc. GECCO '19: Genet. Evol. Comput. Conf.*, pp. 142–151, Jul. 2019.
[57] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, 2019, pp. 350–360.
[58] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, J. Dy and A. Krause, Eds., Jul. 2018, vol. 80, pp. 4045–4054. [Online]. Available: https://proceedings.mlr.press/v80/pardo18a.html
[59] NVIDIA, "Closing the sim-to-real gap: Training spot quadruped locomotion with NVIDIA isaac lab." Accessed: May 18, 2025. [Online]. Available: https://developer.nvidia.com/blog/closing-the-sim-to-real-gap-training-spot-quadruped-locomotion-with-nvidia-isaac-lab/
[60] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2020, pp. 737–744, doi: 10.1109/SSCI47803.2020.9308468.
[61] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, vol. 9, pp. 153171–153187, 2021, doi: 10.1109/ACCESS.2021.3126658.
[62] N. Liu, Y. Cai, T. Lu, R. Wang, and S. Wang, "Real–sim–real transfer for real-world robot control policy learning with deep reinforcement learning," *Appl. Sci.*, vol. 10, no. 5, 2020, Art. no. 1555, doi: 10.3390/app10051555.
[63] S. Gupta, G. Singal, and D. Garg, "Deep reinforcement learning techniques in diversified domains: A survey," *Arch. Comput. Methods Eng.*, vol. 28, pp. 4715–4754, 2021, doi: 10.1007/s11831-021-09552-3.
[64] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2817–2826.
[65] F. Shi, C. Zhang, T. Miki, J. Lee, M. Hutter, and S. Coros, "Rethinking robustness assessment: Adversarial attacks on learning-based quadrupedal locomotion controllers," 2024. [Online]. Available: https://arxiv.org/abs/2405.12424
[66] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 3803–3810, doi: 10.1109/ICRA.2018.8460528.
[67] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning; lessons we've learned," *J. Robot. Res.*, vol. 40, no. 4–5, pp. 698–721, 2021, doi: 10.1177/0278364920987859.
[68] S. Singh, R. P. Russell, and P. M. Wensing, "Multi-shooting differential dynamic programming for hybrid systems using analytical derivatives," 2023, *arXiv:2307.12606*.
[69] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *Proc. IEEE-RAS 18th Int. Conf. Humanoid Robots*, 2018, pp. 1–9.
[70] M. Aractingi, P.-A. Léziart, T. Flayols, J. Perez, T. Silander, and P. Souères, "Controlling the solo12 quadruped robot with deep reinforcement learning," *Sci. Rep.*, vol. 13, 2023, Art. no. 11945, doi: 10.1038/s41598-023-38259-7.

ANTONELLO SCALDAFERRI received the Master of Science degree in computer engineering from the University of Studies of Salerno, Fisciano, Italy, in 2020. He is currently working toward the Ph.D. degree in robotics engineering with the Research Center "E. Piaggio," University of Pisa, Pisa, Italy.

He worked with the Active Perception and Robot Interactive Learning Laboratory, Italian Institute of Technology, as a Research Fellow in the advanced robotics research line, on the VINUM project, which concerns robotic grapevine winter pruning automation. His current research focuses on reinforcement-learning-based locomotion planning and control of quadrupedal robots for highly dynamic motor skills in natural environments, in the context of the Natural Intelligence H2020 EU project, concerning robotic environmental monitoring of natural habitats.

SIMONE TOLOMEI received the B.S. degree in electrical engineering in 2019 and the M.S. degree (cum laude) in automation and robotics engineering in 2022 from the University of Pisa, Pisa, Italy, where he is currently working toward the Ph.D. degree in robotics and intelligent machines.

His main research interests include reinforcement learning for robotics, mobile manipulation, and locomanipulation.

FRANCESCO IOTTI (Student Member, IEEE) received the B.S. degree in mechatronics engineering in 2020 from the University of Modena and Reggio Emilia, Reggio Emilia, Italy, and the M.S. degree in automation and robotics engineering in 2023 from the University of Pisa, Pisa, Italy, where he is currently working toward the Ph.D. degree in robotics and intelligent machines.

His main research interests include multimodal robot design and automatic control.

PAOLO GAMBINO received the B.S. degree in mechanical engineering and the M.S. degree (cum laude) degree in automation and robotics engineering from the University of Pisa, Pisa, Italy, in 2021 and 2024, respectively.

His main research interests include reinforcement learning for motion planning and mobility tasks, dynamic control, and identification of robots in natural environments.

MICHELE PIERALLINI (Member, IEEE) received the B.S. degree in biomedical engineering in 2017, the M.S. degree (cum laude) in automation and robotics engineering in 2020, and the Ph.D. degree in robotics in 2024 from the University of Pisa, Pisa, Italy, where he is currently working toward the Postdoctoral degree in robotics.

His current research interests include the planning and controlling soft robotic systems and learning control.

FRANCO ANGELINI (Member, IEEE) received the B.S. degree in computer engineering, the M.S. degree (cum laude) in automation and robotics engineering, and the Ph.D. degree (cum laude) in robotics from the University of Pisa, Pisa, Italy, in 2013, 2016, and 2020, respectively.

He is currently an Assistant Professor with the University of Pisa. His main research interests include control of soft robotic systems, robotic environmental monitoring, grasping, and impedance planning.

MANOLO GARABINI (Member, IEEE) received the graduation degree in mechanical engineering and the Ph.D. degree in robotics from the University of Pisa, Pisa, Italy, in 2010 and 2014, respectively.

He is currently a Professor with the University of Pisa. He is currently the Principal Investigator with the THING H2020 EU Research Project for the University of Pisa and the Coordinator of the Dysturbance H2020 Eurobench Subproject. He is also the Coordinator of the H2020 EU Research Project Natural Intelligence. His main research interests include the design, planning, and control of soft adaptive robots.