# Learning System Adaptive Legged Robotic Locomotion Policies




**Siddhant Gangapurwala**

St. Anne's College

University of Oxford

Supervisors: Dr. Ioannis Havoutis and Prof. Ingmar Posner

A thesis submitted for the degree of
*Doctor of Philosophy*

20th February 2022

# Abstract

The ability to form support contacts at discontinuous locations makes legged robots suitable for locomotion over highly unstructured terrains. While recent years have witnessed significant robotic developments, delivering extremely dynamic and robust hardware solutions, the control intelligence for legged robots to perform agile and sophisticated maneuvers remains an active area of research. This thesis, therefore, focuses on the control of legged systems, particularly, quadrupedal robots.

The research presented in this thesis is driven by the motivation that *a controller governing the behavior of a system should thoroughly utilize its potential while also adapting to variations in system dynamics through emergence of behavior that still achieves the control objective.*

Sampling-based search methods allow explorations along vast regions of operation, thereby, enabling discovery of near-optimal solutions. Similarly, data-driven reinforcement learning (RL) strategies allow development of controllers which exploit system dynamics to achieve the control objective described by a high-level reward function. The problem of legged robot locomotion can thus be approached using reinforcement learning to obtain robust and dynamic control solutions. Additionally, the control policy describing the behavior of the system can be parameterized as a deep neural network to perform complex non-linear mapping of the robot state information to desired control action. This approach of utilizing a deep neural network is then referred to as *deep* reinforcement learning. In this dissertation I focus on employing deep RL strategies for quadrupedal locomotion.

I present that encouraging the RL control policy to even implicitly model system dynamics allows for emergence of adaptive control behaviors. I use this observation throughout this thesis to develop system adaptive control strategies leading up to an ambitious goal of obtaining an RL locomotion policy capable of zero-shot transfer to quadrupeds of varying kinematic and dynamic properties.

Although the main contributions relate to RL, I also recognize the benefits and drawbacks of different control approaches, and thus explore modular control architectures which utilize both model-based and data-driven model-free strategies for

robot locomotion. In this regard, I present training and control architectures which exhibit dynamic locomotion behavior over terrains with varying steps and inclines, both in lab experiments and field trials.

In combination, the works presented in this thesis investigate, and, in my firm belief, further the state of legged robotic control advocating for development of artificial control intelligence which matches the level of complexity demonstrated by massively evolved biological counterparts.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. The thesis is entirely my own work, and except where otherwise stated, describes my own research.

Siddhant Gangapurwala, November 2021.

*This thesis is dedicated to my parents, Mrs. Sangita and Justice Sanjay Gangapurwala, and my sister, Dr. Sonica Gupta, for their boundless love and unconditional support.*

# Acknowledgements

I express my sincere gratitude to my supervisor Dr. Ioannis Havoutis for his consistent support during my DPhil study. Without his motivation, enthusiasm and guidance, this thesis would not have been possible. I also thank Prof. Ingmar Posner for his support and advice.

Besides my advisors, I have been extremely fortunate and grateful to work with Prof. Jemin Hwangbo who has been a true mentor helping me navigate through my field of research.

I also express my gratitude to my collaborators Dr. Mathieu Geisert, with whom I worked on several topics in and outside of this thesis and without whom research would have not been as productive, and Dr. Romeo Orsolino who has significantly contributed to my research and with whom work was always fun. I also thank the members of DRS group for their support.

I thank the EPSRC AIMS CDT and the EU Memory of Motion Studentship for funding my research.

Within the AIMS CDT, I am particularly indebted to the encouraging support of Wendy Poole. Her tremendous efforts and passion for work made my doctoral research experience extremely pleasant.

I thank my grandparents, uncles, aunts, cousins and friends for their love and encouragement, helping me grow as a person. I consider myself privileged to have met amazing people during the course of my DPhil study providing me with the inspiration to keep marching on. I also look forward to continue working on ambitious projects with David, Paarth and Frank.

I greatly appreciate the help of M.K. who made it possible for my move to the UK to be as smooth as it could ever be. Thank you Caitlin for being a fan and always believing in me. And most importantly, thanks to my nephews, Aadi and Bunny, for being a constant source of joy and happiness.

# Contents

**1 Introduction** **1**

1.1 Motivation and Objectives . . . . . 1

1.2 Key Ideas and Literature Review . . . . . 4

1.2.1 Dynamic Models . . . . . 5

1.2.2 Stability Criteria . . . . . 9

1.2.3 Control Methods . . . . . 11

1.3 Thesis Organization . . . . . 19

**2 Contributions** **20**

2.1 Paper I . . . . . 20

2.2 Paper II . . . . . 26

2.3 Paper III . . . . . 28

2.4 Paper IV . . . . . 30

2.5 Paper V . . . . . 32

**3 Paper I**
**Guided Constrained Policy Optimization**
**for Dynamic Quadrupedal Robot Locomotion** **35**

**4 Paper II**
**Rapid Stability Margin Estimation for**
**Contact-Rich Locomotion** **46**

**5 Paper III**
**Real-Time Trajectory Adaptation for Quadrupedal Locomotion using Deep Reinforcement Learning** **56**

**6 Paper IV**
**RLOC: Terrain-Aware Quadrupedal Locomotion using Reinforcement Learning and Optimal Control** **66**

**7 Paper V**
**Learning Dynamics Conditioned Platform Adaptive Quadrupedal Locomotion Policy** **94**

**8 Conclusion and Outlook** **111**

8.1 Achievements . . . . . 111

8.2 Future Directions . . . . . 112

**Bibliography** **114**

**Appendices**

# 1 Introduction

This chapter provides an overview of the research motivation, describes the research direction, and presents a review of existing literature.

## 1.1 Motivation and Objectives

Human history has been witness to several ideas introducing biologically-inspired legged machines [Silva and Tenreiro Machado, 2007]. This creative inspiration, in the $19^{th}$ century, led to designs of legged systems that relied on energy sources for generation of motions. These included the *Mechanical Horse* [Rygg, 1893] and the *Steam Man* [Rosheim, 1994]. Over the past century, the design focus shifted from replicating limbed biological systems to feasible mechanical robots that could be utilized to traverse complex terrains, however, these machines were still limited to quasi-static motions [Waldron and McGhee, 1986; Lim and Takanishi, 2007; Gurfinkel et al., 1981; Hirose and Umetani, 1980; Wieber et al., 2016]. It was over the past few decades that the robotics community observed momentous evolution of legged systems that could display much more dynamic behavior, such as the hydraulically actuated robots, *BigDog* [Raibert et al., 2008] and *HyQ* [Semini et al., 2011]. In contrast

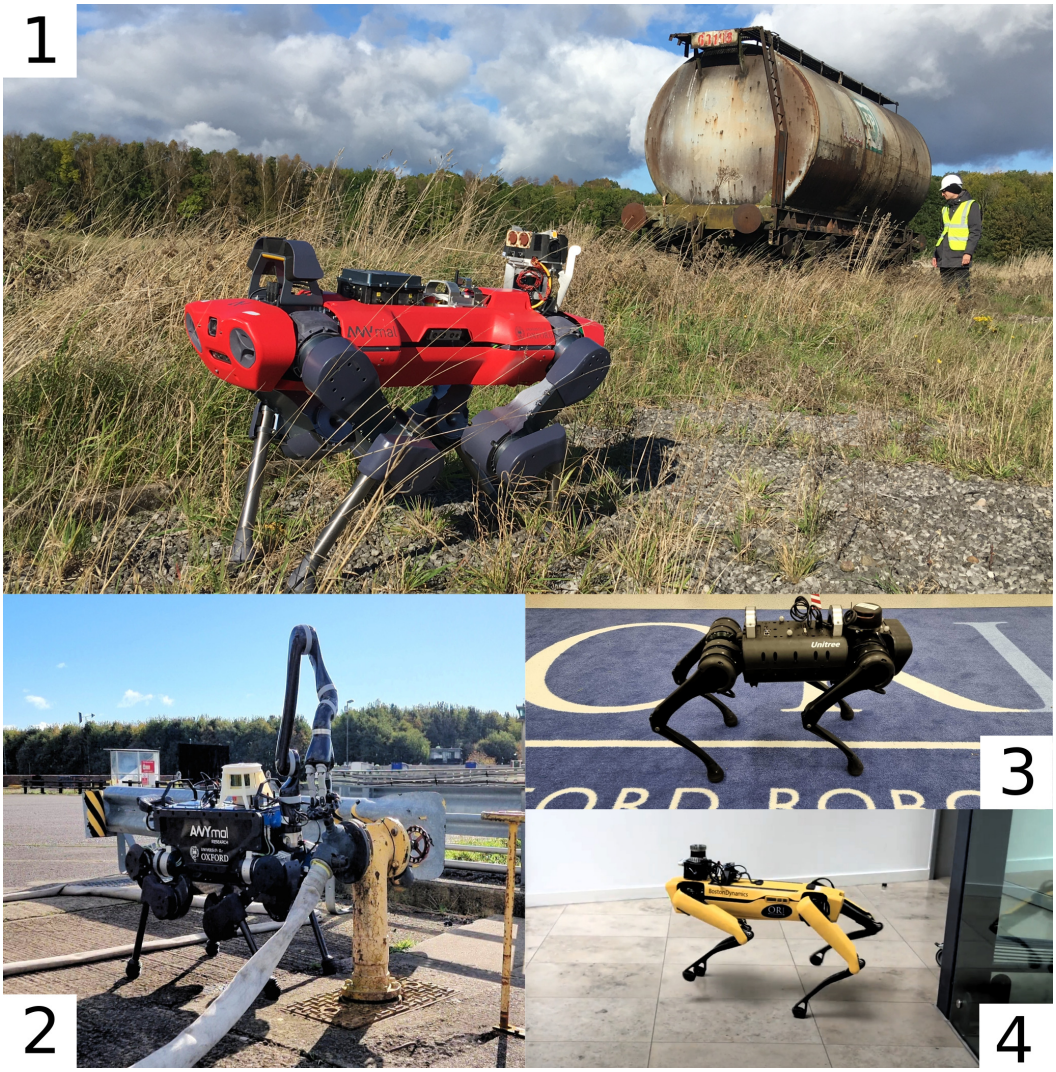



**Figure 1.1:** The (1) ANYmal C, (2) ANYmal B, (3) A1, and (4) Spot quadrupedal platforms deployed indoors and outdoors.

to hydraulic actuators (which are suitable for high-load applications but suffer from the need for continuous monitoring, complex power units, and fluid leakage), electric actuators offer extremely precise control in a desirably small form-factor. Electrically-actuated robots such as *MIT Cheetah* [Seok et al., 2013], *Spot* [Ackerman, 2016], and *Cassie* [Ackerman, 2017] have demonstrated impressive agile locomotion skills and are further applicable for indoor and constrained use.

These recent developments in the field of legged robotics have mainly been motivated by their potential for deployment in highly unstructured environments. This is characterized by their ability to form discrete support contacts with the environment as required,

making them suitable for a wide range of applications such as remote monitoring of offshore platforms [Brandão et al., 2020], nuclear decommissioning [ICSF/EPSRC RAIN Hub], inspection of sewer systems [Kolvenbach et al., 2020], and rescue missions in disaster areas [Bellicoso et al., 2018a].

While the advances in hardware designs have led to delivery of rugged and dynamic robots such as *ANYmal B* [Hutter et al., 2016], *ANYmal C* [Ackerman, 2019], *A1* [Unitree] and *Vision 60* [Ghost Robotics], their capabilities are inhibited by control approaches which do not fully exploit the mechanical properties of the system, leading to behaviors which demonstrate limited efficiency, versatility, reactivity, and robustness [Wieber et al., 2016]. This is also true of controllers which assume an invariable system dynamics model wherein the control parameters are fine-tuned to exhibit optimal behavior for the modeled system dynamics, which, when transferred to domains not considered during development, result in reduced control performance. This adaptive property is relevant to cases such as:

- transfer of learned control policies from simulation to corresponding physical robot;
- change of robot mass due to added sensory or manipulation components;
- deployment for loading, carrying and unloading tasks resulting in variation of center of mass (CoM);
- inability to accurately model highly non-linear actuation dynamics; and
- change of mechanical properties due to continuous wear-and-tear.

The research work presented in this thesis is driven by the motivation that a controller should:

1. thoroughly utilize system capabilities; and
2. adapt to variations in system dynamics.

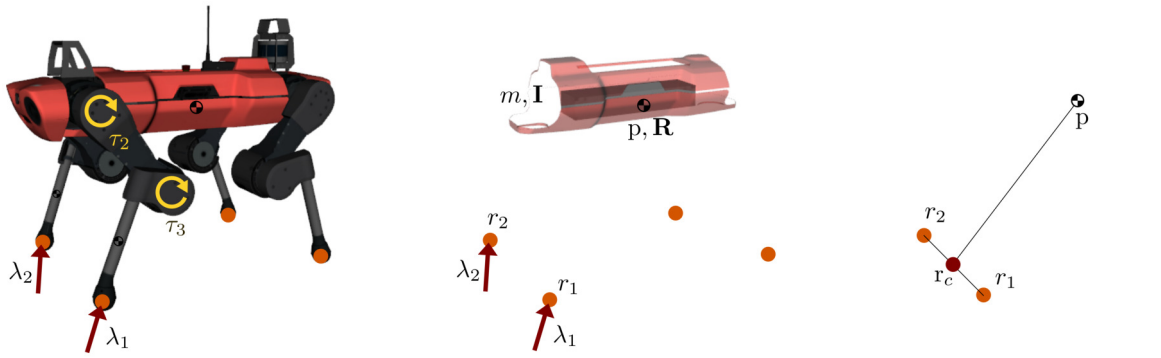
The goal of this thesis is therefore to investigate and present methods for adaptive control conditioned on system dynamics while exploiting system potential. Note that, in the context of this thesis, *thorough utilization of system capabilities* refers to the ability of a controller to exhibit behaviors that are not limited by modeling approximations or predefined templates. For example, the control approach presented in [Bellicoso et al., 2018b] describes a quadruped's base motion by a quintic spline. This motion predefinition restricts the resulting behavior thereby affecting the control agility. Moreover, system modeling approximations, detailed in Section 1.2.1, introduce operational assumptions in the control problem which further inhibit the robot's motions.

This work additionally explores modular control architectures which leverage a combination of model-based and model-free techniques to:

1. address sample inefficiency related to learning control policies;
2. guarantee safety-critical constrained operations;
3. enhance performance of previously proposed control architectures; and
4. perform zero-shot transfers to domains not explicitly modeled during development.

## 1.2 Key Ideas and Literature Review

The problem of legged robotic locomotion has been approached with a set of diverse planning and control solutions which either employ modeling techniques within a spectrum of analytical and learned system dynamics, or generate control behavior without utilizing an explicit dynamics model. This section discusses the various dynamics modeling formulations utilized for legged systems, provides an overview of robot stability estimation strategies, and presents the relevant control methods investigated in this thesis.




Rigid Body Dynamics | Single Rigid Body Dynamics | Linear Inverted Pendulum

**Figure 1.2:** Visual representations of the various dynamic models used for control of legged robots. Refer to the work of *Winkler* for a detailed description of these models [Winkler, 2018].

### 1.2.1 Dynamic Models

System dynamics describe the effect of a control input $\mathbf{u}$ to the system state x resulting in a transition expressed by $\dot{\mathrm{x}}$. The dynamics can then be modeled as

$$\dot{\mathrm{x}}(t) = \mathbf{F}(\mathrm{x}(t), \mathbf{u}(t)). \tag{1.1}$$

The state and control input representations vary depending upon the complexity of the model defined by $\mathbf{F}$. While certain architectures rely on relatively accurate models [Mordatch et al., 2012; Mastalli et al., 2020] that can generate complex multi-contact motions but often introduce significant control design and computational complexities, other research works develop upon simplified models which only capture the main system dynamics necessary for producing a desired behavior [Full and Koditschek, 1999]. These models are referred to as *templates* and are typically linear with respect to the system states. Examples of templates include the linear inverted pendulum (LIP) model [Kajita et al., 2001] and the spring-loaded inverted pendulum (SLIP) model [Poulakakis et al., 2006]. The inverse mapping relating a template to the corresponding whole-body dynamics model is called the *anchor*. This sub-section briefly presents the relevant dynamics modeling techniques. A visual representation of some of these models are provided in Figure 1.2.

### Rigid Body Dynamics

One approach to modeling system dynamics is through a combination of non-deformable rigid bodies with actuated connections. Based on this, for a general quadruped, comprising a floating base $B$ with four attached limbs containing actuated joints, the robot state can be described through its generalized coordinates, q, w.r.t. a global inertial reference frame $W$. In this regard, the base position is represented as ${}_W r_{WB} \in \mathbb{R}^3$. Here, the $WB$ post-subscript refers to the measurement of a quantity defined in frame $B$ relative to frame $W$. The $W$ pre-subscript is the frame in which the quantity is expressed. Therefore, ${}_W r_{WB}$ is the base position measured relative to and expressed in the global frame $W$. The base orientation, $\mathrm{q}_{WB} \in SO(3)$, is represented using Hamiltonian unit quaternions with a corresponding rotation matrix $\mathbf{R}_{WB} \in SO(3)$. The generalized coordinates can then be written as

$$\mathrm{q} = \begin{bmatrix} {}_W r_{WB} \\ \mathrm{q}_{WB} \\ \mathrm{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \tag{1.2}$$

where $\mathrm{q}_j \in \mathbb{R}^{n_j}$ denotes the joint positions. The effective motion generated due joint torque $\tau_j \in \mathbb{R}^{n_j}$ is described by the rigid body dynamics expressed as

$$\mathrm{M}(\mathrm{q})\ddot{\mathrm{q}} + \mathrm{h}(\mathrm{q}, \dot{\mathrm{q}}) = \mathbf{S}^T \tau + \mathrm{J}(q)^T \lambda, \tag{1.3}$$

where $\mathrm{M} \in \mathbb{R}^{(6+n_j)\times(6+n_j)}$ is the mass matrix relative to the joints, $\mathrm{h} \in \mathbb{R}^{6+n_j}$ comprises Coriolis, centrifugal and gravity terms, $\mathbf{S}^T = [\mathbf{0}_{n_j \times 6} \ \mathbf{I}_{n_j \times n_j}]^T$, and J is the Jacobian which maps the forces $\lambda \in \mathbb{R}^{n_f}$ at $n_f$ end-effectors to generalized forces. An example of work which utilized this model is that of *Neunert et al.* for model predictive control of quadrupedal systems [Neunert et al., 2018].

### Centroidal Dynamics

Equation 1.3 can be further split into separate equations describing the motion of unactuated base ($b$) and actuated joints ($j$) as shown below.

$$\mathrm{M}_b(\mathrm{q})\ddot{\mathrm{q}} + \mathrm{h}_b(\mathrm{q}, \dot{\mathrm{q}}) = \mathrm{J}_b(q)^T \lambda \ , \text{ and} \tag{1.4}$$

$$\mathrm{M}_j(\mathrm{q})\ddot{\mathrm{q}} + \mathrm{h}_j(\mathrm{q}, \dot{\mathrm{q}}) = \tau + \mathrm{J}_j(q)^T\lambda. \tag{1.5}$$

Reformulation of Equation 1.4 to obtain a representation of change of momentum in a frame anchored at the center of mass (CoM), p, can be expressed as

$$\mathbf{A}(\mathrm{q})\ddot{\mathrm{q}} + \dot{\mathbf{A}}(\mathrm{q})\dot{\mathrm{q}} = \begin{bmatrix} m\mathbf{g} + \sum_{f=1}^{n_f} \lambda_f \\ \sum_{f=1}^{n_f} \lambda_f \times (\mathrm{p}(\mathrm{q}) - r_f(\mathrm{q})) \end{bmatrix}, \tag{1.6}$$

and is referred to as centroidal dynamics. In the above equation, $\mathbf{A} \in \mathbb{R}^{6\times(6+n)}$ is the centroidal momentum matrix which maps the velocities of each of the bodies into a reference frame relative to the CoM, $m$ is the mass of the robot, $\mathbf{g}$ is the acceleration due to gravity, and $\mathrm{r}_f$ is the position of $f$-th end-effector. The work of *Dai et al* leverages this formulation to obtain highly-dynamic motion plans [Dai et al., 2014] for the Atlas humanoid and LittleDog quadruped.

### Single Rigid Body Dynamics

While Equation 1.3 and Equation 1.6 depend upon joint states which contribute to the non-linearity of the formulation, by introducing the assumptions: (1) joint velocities produce negligible momentum, and (2) whole-body inertia remains similar to that in nominal configuration, the centroidal dynamics can be simplified to obtain the Newton-Euler formulation describing the linear and angular motion of a single rigid body given by

$$m\ddot{\mathrm{p}} = m\mathbf{g} + \sum_{f=1}^{n_f} \lambda_f, \text{ and} \tag{1.7}$$

$$\mathbf{I}(\mathbf{R})\dot{\omega} + \omega \times \mathbf{I}(\mathbf{R})\omega = \sum_{f=1}^{n_f} \lambda_f \times (\mathrm{p} - \mathrm{r}_f), \tag{1.8}$$

where $m$ is the combined system mass, $\omega$ is the angular velocity, $\mathbf{R}$ refers to the base orientation, and $\mathbf{I}(\mathbf{R}) \in \mathbb{R}^{3\times3}$ is the combined whole-body inertia of individual limbs expressed in the nominal joint configuration. This formulation is especially relevant to systems such as the Spot, A1 and HyQ for which much of the system mass is concentrated at the robot's base. This model, however, is insufficiently accurate for robots such as humanoids where the mass is distributed throughout the links and the inertial characteristics are significantly dependent on the whole-body as opposed to only the base. *Winkler et al.* demonstrated generation of highly dynamic motion plans for monopeds, bipeds and quadrupeds using this model [Winkler et al., 2018].

### Linear Inverted Pendulum

The single rigid body dynamics model can be further simplified by introducing the assumptions: (1) $\omega = 0$ and $\dot{\omega} = 0$, (2) CoM height, $\mathrm{p}_z$, remains constant, and (3) height of footholds, $\mathrm{r}_{f_z}$, remains constant. In this regard, the $x$ component of the horizontal motion can be rewritten as

$$m\ddot{\mathrm{p}}_x = \frac{m\mathbf{g}}{h}(\mathrm{p}_x - \mathrm{r}_{c_x}), \tag{1.9}$$

where $h$ represents the base height, and $\mathrm{r}_{c_x}$ is the center or pressure (CoP) expressed as

$$\mathrm{r}_{c_x} = \frac{\sum_{f=1}^{n_f} \lambda_{f_z} \mathrm{r}_{\mathrm{f_x}}}{\sum_{f=1}^{n_f} \lambda_{f_z}}. \tag{1.10}$$

The $y$ component of horizontal motion can be similarly expressed. Despite its simplicity and strong assumptions, this model was central to early works for calculating footfall positions for both bipeds and quadrupeds [Raibert, 1986]. The work of *Raibert* was adapted for succeeding research. For example, *Kajita et al.* presented the use of this model to generate bipedal walking patterns [Kajita et al., 2001]. The linear inverted pendulum model is still widely used to develop control solutions for legged systems.

### Learned System Dynamics

While the modeling techniques described above relate to explicit analytical representations of system dynamics, several concepts which utilize a *learned* dynamics model to then obtain a control policy have also been proposed [Werbos, 1987, 1989; Nguyen and Widrow, 1990; Schmidhuber, 1991]. In this direction, *Deisenroth et al.* introduced an efficient model-based policy search approach, PILCO [Deisenroth and Rasmussen, 2011], for control of complex systems. PILCO learns the system dynamics expressed in Equation 1.1 using a Gaussian process (GP) model, and uses this model to sample trajectories which are then used to train a controller. As an alternative to GPs, which is suitable for low-dimensional embeddings, recent works utilize Bayesian neural networks to learn system dynamics [Gal et al., 2016; Depeweg et al., 2016]. In order to learn temporal characteristics to better encode system dynamics,

the use of recurrent neural networks (RNNs) has also been presented [Schmidhuber, 2015]. *Ha et al.* further introduced an approach of employing a generative RNN to learn *world models* that are used to obtain control policies through evolutionary strategies [Ha and Schmidhuber, 2018]. Extending the concept of world models, *Sekar et al.* proposed Plan2Explore [Sekar et al., 2020] to perform zero or few-shot adaptations of tasks during evaluation.

As an alternative to learning the whole system dynamics model, analytically represented models can be further fine tuned through data-driven techniques. In this regard, *Grandia et al.* [Grandia et al., 2018] presented an approach to learn error terms for rigid body dynamics and actuation dynamics using Gaussian process regression. Extending this idea of fine tuning through learning, *Chebotar et al.* further introduced a framework to address the simulation-reality gap by learning the simulation parameters through policy rollouts in the real world [Chebotar et al., 2019].

### 1.2.2 Stability Criteria

To obtain motion plans, a stability criterion is typically introduced in the control problem. Note that, here, the notion of stability refers to the ability of a legged system to maintain balance during locomotion. This is different from the concept of Lyapunov stability often utilized in classical control system design to describe the stability of control solutions near an equilibrium point [Shevitz and Paden, 1994]. With regards to legged robots, the static stability of the system can be described with *fixed points* that represent the configurations of the robot in which it can stand still [McGhee and Frank, 1968]. This can be expressed as a solution to Equation 1.1 where a control input $\mathbf{u^p}$ exists, such that, a fixed point denoting the robot state, $\mathrm{q^p}$, maintains its configuration, implying, $\dot{\mathrm{q}}^{\mathbf{p}} = 0$ and $\ddot{\mathrm{q}}^{\mathbf{p}} = 0$. This relates to the condition given by

$$\mathrm{p}_{xy}^{\mathbf{eg}} \in \mathrm{conv}\{\mathrm{r}_{\mathrm{f_{xy}}}\}, \tag{1.11}$$

where $\mathbf{e^g}$ refers to the gravity axis and $\mathrm{p}_{xy}^{\mathbf{eg}}$ is the projection of the CoM along the direction of gravity. The condition can thus be stated as: a legged system is statically

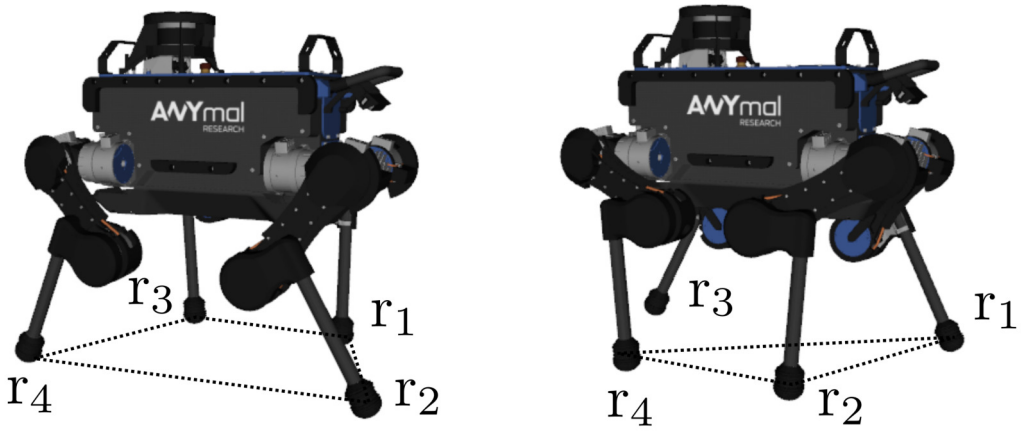



**Figure 1.3:** The support polygon, denoted by the dotted lines, is defined as the convex hull of the foot support area. In the above representations, the vertices of the support polygon are described by sets of contact points $\{r_1, r_2, r_3, r_4\}$ (left) and $\{r_1, r_2, r_4\}$ (right).

stable when its CoM, projected along the gravity axis, lies within the *support polygon*, described as the convex hull of contact points, $r_{f_{xy}}$. The *stability margin* can then be defined as the distance between the reference point, in this case, the CoM, and the boundaries of the support polygon. The visual representation of the support polygon is provided in Figure 1.3. Although the fixed point criterion, represented by Equation 1.11, has been demonstrated for slow walking [McGhee and Frank, 1968], it is insufficient to represent stability of the robot where the support polygon collapses to a line or a point which is often the case for fast walking and running gaits. In this regard, the zero-tilting moment point (ZMP) [Vukobratović and Stepanenko, 1972] and the more recent instantaneous capture point (ICP) [Pratt et al., 2006] have been proposed. The dynamic stability is then characterized by the projection of these 2D points along the gravity axis within the support polygon. Both, ZMP and ICP, based on the linear inverted pendulum template, are only suitable for coplanar contacts and their descriptive reliability diminishes for complex environment geometries. Instead, the 3D virtual repellent point [Englsberger et al., 2015] and the 6D aggregated centroidal wrench [Orin et al., 2013], which also consider the vertical and angular accelerations, have been proposed. These approaches, however, are often numerically inefficient to be introduced for real-time control. In contrast to the above methods which are described by the reference points and corresponding support regions, efficient evaluation of robot

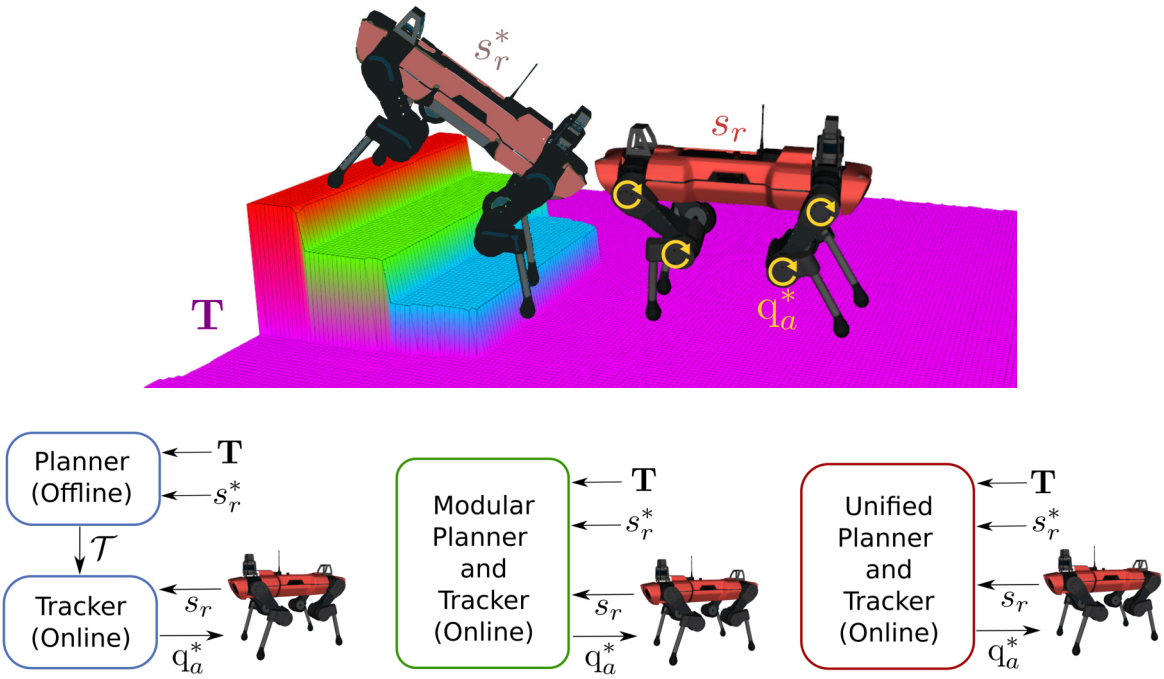



**Figure 1.4:** Different control architectures used for legged locomotion. Here, $s_r$ represents the current robot state information, $s_r^*$ is the control objective denoting the desired robot state, $\mathbf{T}$ refers to terrain features, $\mathcal{T}$ represents robot state trajectory, and $q_a^*$ refers to the actuation commands.

stability directly through contact forces [Herzog et al., 2015; Carpentier and Mansard, 2018] for multi-contact locomotion has also been presented.

### 1.2.3 Control Methods

While the majority of the proposed control architectures share similarities with regards to modularity wherein a control method employs a *planner*, to generate motion plans, and a *tracker*, which generates low-level actuation commands, this dissertation categorizes these architectures into three types as shown in Figure 1.4. Note that the tracker minimizes the error between the desired system state obtained from the generated motion plan and the measured system state. The control architectures are summarized below:

1. *Offline-Planning and Online-Tracking:* Trajectory optimization solvers which rely on accurate system dynamic models are typically computationally expensive

to be executed online during locomotion. This problem is often addressed by running the optimization offline to generate motion plans that are then tracked online [Posa et al., 2014; Carius et al., 2018]. This open-loop architecture therefore introduces two independent planning and tracking modules necessitating precise parameter tuning to enable accurate motion plan tracking. Paper II, presented in Chapter 5, employs this control method to perform online tracking of motion plans generated offline.

2. *Online Modular Planning and Tracking:* Alternatively, the trajectory optimization problem can be executed online in a receding horizon framework by introducing strong assumptions on the system dynamics, pre-specifying contact points, and generating short-horizon motion plans. The trajectory can thus be continuously adapted during tracking using a model-predictive control (MPC) approach [Farshidian et al., 2017a; Aceituno-Cabezas et al., 2017; Mastalli et al., 2020]. Another method that utilizes online planning and tracking architecture is based on distributing the control problem into smaller decoupled submodules. For example, *Kalakrishna et al.* proposed a modular framework for terrain-aware quadrupedal locomotion. The work in [Kalakrishnan et al., 2010] employed a separate footstep planner, a CoM motion planner and a foot motion planner to achieve fast and robust locomotion over uneven terrain. The more recent work of *Bellicoso et al.* demonstrated dynamic outdoor locomotion, with the ability to perform smooth gait transitions [Bellicoso et al., 2018b]. This architecture is relevant to the modular perceptive locomotion framework introduced in Paper IV. This work is presented in Chapter 6.

3. *Online Unified Planning and Tracking:* In contrast to above methods, which rely on explicit formulation of motion plans, parameterizing a control policy as a deep neural network enables direct mapping between the current state of the robot to low-level actuation commands. The input to the network may additionally include the control objective to allow implicit motion planning. This is especially relevant to deep reinforcement learning solutions. In these cases, the neural

network control policy is typically trained to embed a motion plan conditioned on the control signal and tracking behavior. For example, *Hwangbo et al.* utilized a history of joint states and tracking errors, along with a velocity command, to train a deep RL control policy which demonstrated dynamic locomotion behavior while being reactive to perturbations [Hwangbo et al., 2019]. Chapter 3 (Paper I) and Chapter 7 (Paper V) build upon this control method. In both of these works, a control policy, parameterized as a neural network, is used to perform a direct mapping of robot state information to a control action.

The rest of this subsection describes the relevant control approaches investigated in this thesis.

### Trajectory Optimization

One of the earliest and prominent approaches for legged robot motion planning has relied on numerical optimization [Chow and Jacobson, 1971] to obtain system trajectory by minimizing a cost function [Kuindersma et al., 2014]. This *trajectory optimization* (TO) approach to motion planning has been utilized to demonstrate remarkably complex maneuvers on humanoids and quadrupeds [Mordatch et al., 2012; Winkler et al., 2018]. A popular formulation of TO is through transcription to a nonlinear programming problem (NLP) which can be expressed as

$$\min_{\zeta} \; f(\zeta) \qquad \text{subject to} \quad g(\zeta) = 0, \quad h(\zeta) \leq 0, \tag{1.12}$$

where $\zeta$ represents the optimization variables, $f(\zeta)$ is the cost function, $g(\zeta)$ denotes the equality constraints, and $h(\zeta)$ denotes the inequality constraints. Such a formulation can then be solved using techniques such as interior point method [Mehrotra, 1992] and sequential quadratic programming (SQP) [Nocedal and Wright, 2006].

The appeal of TO is based on the emergence of sophisticated behaviors for diverse tasks without the need for specifically designed control architectures. This general solution, however, is computationally expensive for the complex task of legged locomotion, and

is therefore often approached through a series of strong assumptions. One typical assumption confines the system model to make contacts with the environment at specific points. Therefore, with the limited contact points, the optimal control trajectory can be obtained through a combinatorial search problem. In this direction, *Mordatch et al.* demonstrated convex optimization strategy, employing simplified system dynamics models, for character animation [Mordatch et al., 2012]. Alternatively, a popular approach for trajectory optimization has involved specifying contact sequences [Farshidian et al., 2017a], while some works also introduced contact timings in the optimization problem [Koch et al., 2012; Farshidian et al., 2017b].

Although techniques based on whole-body TO can generate complex motions, these methods typically involve huge computational costs which make them unsuitable for online optimization. In contrast, recent works which formulate TO as a differential dynamic programming (DDP) problem [Grandia et al., 2019; Mastalli et al., 2020] have shown efficient computation of whole-body motions that can be executed online. These methods, however, have yet to demonstrate their effectiveness on complex multi-contact environments. Paper III, presented in Chapter 5, utilizes a trajectory optimization solver [Winkler et al., 2018] to generate long-horizon quadrupedal motion plans.

### Hierarchical Optimization

The complexity associated with whole-body motion planning and control algorithms can be addressed by representing the locomotion problem, $\mathbf{P}$, as a set of simpler tasks described by linear equality or inequality constraints [Herzog et al., 2016]. For a solution vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{P}$ can be expressed as

$$\mathbf{T} : \begin{cases} \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{w} \\ \mathbf{D}\mathbf{x} - \mathbf{f} \leq \mathbf{v} \end{cases}, \tag{1.13}$$

where $\mathbf{w}$ and $\mathbf{v}$ are slack variables to be minimized. The sub-tasks can then be solved through priority based hierarchical optimization (HO) to obtain an optimal solution. In the context of legged whole-body control, these sub-tasks may include (1) maintaining base orientation, (2) tracking feet motion plans, and (3) following

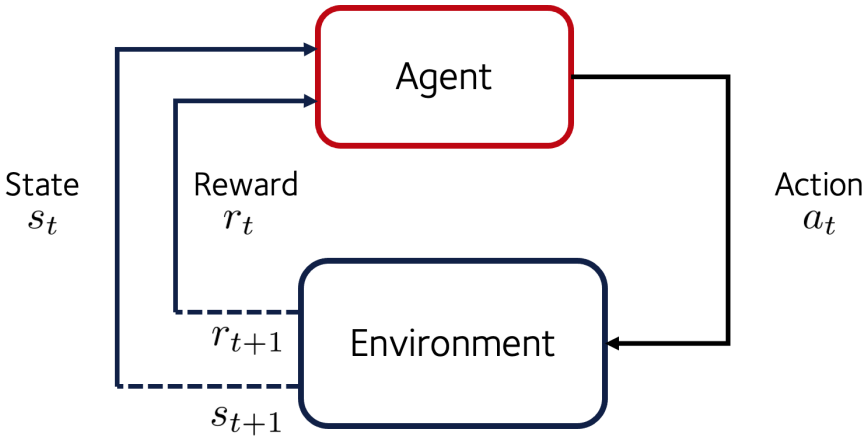



**Figure 1.5:** Interaction of an agent with the environment described in the framework of a Markov Decision Process [Sutton et al., 1998].

contact force constraints. The early implementations of HO mainly dealt with equality constraints such as motion tracking [Siciliano and Slotine, 1991]. Later, *Kanoun et al.* presented a framework wherein the sub-tasks were solved as a cascade of quadratic programming (QP) problems [Kanoun et al., 2011] which included inequality constraints. This approach, however, was computationally inefficient for real-time control. A solution to this problem is to combine explicit inequality constraint handling with search in the null space of higher priority tasks [Herzog et al., 2016]. Based on these principles of HO, *Bellicoso et al.* proposed a control architecture for quadrupedal locomotion that exhibited behaviors which adapted to uneven and unknown ground allowing for perception-less traversal over challenging terrains [Bellicoso et al., 2016]. This work of *Bellicoso et al.* has been utilized in Chapter 4 (Paper II), Chapter 5 (Paper III) and Chapter 6 (Paper IV).

### Deep Reinforcement Learning

The problem of sequential decision making can be formulated in the framework of a discrete time Markov decision process (MDP) [Sutton et al., 1998]. An MDP is defined by the tuple - $\langle S, A, R, P, \mu \rangle$, where $S$ represents a set of states, $A$ a set of actions, $R : S \times A \times S \to \mathbb{R}$ the reward function, $P : S \times A \times S \to [0, 1]$ the state transition probability, and $\mu$ the initial state distribution. As shown in Figure 1.5, an RL agent

described in this MDP framework interacts with an environment based on a policy $\pi : S \to \mathcal{P}(A)$. This policy can be defined as a function mapping states to probability distributions over actions such that $\pi(a|s)$ denotes the probability of selecting action $a$ in state $s$. The objective of the RL agent is to then obtain a policy $\pi_\theta$ for parameters $\theta$ to maximize the expected cumulative discounted return,

$$J(\pi) \doteq \underset{\mathcal{T}\sim\pi_\theta}{\mathrm{E}}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t, s_{t+1})\right], \tag{1.14}$$

where $\gamma \in [0, 1)$ is the discount factor and $\mathcal{T} = (s_0, a_0, s_1, ...)$ denotes a trajectory dependent on $\pi_\theta$. In an episodic problem, an agent interacts with the environment for a finite time horizon, after which a terminal state is reached and the environment is reset. The value function $V_\pi(s)$ is defined as the expected return for following the policy $\pi$ from a state $s$. The action-value function $Q_\pi(s, a)$ is the expected return obtained by following $\pi$ from a state $s$ and taking an action $a$. The advantage function can then be defined as $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$. This describes the *advantage* of taking an action $a$ in state $s$ compared to following the policy $\pi$.

Value-based methods for deep RL aim to learn a value function which is then used to define a policy. The popular $Q$-learning algorithm [Watkins and Dayan, 1992] employs a lookup table of state-action values $Q(s, a)$ and utilizes a Bellman update rule to learn the optimal state-action value function $Q^*(s, a)$. The neural fitted Q-learning (NFQ) algorithm [Riedmiller, 2005] parameterizes the $Q$-values as a neural network where the parameters are updated by stochastic gradient descent. Extending upon this, *Mnih et al.* introduced the deep Q-network (DQN) algorithm which leveraged a replay memory buffer and a slower update rate for the target $Q$ network. DQN was able to demonstrate strong performance for playing Atari games [Mnih et al., 2013]. Stability improvements to DQN such as double DQN [Van Hasselt et al., 2016] and dueling network architecture [Wang et al., 2016] have also been proposed and implemented.

Policy-based methods use a learning signal derived from sampling a family of policy parameters and the direction of policy learning occurs towards the policies that achieve better returns [François-Lavet et al., 2018]. In addition to gradient ascent based

approaches, policy-based methods also include evolution strategies [Salimans et al., 2017]. The widely used deep deterministic policy gradient (DDPG) [Lillicrap et al., 2015] approach introduces a direct representation of a policy and moves it in the direction of the gradient of $Q$ function. Such policy gradient approaches typically require an estimate of the value function for the current policy. For this, actor-critic methods can be utilized [Konda and Tsitsiklis, 1999] where the actor represents the control policy whereas the critic represents the value function. Algorithms such as advantage actor-critic (A2C) and asynchronous advantage actor-critic (A3C) [Mnih et al., 2016] are employ the actor-critic architecture. Based on the natural policy gradient method [Kakade, 2001], policy optimization strategies based on trust regions aim at improving the policy whilst focusing on restricting significant changes in the current policy. Trust region policy optimization (TRPO) [Schulman et al., 2015] and proximal policy optimization (PPO) [Schulman et al., 2017b] algorithms are two widely used policy optimization strategies. Another approach to deep RL combines policy gradient methods with off-policy $Q$ learning [François-Lavet et al., 2018; Haarnoja et al., 2017; Schulman et al., 2017a; Haarnoja et al., 2018].

RL enables natural emergence of optimal behavior obtained by maximizing a reward feedback through interactions with the environment of operation [Arulkumaran et al., 2017]. The recent advances in deep RL have enabled achieving sophisticated control skills for solving a Rubik's cube [Akkaya et al., 2019], commanding tokamak coils to control nuclear fusion plasma [Degrave et al., 2022], and learning dexterous manipulation skills [Jeong et al., 2020]. In the context of legged locomotion, early works on RL investigated optimizing locomotion gaits [Kohl and Stone, 2004], improving dynamic stability [Tedrake et al., 2002], and even learning dynamic walking behavior [Schuitema et al., 2005]. One of the notable works is that of *Sims* who utilized gradient-free genetic algorithms to evolve creatures that swim, walk and run [Sims, 1994]. A similar work of *Geijtenbeek et al.* focused on optimizing muscle routing and control parameters for bipedal locomotion [Geijtenbeek et al., 2013]. *Cully et al.* further presented an adaptive approach that exhibits damage compensatory behavior without requiring self-diagnosis [Cully et al., 2015].

One of the major works which pushed forward the development of deep RL control policies for complex legged systems is that of *Hwangbo et al.*. The work [Hwangbo et al., 2019] leveraged an extremely fast simulation framework to train a locomotion policy for the ANYmal B quadruped. To address the simulation-reality gap, the authors proposed learning an actuator network to accurately model the actuation dynamics in the simulator while training the control policy. Additionally, the work employed domain randomization [Tobin et al., 2017] techniques to obtain a robust control policy that could be successfully transferred to the real system. The success of the method led to further research works which demonstrated impressive control behavior on different robots such as Laikago [Peng et al., 2020] and Cassie [Xie et al., 2020]. Additional research that leveraged the ability of recurrent networks to perform dynamics identification was also presented [Siekmann et al., 2020]. These works, however, overlooked the primary objective of legged systems, that is, mobility over uneven terrain. *Peng et al.* demonstrated a control policy for an animated biped to traverse randomly placed obstacles [Peng et al., 2017]. *Tsounis et al.* proposed a hierarchical terrain-aware gait planning and control architecture based on deep RL for navigation to a desired goal position [Tsounis et al., 2020]. The developed controller, however, was not demonstrated on a physical platform. *Lee et al.* showed locomotion over challenging terrains [Lee et al., 2020], however, the work only focused on use of proprioceptive feedback and was thus not perceptive. *Rudin et al.* introduced a training approach for obtaining terrain-aware robotic locomotion policies by leveraging massively parallel learning framework [Rudin et al., 2021]. Recently, *Miki et al.* demonstrated impressive robust and perceptive locomotion skills in unstructured environments [Miki et al., 2022].

Although much of legged locomotion has been approached with model-free deep RL techniques which are typically sample inefficient, an alternate line of work has focused on investigating model-based methods which enable an RL agent to predict state transitions and rewards. This allows the agent to plan ahead and learn a policy that can explicitly make decisions conditioned on the desired task [Sekar et al., 2020]. In the direction of model-based RL, *Nagabandi et al.* presented MBMF [Nagabandi et al., 2018], an approach based on model predictive control (MPC) which leverages learned

dynamics representations to achieve range of locomotion tasks. Other works, such as *ExIt* [Anthony et al., 2017] and *AlphaZero* [Silver et al., 2018], utilize an expert planner to iteratively improve the performance of a learned policy. In contrast, *world models* learn the spatial and temporal representations of the environment and use the extracted features to train RL agents [Ha and Schmidhuber, 2018]. To transfer policies across different morphologies, *Huang et al.* proposed the use of graph neural networks (GNNs) [Huang et al., 2020] to obatin a global control policy that generalizes to variations in dimensionality of state and action spaces. *Kurin et al.* additionally introduced AMORPHEUS that replaces GNNs with transformers for multi-task RL. To obtain a policy that performs well across a large set of environments *Dennis et al.* proposed an adaptive unsupervised environment design approach [Dennis et al., 2020] to generate structured and solvable environments. In the direction of stabilizing deep RL by addressing the sensitivity of popular algorithms to hyperparameters such as learning rate and entropy cost, and reducing reward collapse, population based training approach has also been investigated [Jaderberg et al., 2017]. Such a population-based training method even offers improved performance in complex multi-agent reinforcement learning environments such as StarCraft II [Vinyals et al., 2019]. While these approaches haven't been thoroughly investigated for complex legged robotic locomotion, the above-mentioned research, nonetheless, provides a reference for developing modular system adaptive control strategies.

## 1.3 Thesis Organization

This introduction is followed by Chapter 2 that provides a brief summary of the motivations and contributions of each of the manuscripts included in this dissertation. Chapter 2 further presents relevant observations supporting the research direction. The research articles are cumulatively provided in Chapters 3 through 7. Chapter 8 concludes this thesis and discusses the achievements and outlook relating to the presented research.

# 2
# Contributions

This chapter introduces the research manuscripts included in this thesis, providing an overview on the research motivation and contribution of each of the papers.

## 2.1 Paper I

*Guided Constrained Policy Optimization for Dynamic Quadrupedal Robot Locomotion*, Siddhant Gangapurwala, Alexander Mitchell and Ioannis Havoutis, IEEE Robotics and Automation Letters (RA-L), vol. 5, no. 2, pp. 3642-3649, April 2020.

This work presents a deep RL training approach for faster convergence and safety critical operations, and evaluates the performance of the trained RL locomotion policy on a simulated and real quadruped.

### Motivation

Deep reinforcement learning (DRL) has emerged as a promising approach for developing efficient, dynamic and robust control solutions. In contrast to previously proposed model-based techniques for legged robot locomotion, DRL allows for emergence of

natural behaviors that are capable of fully utilizing system potential [Hwangbo, 2018; Arulkumaran et al., 2017]. Despite their success on real robots, the inability to perform a formal behavioral analysis of policies trained using DRL limits their applicability to constrained safety-critical tasks. Additionally, DRL algorithms require exhaustive reward and hyper-parameter tuning, and suffer from poor sample efficiency. This paper, therefore, investigates an approach to obtain policies trained using DRL that exhibit behaviors within constrained regions and further presents a strategy for faster learning convergence close to a desired optimum.

## Contribution

This work introduces a guided constrained policy optimization (GCPO) approach for training a quadrupedal locomotion control policy parameterized as a deep neural network. The presented constrained proximal policy optimization (CPPO) algorithm extends the widely-used proximal policy optimization (PPO) algorithm. CPPO is utilized for training DRL control policies that exhibit control behaviors within desired regions of operation. These constraints include limits on joint velocity, joint acceleration, feet positions and ZMP. In order to accelerate learning, GCPO employs a guided policy update (GPU) method, motivated by behavior cloning and guided policy search, to direct policy convergence towards a desired optimum eliminating the need for precise reward and hyperparameter tuning. GPU utilizes reference oscillatory joint motions to guide the control policy behavior. Training a locomotion policy using GCPO requires less than 7% of training samples required for training with PPO. Additionally, GCPO is shown to offer near-perfect guarantees for constrained operations. Furthermore, this paper demonstrates robust and dynamic control behavior observed on a real ANYmal B quadruped, requiring, on average, less than 20 N m torque compared to a baseline model-based controller while being able to handle external perturbations of up to 100 N.

## Observations

In addition to the results presented in the manuscript, the following observations were critical to the research presented in this thesis:

1. Domain randomization, particularly, dynamics randomization, was extremely important for sim-to-real transfer. While learning the actuator network [Hwangbo et al., 2019] aided in reducing the disparities between the simulated and real robot, transferring the policy to the physical robot without dynamics randomization resulted in aggressive low-level joint position tracking behavior. This was largely due to imperfect modeling of the actuation dynamics as a result of temporally constrained and small network architecture. An analysis on this has been provided in Chapter 5 (Section IV-F) and Chapter 6 (Appendix-F).

2. Dynamics randomization was ineffective without introduction of the history of joint states, $s_j \in \mathbb{R}^{84}$, implying, without state history, the agent cannot identify system dynamics. Without $s_j$, the RL agent failed to learn a stable locomotion policy. Here,

$$s_j := [\Delta \mathrm{q}_{j_{t-8}}^T \quad \Delta \mathrm{q}_{j_{t-4}}^T \quad \mathrm{q}_{j_t}^T \quad \dot{\mathrm{q}}_{j_{t-8}}^T \quad \dot{\mathrm{q}}_{j_{t-4}}^T \quad \dot{\mathrm{q}}_{j_t}^T \quad {\mathrm{q}_{j_{t-1}}^*}^T]^T \tag{2.1}$$

where $\mathrm{q}_{j_t} \in \mathbb{R}^{12}$ represents the joint positions at control step $t$, $\dot{\mathrm{q}}_{j_t} \in \mathbb{R}^{12}$ represents the joint velocities, $\mathrm{q}_{j_{t-1}}^* \in \mathbb{R}^{12}$ is the vector of previous actions describing the desired joint positions, and $\Delta \mathrm{q}_{j_t} = \mathrm{q}_{j_t}^* - \mathrm{q}_{j_t}$. A recurrent policy is also expected to offer ability to implicitly model system dynamics [Siekmann et al., 2020]. However, recurrent neural networks (RNNs) require sequential forward passes to be trained using back-propagation through time (BPTT) [Mozer, 1995] which is significantly slower than training an MLP. In our preliminary experiments we observed that RNNs required approximately 4 times the training duration than MLPs. In this regard, the presented work focused on utilizing a history of quantities in the observation space as opposed to RNNs.

3. $s_j$ served two main purposes: (1) to support encoding of temporally-dependent series elastic actuation dynamics [Gehring et al., 2016], and (2) detection of feet contact events that predominantly describe the locomotion behavior.

4. Introduction of $s_j$ in the state space was sufficient to adapt to changes in mass. Introducing a history of base velocities, in addition to $s_j$, to address the variations in base properties did not offer improved performance.

5. The combination of dynamics randomization and setup of state space which allowed for implicit modeling of dynamics properties enabled transfer of the locomotion policy to the physical robot. The robot was then also deployed for outdoor and loco-manipulation use as shown in Figure 2.1.

6. The adaptive property of the control policy also led to emergence of a staggering gait when the actuation tracking gains and torque were deliberately limited on the knee joint of the left-front foot (LF_KFE) in order to emulate a weak actuator. The walking style is shown in Figure 2.2.

7. The normalized saliency maps provided in Figure 2.3 compare the dependency of the desired joint position of LF_KFE, $q_2^*$, as generated by the control policy, on the joint states $s_j$. For the case of normal actuation, the control policy output corresponding to LF_KFE is largely conditioned on LF_KFE's velocity and joint tracking characteristics. In contrast, during weak LF_KFE actuation, the dependency of the control policy output, $q_2^*$, on the overall joint states of the system significantly increases.

8. The locomotion policy trained without domain randomization fails in the weak actuation scenario. The robot is unable to track reference velocity commands, eventually falling on the side. The saliency map unrolled for 1 s, in this case, resembles the one obtained for normal actuation.

In conclusion, a control policy capable of encoding system dynamics is able to exhibit adaptive behavior. This is fostered with an appropriate setup of the state space



**Figure 2.1:** Locomotion policy trained using GCPO deployed on the real ANYmal B quadruped outdoors and with added mass.

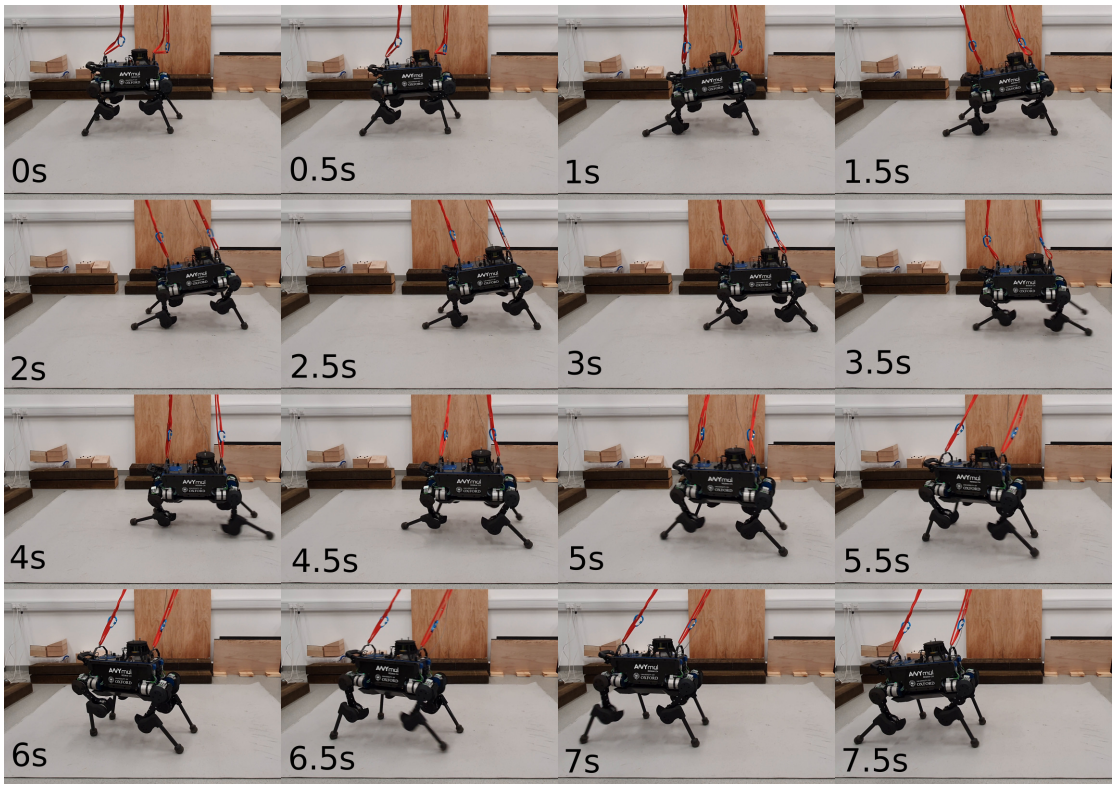



**Figure 2.2:** Emergence of behavior observed by emulating a weak actuator at the front left knee.

and utilization of techniques such as domain randomization. It is important to note that a control policy, parameterized as a neural network, can lead to emergence of behavior within a constrained range of states. The inability of a neural network to extrapolate implies that a deep RL policy is only reliable within the state distributions experienced during training.

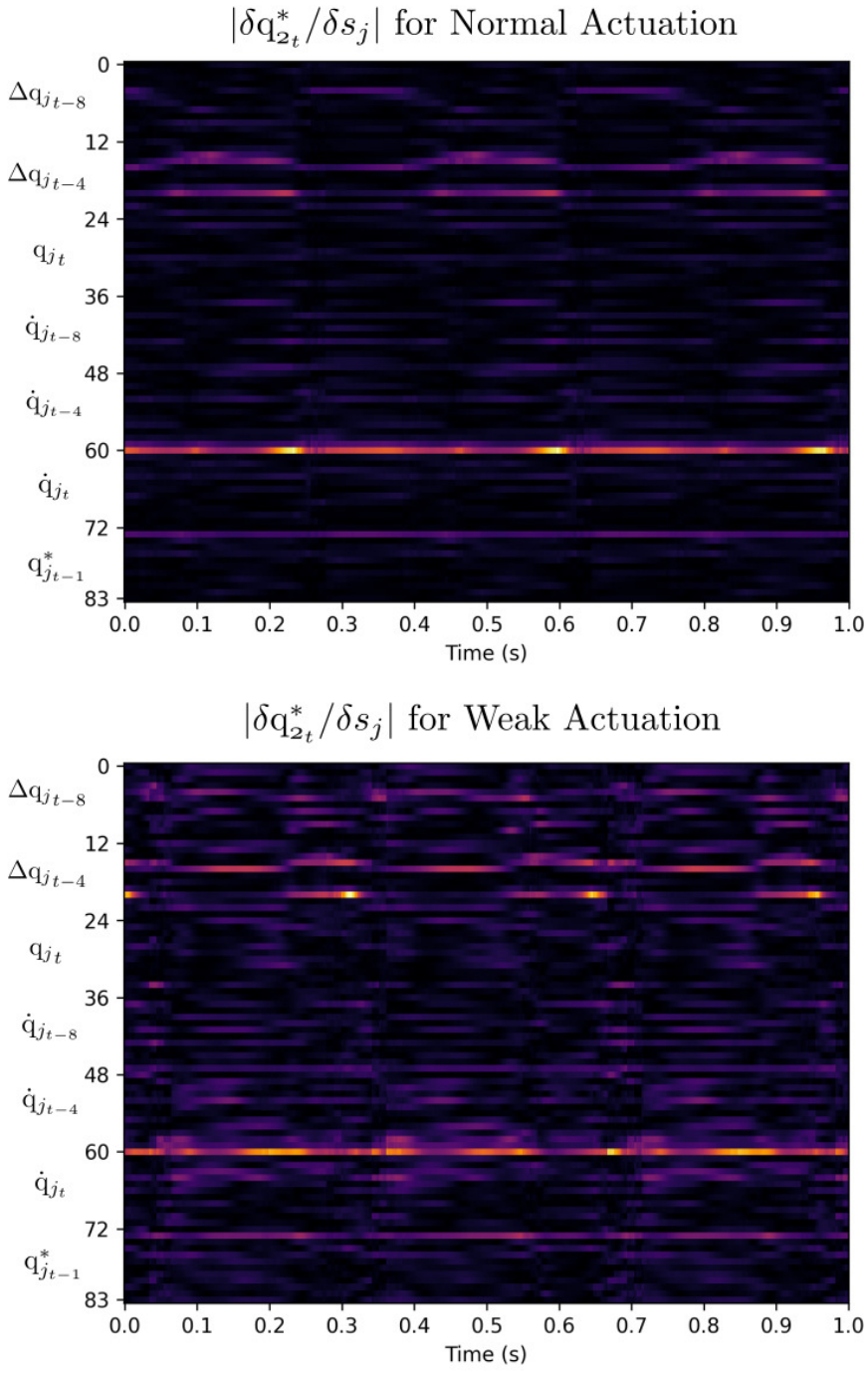



**Figure 2.3:** Saliency maps unrolled across history of join states for 1 s.

## Interrelations

This work focused on one of the main objectives of this thesis which is to obtain a controller that exploits the system's potential to achieve a high degree of efficiency, agility,

robustness and reactivity. The observations made during this research contributed towards personal understanding of setup of the RL training environment and state space that allows for emergence of adaptive behavior. These observations were further useful for development of the proceeding works (Chapter 5 and Chapter 6) which also include the state history in the observation space to enable the RL agents to identify system dynamics. For reward function design and safety constraints, the manuscript also details use of model-based strategies for stability estimation which is further considered in the attached papers. Additionally, this research explored an approach for faster convergence. This is relevant for training a control policy which would have otherwise required significantly larger training time as is presented in Paper IV.

## 2.2 Paper II

*Rapid Stability Margin Estimation for Contact-Rich Locomotion*, Romeo Orsolino, Siddhant Gangapurwala, Oliwier Melon, Mathieu Geisert, Ioannis Havoutis and Maurice Fallon, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2021.

This work presents an approach for fast estimation of quadrupedal robot stability offering a three order of magnitude increase in computational performance over the corresponding analytical method. This makes it suitable for training deep RL locomotion policies and performing real-time trajectory optimization.

### Motivation

Dynamic stability is an important criterion for development of motion planners and controllers. The existing quadrupedal stability estimation approaches that are capable of real-time execution are limited to flat ground or quasi-static locomotion. For traversing uneven terrain, it is necessary to perform an *efficient* evaluation of the robot's dynamic stability which would otherwise result in significant computational overhead for footstep planning and long-horizon trajectory generation. This paper

presents that approximating the dynamic stability prediction process as a function mapping robot state and terrain information to a stability metric allows for extremely fast estimation of robot's stability with an acceptable loss in accuracy.

## Contribution

This work introduces a deep neural network parameterization, trained using supervised learning, for estimating quadrupedal *stability margin*. The presented approximation results in a speed-up of up to three orders of magnitude. To demonstrate satisfactory accuracy, the paper also presents use of the learned stability margin estimator for locomotion over uneven terrain using a trajectory optimization approach expressed in the framework of receding-horizon planning and control. Additionally, the work investigates that utilizing the stability margin estimation as part of the RL reward function results in more stable locomotion behavior whilst providing an early signal for value estimation as opposed to obtained using episode termination criteria.

## Observations

Introduction of the stability margin reward term in the RL formulation resulted in faster training convergence. The obtained control policy exhibited smoother and less-aggressive motions compared to locomotion policies trained without the stability reward. These observations are detailed in the manuscript.

## Interrelations

This research investigated techniques to encourage dynamically stable control behavior of an RL policy especially for locomotion over uneven terrain. The work further presented an approach to perform rapid estimation of the stability margin. Without this approximation, each episode rollout step would have required more than 5 ms compared to less than 50 µs for the learned margin estimator (requiring approximately

5 µs for network forward pass). These observations and developments were employed in Paper III and Paper IV to obtain planning and control RL policies.

## 2.3 Paper III

*Real-Time Trajectory Adaptation for Quadrupedal Locomotion using Deep Reinforcement Learning*, Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon and Ioannis Havoutis, IEEE International Conference on Robotics and Automation (ICRA) 2021.

This work introduces a control architecture to perform online adaptation of offline generated motion plans in order to address inaccurate whole-body motion tracking and react to unexpected perturbations.

### Motivation

Approaches based on trajectory optimization have been demonstrated to generate impressive terrain aware long horizon motion plans that exhibit dynamic locomotion behavior. However, to address the computationally exhaustive task of trajectory optimization, the long horizon motion plans are often generated offline. This necessitates precise whole-body tracking of the reference trajectories for online execution. In case of unexpected perturbations and inaccurate tracking, however, accumulation of tracking errors may result in aggressive control corrections eventually causing failures. This makes such an approach unreliable for outdoor deployment.

### Contribution

This paper introduces a real-time trajectory adaptation module, trained using RL, to account for inaccurate motion plan tracking and external disturbances by introducing corrective deviations in the reference trajectory online. The proposed control architecture thus allows for corrective feedback-based tracking. The motion plan adapted

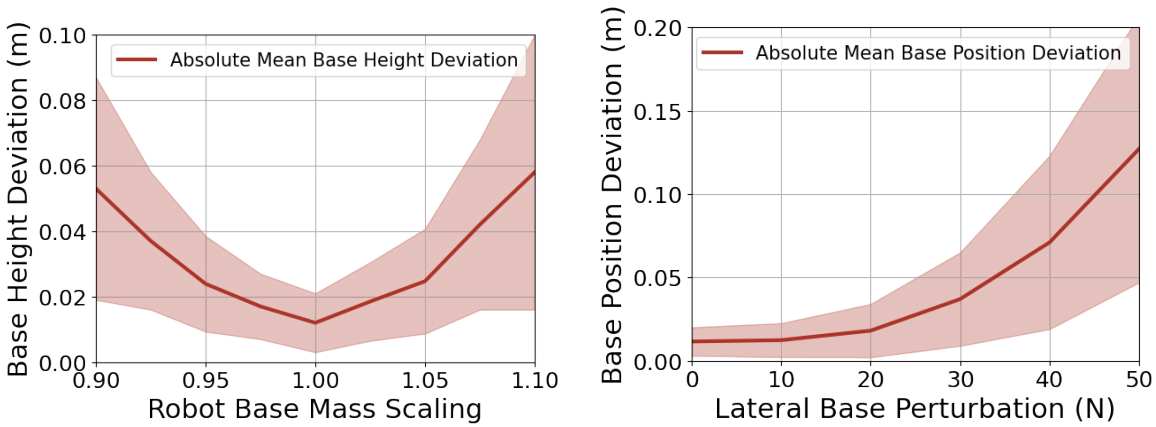



**Figure 2.4:** Output of the trajectory adaptation RL policy representing the desired deviations in the base position along the $z$-axis (left) and the $y$-axis (right) to account for variations in base mass and external perturbations applied laterally on the robot's base. The error bands represent the standard deviation of the quantities.

by the RL policy, parameterized as a deep neural network, is tracked using a model-based whole-body controller which generates the desired low-level actuation commands. This research shows that online trajectory adaptation results in significant increase in motion plan tracking performance on both flat ground and complex uneven terrains. Furthermore, performing a direct mapping of the tracking error history to the deviations in reference trajectory by an RL neural network policy enables real-time adaptation which would otherwise not have been possible using a trajectory optimization approach. The proposed control architecture is also able to deal with changes in system properties such as the robot base mass and actuation dynamics.

## Observations

Based on observations made in Paper I, the trajectory adaptation policy was trained using dynamics randomization and a temporal state representation to implicitly model system and tracking dynamics. In this regard, the adaptation policy was able to address variations in system properties. Figure 2.4 represents the mean and standard deviation of the desired base position deviations generated for tracking a 2 m long trajectory in 4 s for varying mass and external perturbations. Additional observations and results are detailed in the manuscript.

### Interrelations

The adaptive behavior of the RL policy is largely based on the tracking dynamics encoded using a trajectory tracking error history. Paper IV utilizes a similar state space to make the trained RL policies robust to variations in system dynamics. For terrain-aware locomotion, the paper presents an approach for terrain feature extraction using a convolutional encoding network. In addition, to improve generalizability, the work employs a large set of procedurally generated terrains during training. The developed terrain generation tool and the terrain feature extractor were also utilized in Paper IV.

## 2.4 Paper IV

*RLOC: Terrain-Aware Legged Locomotion using Reinforcement Learning and Optimal Control*, Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon and Ioannis Havoutis, under review.

This work introduces a modular reinforcement learning and optimal control based quadrupedal locomotion framework for traversal over uneven terrain. The proposed control architecture is further demonstrated to perform zero-shot transfer to a domain, described by another quadrupedal platform, not introduced during training.

### Motivation

Legged robotics research has been driven by the motivation to develop control frameworks for dynamic locomotion over complex unstructured terrains. Solutions to this problem need to address the dynamic stability of the robot, select appropriate feet contact sequences, and actively balance during locomotion. Prior approaches have relied on exhaustive combinatorial search of desired footholds to traverse uneven terrain. The overhead associated with such a method and the limited on-board computational resources restrict the control behavior to quasi-static motions. As a solution, this

research presents a unified reinforcement learning and optimal control based approach to perform dynamic locomotion over uneven terrain.

## Contribution

The contributions of this paper are threefold. First, the paper presents a training approach to obtain a perceptive footstep planning RL policy which adapts to and exploits the domain largely described by the motion controller employed to track the generated footsteps. The footstep planning policy, parameterized as a deep neural network, rids us of the need to perform an exhaustive search of optimal footholds. Second, the research introduces a domain adaptive tracking RL policy to address system modeling inaccuracies during whole-body motion tracking. This enables transfer of a motion controller to a robot where the modeled characteristics differ from the actual system dynamics demonstrating an adaptive control behavior. This adaptive tracking is also utilized to train a footstep planning policy which can be used on systems with varying kinematic structure and dynamics properties. Third, the work realizes a modular perceptive framework combining RL and optimal control based approaches to perform zero-shot transfer to a domain not explicitly modeled during training. The paper demonstrates that the policies trained in simulation using ANYmal B, can be transferred to the physical ANYmal C quadruped.

## Observations

The results and observations relating to this work have been extensively covered in the manuscript. In addition, the developed framework was also transferred to ANYmal C outdoors as shown in Figure 2.5.

## Interrelations

This work heavily develops upon the observations and software frameworks introduced in each of the previous papers, and demonstrates a system adaptive terrain-aware dynamic



**Figure 2.5:** The developed RLOC framework deployed on ANYmal C for outdoor locomotion over uneven terrains.

locomotion framework that can transfer to platforms with different kinematic and dynamic characteristics. The footstep planning and domain adaptive tracking policies introduced in this work allow for thorough utilization of system capabilities while the safety-critical constraints are mainly addressed by the use of a model-based motion controller. In addition, the emergency recovery control RL policy, based on the work presented in Paper I, was also demonstrated to transfer to a different robot than the one used for training. This research contributed to development of a framework which thoroughly utilized system capabilities while allowing for emergence of an adaptive control behavior as demonstrated by zero-shot transfer from ANYmal B to ANYmal C.

## 2.5 Paper V

*Learning Dynamics Conditioned Platform Adaptive Quadrupedal Locomotion Policy*, Siddhant Gangapurwala, Suyoung Choi, Mathieu Geisert, Jemin Hwangbo and Ioannis Havoutis, unpublished.

This work introduces a deep RL training method to obtain a locomotion policy that

can transfer to wide domains of quadrupedal platforms without requiring retraining, and further demonstrates its performance on different real robots.

## Motivation

Majority of prior proposed control research has focused on development of solutions for a target system such that the control parameters are tuned to deliver optimal performance for the system considered during development. Transferring the control solution to a different platform requires additional tuning, without which the control performance significantly deteriorates.

## Contribution

To circumvent this cumbersome task of redevelopment of the controller for a new design or a different platform, this work investigates an ambitious research goal of obtaining a universal quadrupedal locomotion policy which can transfer to different quadrupeds without requiring retraining. The paper presents an approach to obtain the latent representation of system dynamics using a recurrent encoding network. This latent representation is used to condition the behavior of the RL locomotion policy in order to obtain a platform adaptive control solution. This control policy is demonstrated on several simulated quadrupedal systems and also on different real robots.

## Observations

The observations made in this paper are consistent with those of previous works. The system adaptive behavior of the locomotion policy emerges due to two main factors: (1) training over a large set of varying domains, and (2) introducing a dynamics representation to condition the control policy output. The resulting policy can be transferred to different quadrupeds without requiring retraining. A detailed analysis of the observations and results is provided in the manuscript.

## Interrelations

This work focuses on the main objective of this thesis: obtaining a system adaptive control policy which thoroughly utilizes system potential. While the previous works employed a design to promote implicit modeling of the system dynamics for emergence of adaptive behavior, this work built upon the skills and experience obtained throughout the prior research to present an approach that explicitly introduced inference of the system dynamics as an objective to obtain a dynamics conditioned locomotion controller.

# 3

# Paper I
# Guided Constrained Policy Optimization for Dynamic Quadrupedal Robot Locomotion

**This work was published in IEEE Robotics and Automation Letters (RA-L) 2020 and was presented at the IEEE International Conference on Robotics and Automation (ICRA) 2020.**

# Manuscript Preface

This work introduces a constrained deep RL training approach to guide quadrupedal locomotion policy convergence towards a desired optimum. The resulting control policy demonstrates robust, efficient, reactive, and dynamic control behavior. The RL setup utilized in this work employs 3 degrees of constraints. These comprise *Soft* ($\rho$), *Hard* ($\kappa$), and *No-go* ($\eta$) constraints and have been detailed in Section II of the attached manuscript. $\rho$-constraints are directly evaluated through the reward feedback. $\kappa$-constraints are introduced in the set of cost functions described in the framework of a constrained Markov Decision Process [Altman, 1999]. $\eta$-constraints prevent exploration around regions which do not contribute towards learning a desirable control behavior. These become part of the early-termination criteria. When any of the $\eta$-constraints are violated, the training episode is terminated and the environment is reset.

This work also compares the training performance of 3 widely used physics simulation frameworks: PyBullet [Coumans and Bai, 2016–2021], MuJoCo [Todorov et al., 2012] and RaiSim [Hwangbo et al., 2018]. In this regard, Section II-B provides the training times involving execution of 10M simulation steps across the different simulators for the task of moving forward with maximum feasible base velocity. While RaiSim allows for faster training in comparison to the other simulators, another important observation is discussed in the manuscript. This relates to emergence of significantly different locomotion behavior due to differently modeled actuation dynamics in each of the investigated simulators. Figure 2 of the attached paper presents the episodic returns obtained while training a control policy for the above-mentioned task. This is evaluated across: (1) different simulators, and (2) different actuation tracking gains in RaiSim. The difference in the learned locomotion styles is also attributed to the very high-level reward function which does not include reward terms that favor a particular locomotion style. In this regard, the RL environments with aggressive actuation (high position tracking gains) and low damping result in convergence towards unstable control behaviors. In contrast, higher damping results in smoother locomotion. These results have been detailed in Section II-B.

The constraints introduced in the locomotion problem include limits on joint velocities and accelerations, feet positions relative to the base, number of feet in contact with the ground, and stability margin as introduced in Section 1.2.2 of Chapter 1. These constraints ensure that the resulting locomotion behavior is smooth, non-aggressive, and dynamically stable. Additional motivation for the constraints on joint velocity and acceleration is based on the ability to perform successful sim-to-real transfer. This is relevant since the actuators on physical systems have finite peak velocity and acceleration. These limits are often not considered in simulations which assume ideal actuation thereby affecting the transferability of the learned control solution on to the real system.

This work presents a behavior cloning inspired guided policy update approach to direct policy convergence towards a desired control behavior. While such an approach often fails due to compounding errors and covariate shift [Ross and Bagnell, 2010; Xie et al., 2020], in this work, guided policy update is utilized in conjunction with exploratory policy optimization. In this regard, guided policy update is only used to infer an underlying oscillatory behavior of a reference controller whereas the data-driven policy optimization is used to stabilize and fine-tune the overall control behavior. Alternative imitation learning approaches, such as DAGGER [Ross et al., 2011] and DART [Laskey et al., 2017] address the compounding error issue associated with behavior cloning through interactive querying of the reference control policy. Such approaches can be employed as substitutes for guided policy update and are expected to offer improved imitation behavior. However, since the work also focuses on learning dynamic control behavior which is robust to variations in system dynamics, exploratory policy optimization still remains the most important component of the proposed training framework.

# Guided Constrained Policy Optimization for Dynamic Quadrupedal Robot Locomotion

Siddhant Gangapurwala, Alexander Mitchell and Ioannis Havoutis

***Abstract*—Deep reinforcement learning (RL) uses model-free techniques to optimize task-specific control policies. Despite having emerged as a promising approach for complex problems, RL is still hard to use reliably for real-world applications. Apart from challenges such as precise reward function tuning, inaccurate sensing and actuation, and non-deterministic response, existing RL methods do not guarantee behavior within required safety constraints that are crucial for real robot scenarios. In this regard, we introduce guided constrained policy optimization (GCPO), an RL framework based upon our implementation of constrained proximal policy optimization (CPPO) for tracking base velocity commands while following the defined constraints. We introduce schemes which encourage state recovery into constrained regions in case of constraint violations. We present experimental results of our training method and test it on the real ANYmal quadruped robot. We compare our approach against the unconstrained RL method and show that guided constrained RL offers faster convergence close to the desired optimum resulting in an optimal, yet physically feasible, robotic control behavior without the need for precise reward function tuning.**

***Index Terms*—Deep Learning in Robotics and Automation, AI-Based Methods, Legged Robots, Robust/Adaptive Control of Robotic Systems, Underactuated Robots**

## I. Introduction

LEGGED locomotion has been an active area of robotics research over the past few decades. Despite our best efforts, achieving extraordinarily dynamic robotic behavior still remains an open problem. Most of the existing work has focused on the use of traditional model-based control techniques, such as offline trajectory optimization (TO) [1] and online model predictive control (MPC) [2] which, due to their mathematical complexity, are often based on simplified models of the systems. Such simplifications result in control solutions that are often mechanically limiting and inefficient.

Considering robotic locomotion as a reinforcement learning (RL) problem [3] offers a model-free data-driven alternative to model-based control. Although RL has witnessed significant contributions from researchers to address issues such as sample inefficiency [4] and hyperparameter tuning [5], it still faces significant challenges to be used for real-world robotic locomotion applications mainly due to no hard guarantees on safety-critical constraints.

Manuscript received: September, 10, 2019; Revised December, 17, 2019; Accepted January, 19, 2020.

This paper was recommended for publication by Editor Nikos Tsagarakis upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the UKRI/EPSRC RAIN and ORCA Hubs [EP/R026084/1, EP/R026173/1], Robust Legged Locomotion [EP/S002383/1] and the EU H2020 Projects MEMMO and THING. It was conducted as part of ANYmal Research, a community to advance legged robotics.

The authors are with the Dynamic Robots Systems Group, Oxford Robotics Institute, University of Oxford, UK. Email: {siddhant,ioannis}@robots.ox.ac.uk.

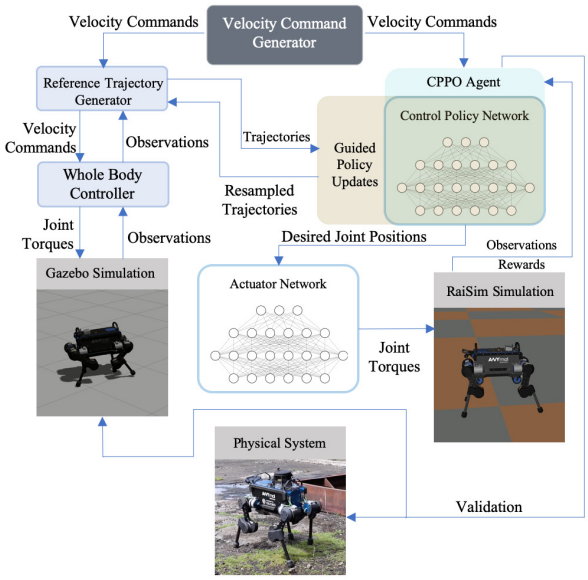



Fig. 1: Overview of our training and validation process. Accompanying video can be found at: https://youtu.be/iPDmG9knkLs

In this work, we develop an RL problem formulation that introduces constraints based on optimal control techniques. We train a control policy, using our constrained proximal policy optimization (CPPO) method based upon proximal policy optimization (PPO) [5], for tracking user-generated reference base velocities on the ANYmal [6] quadruped, a 33 kg legged robot. We experimentally validate its performance in comparison with unconstrained training procedures in a physically realistic simulation environment and on the real ANYmal robot.

### A. Related Work

Control architectures for robotic quadrupedal locomotion have seen various forms. One of the common approaches leverages mathematical optimization techniques [7] to generate reference trajectories by solving an optimal control (OC) [8] problem with objectives such as minimization of energy consumption, and constraints that consider the dynamics of the robotic systems. Authors of [9] presented a TO formulation for legged locomotion that automatically generates reference motions without requiring any prior footstep planning.

Extending upon OC, some of the work has focused on formulating locomotion as multiple tasks [10], such as maintaining robot stability and tracking desired limb motions, solved by prioritizing each individual task using quadratic programming [11] solvers. This principle of sub-dividing tasks

into simpler problems has also been followed in some of the deep RL research [12].

Model-based control methods often use simplified mechanical models to ease on the mathematical complexity of the problem. For example, most formulations consider the robotic system as a point mass with massless limbs. These approximations result in control solutions that cannot exploit the full range of capabilities of the systems. Moreover, several model-based controllers require hand-tuned costs by human-experts which are specific to each system's task thereby limiting their generality.

Deep RL methods attempt to address some of these limitations of model-based control by employing model-free techniques which optimize over a control policy, a neural network which maps states into actions, so as to maximize a task-specific reward signal by means of trial and error. Such methods have been investigated for legged locomotion tasks [13], [14] but have mainly been demonstrated in simulations using unrealistic robot models, e.g. ideal torque sources, infinite velocity/torque ranges. Moreover, RL techniques usually require large amounts of data making training on a real robot infeasible. This necessitates the use of physics simulators. However, policies trained using simulations often do not transfer for real world tasks since the reality gap between simulations and the physical world are strongly pertinent. These issues of RL have been tackled using techniques such as actuator modeling [15], implementing a modular training approach [16], introducing domain randomization [17], increasing policy generalization [18] and adding noise to observations and actions in the training environments. Authors of [15] have demonstrated the use of a deep RL approach to complex legged locomotion tasks. Our work, extends upon their training methods to a constrained learning approach as discussed in section III.

Despite the success of deep RL approaches on real-world robotic applications, one of the main challenges in solving an RL problem is precise reward function tuning. As a solution, an inverse reinforcement learning (IRL) [19] problem can be characterized as: given reference trajectories of an agent in various circumstances, determine the reward function to be minimized. This reward function is then used to solve an RL problem. The authors of [20] and [21] have successfully implemented IRL methods for perception and control tasks, however, the need for the extra step of solving an RL problem adds to training delays. Instead, designing the problem as that of behavioral cloning (BC) [22] gets rid of the reward recovery step, and directly optimizes over a policy given reference demonstrations. Along similar lines, guided policy search (GPS) [23] techniques can be used to reduce training times by directing policy learning in turn avoiding poor local optima.

Model-based RL [24] techniques, which require a knowledge of system dynamics, have also been proposed as an approach to boost convergence along desired optima. In the pursuit of making RL methods desirable for use in safety critical systems, methods such as constrained policy optimization (CPO) [25] have also been investigated to ensure that an RL control policy obeys the necessary safety constraints during operation.

### B. Contributions

Our work extends upon the above research to realize an RL problem formulation that considers the constraints required to guarantee the stability of a quadrupedal robot system. Furthermore, our problem formulation introduces constraints such as end-effector boundaries, joint velocity limits, and joint acceleration limits that direct policy optimization towards a desired quadrupedal locomotion behavior. This constrained formulation, coupled with techniques motivated by BC and GPS, which in our case is guided policy updates (GPUs), further results in the reduction of training time while also eliminating the need for precise reward function tuning.

We also present the importance of the setup of an RL environment, and show how differences in dynamic properties can result in a significantly different learnt behavior. We also compare different physics simulation frameworks and detail upon the motivations of preferring one over the others.

We introduce several schemes that make the quadrupedal system more robust and therefore better suited for use in real-world applications. Since we do not use approximations required for model-based control, our learnt policies better utilize system dynamics to generate efficient locomotion behavior requiring significantly lesser torque compared to a model-based trot controller. We successfully transfer the control policy trained in a simulator to the real system and further provide evidence of dynamic behavior of the control policy by testing its response after changing the physical properties of the real system, and also by continuously varying control step times.

## II. Approach

In this section we describe the RL methods we use for the task of quadrupedal locomotion.

### A. Algorithm

Based on the framework of Markov decision processes (MDPs) [26], a constrained Markov decision process (CMDP) [27] is defined as a tuple $(S,A,R,C,P,d,\mu)$, where $S$ is the set of states, $A$ is the set of actions, $R : S \times A \times A \to \mathbb{R}$ is the reward function, $P : S \times A \times S \to [0,1]$ is the state transition probablity, and $\mu$ is the starting state distribution as chracterized in the MDP tuple. CMDPs augment the MDP with a set $C$ of cost functions, $C_1,...,C_m$, with $C_i : S \times A \times S \to \mathbb{R}$, and limits $d_1,\ldots,d_m$ as described in [25]. Being consistent with the definitions and notation used by the authors of [25], a stationary policy $\pi : S \to \mathcal{P}(A)$ is defined as a function mapping states to probability distributions over actions. The set of stationary policies is defined as $\Pi$. $\pi(a|s)$ denotes the probability of selecting action $a$ in state $s$.

Given a performance measure,

$$J(\pi) \doteq \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1})\right],$$

where $\gamma \in [0,1)$ is the discount factor, $\tau$ denotes a trajectory dependent on $\pi$, we aim to select a policy $\pi$ which maximizes $J(\pi)$. For a CMDP, the expected discount cost return $J_{C_i}(\pi) \doteq \mathop{\mathrm{E}}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t C_i(s_t,a_t,s_{t+1})]$ for a policy $\pi$ with

cost function $C_i$. The set of feasible stationary policies $\Pi_C \doteq \{\pi \in \Pi : \forall_i, J_{C_i}(\pi) \leq d_i\}$.

The RL problem is then expressed as

$$\pi^* = \arg\max_{\pi \in \Pi_C} J(\pi).$$

For a policy $\pi_\theta$ parameterized with $\theta$, most policy optimization strategies iteratively update the base policy using local policy search methods [28] by maximizing $J(\pi)$ over a trust region [29]. For a CMDP, policy iteration using trust regions [25] can be expressed as

$$\begin{aligned} \pi_{k+1} &= \arg\max_{\pi \in \Pi_\theta} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}} \left[A^{\pi_k}(s,a)\right] \\ \text{subject to } & J_{C_i}(\pi_k) + \frac{1}{1-\gamma} \left( \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}} \left[A_{C_i}^{\pi_k}(s,a)\right] \right) \leq d_i \quad \forall_i \\ & \bar{D}_{KL}(\pi || \pi_k) \leq \delta. \end{aligned} \tag{1}$$

where $\bar{D}_{KL} = \mathop{\mathrm{E}}_{s\sim\pi_k} [D_{KL}(\pi||\pi_k)[s]]$, and $\delta > 0$ is a step size. $D_{KL}$ refers to the Kullback-Leibler divergence. Authors of [25] show that developing CPO as a trust region method implies CPO inherits the performance guarantee given by certain lower bound detailed in [25]. The authors of [25] also define the worst-case upper bound on the cumulative discounted return for a CMDP.

The PPO technique introduces a clipped objective function

$$L_t^{CLIP}(\theta) = \mathrm{E}_t \left[ \min \left( r_t(\theta) A_t, \mathrm{clip}\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right) A_t \right) \right]$$

where $\varepsilon$ is a hyperparameter. In our implementation, we introduce an approximation of the constraint expressed in (1) to the above objective, and rewrite it as

$$L_t^{CCLIP}(\theta) = L_t^{CLIP}(\theta) - \sum_i \zeta_i J_{C_i,t}(\pi_\theta), \tag{2}$$

where $\zeta_i$ is an experimentally tuned hyperparameter. The objective $L_t^{CLIP}$ is often augmented to include a value function loss term $L_t^{VF}$ and an entropy term $S$ [5]. The objective function, with coefficients $c_1$ and $c_2$ is then

$$\begin{aligned} & L_t^{CCLIP+VF+S}(\theta) \\ & = \mathrm{E}_t \left[ L_t^{CCLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right]. \end{aligned} \tag{3}$$

We introduced this approximation as, in our experiments, we observed that the constrained policy optimization objective was extremely sample inefficient. We observed no improvements in our training even after 5 billion sampling steps for the task of tracking base velocity commands. As a solution, we implement an approximated constrained proximal policy optimization (aCPPO) method along with generalized advantage estimate (GAE) [30], and optimize over the loss function represented in (3). After convergence, we perform hard constrained proximal policy optimization (hCPPO) using the objective $L_t^{CLIP+VF+S}$ and by introducing the cost return constraint expressed in (1). We collectively refer to both these optimization steps as constrained proximal policy optimization (CPPO).

In our work, we introduce three degrees of constraints and handle them accordingly:

1) *Soft* ($\rho$) constraints are included as part of the reward function. These need not be critical for safe operations. Instead these are introduced in order to direct policy search towards a desired behavior.
2) *Hard* ($\kappa$) constraints cannot be violated and are included in the set of constrained cost functions $C$. These are directly included during policy updates. In case of aCPPO, these are included in (2), and for hCPPO these are included as a constraint, as shown in (1) for the objective $L_t^{CLIP+VF+S}$.
3) *No-go* ($\eta$) constraints are introduced during training such that when $\kappa$-constraints are violated beyond a certain threshold the training episode is terminated to prevent exploration around regions which do not contribute towards policy optimization.

We use guided policy updates to warm start our control policy and then perform constrained proximal policy optimization for policy exploration. We then alternate between these during training as described in Alg. 1.

**Algorithm 1** Guided Constrained Policy Optimization

**Input:** $\lambda_d$, $s_{max}$, $c_{max}$, $n_{steps}$, $n_{batch}$, $in_{user}$, $c^h_{max}$, $H_a$, $H_h$
*Initialize* $\theta$, $\alpha = 1$
Generate set of trajectories $\mathcal{D} = \textsc{Controller}(in_{user})$ ▷ using model-based control strategy
**for** $t = 0, 1, 2 \ldots H_a$ **do**
  $\theta, \mathcal{D} = \textsc{GuidedPolicyUpdate}(\theta, \mathcal{D}, \alpha s_{max}, n_{batch})$
  $\theta = \textsc{PolicyOptimization}(\theta, n_{steps}, (1-\alpha)c_{max}, false)$
  $\alpha = \mathrm{e}^{\lambda_d t}$
**end for**
**for** $t = 0, 1, 2 \ldots H_h$ **do**
  $\theta = \textsc{PolicyOptimization}(\theta, n_{steps}, c^h_{max}, true)$
**end for**
**function** GuidedPolicyUpdate($\theta$, $\mathcal{T}$, $it_{max}$, $n_{batch}$)
  $l_{batch} = it_{max}/n_{batch}$
  Sample $l_{batch}$ state-action pairs $(s, a^*)$ from $\mathcal{T}$
  **for** $i = 0$ **to** $n_{batch} - 1$ **do**
    Generate $\{a\}$ for $\{s\}$ using $\pi_\theta$
    Update $\theta$ by minimizing $\sum_{j=0}^{l_{batch}} ||a_j - a_j^*||^2$
  **end for**
  $\mathcal{T} = \mathcal{T} \setminus \{(s, a^*)\}$
  **return** $\theta$, $\mathcal{T}$
**end function**
**function** PolicyOptimization($\theta$, $l_{episode}$, $it_{max}$, hard)
  $it = 0$, $\tau = \{\}$
  **while** $it < it_{max}$ **do**
    **for** $t = 0$ **to** $l_{episode} - 1$ **do**
      Sample $a_t \sim \pi_\theta(a|s_t)$
      $s_{t+1}, r_t, \{c_t\} = \textsc{RLEnvironmentStep}(a_t)$
      $it = it + 1$, $\tau = \tau \cup (s_{t+1}, r_t, \{c_t\})$
    **end for**
    **if** hard == $false$
      Update $\theta$ using aCPPO
    **else**
      Update $\theta$ using hCPPO
    **end if then**
  **end while**
  **return** $\theta$
**end function**

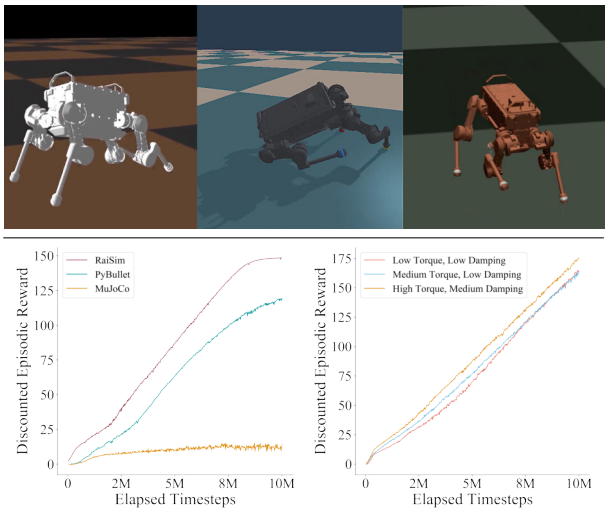



Fig. 2: Comparing differences in learnt behavior for different actuator models. **(top)** *left:* the RaiSim environment manages to sprint forward using a gait similar to galloping, *center:* the PyBullet environment converges at a local optimum where it uses hind legs to push its body forward while the front legs are not used, *right:* the MuJoCo environment struggles to find an optimal policy mainly because the desired joint positions at each step are not tracked reliably. **(bottom)** *left:* average discounted reward curves observed during training, *right:* average discounted reward curves observed during training in RaiSim for different ANYmal actuator models.

### B. Simulation

Most RL algorithms are sample inefficient and require significant amount of trials to learn a desirable control policy. Instead of training on a physical platform, which is slow and unsafe, we train the locomotion policy on a significantly faster simulation environment. However, policies trained in simulations often do not perform well in real-world systems. This is mostly due to the reality gap associated with simulations which do not perfectly model the physical world. Moreover, while performing experiments we realized different actuator models for ANYmal resulted in considerably different behaviors for the same training parameters.

We tested ANYmal simulations in RaiSim [31], PyBullet [32] and MuJoCo [33] for a simple task of moving forward with maximum feasible base velocity in order to compare the generated behaviors and training simulation times. The input to the control policy (34-dimensional state vector) consisted of $\{base_{height}, base_{orientation}, base_{twist}, joint_{states}\}$, and the output (12-dimensional action vector) consisted of $\{joint_{positions}\}$ desired for the next state. We trained the policies using PPO with the same hyperparameters, and on the same device, using the reward function $0.3 \times base_{forwardVel} - 4e{-}5 \times \|joint_{torque}\|^2$.

We trained the policies for up to 10M time steps with each iteration comprising of 76.8k episodic step samples. Using a discount factor $\gamma = 0.998$, and maximum episode length of 6.4k simulation steps we achieved the results represented in Fig. 2.

We performed experiments using different simulators and actuator models to validate our point that learnt behavior significantly depends on the setup of the RL environment, further substantiated by running experiments in RaiSim using different feed-forward torque and damping parameters for the actuators, as represented in Fig. 2. Moreover, none of the policies trained with different actuator models were observed to be inherently stable, necessitating the use of a good actuator model. For this, we used the same technique as authors of [15] to approximate the actuator model using a neural network trained through supervised learning. We also used the same network architecture as used in [15] for training the actuator network.

Moreover, due to sample inefficiency in most RL algorithms, it is important to consider time required for each simulation step. In our experiments, we observed that for same number of parallel executions, RaiSim was faster than both PyBullet and MuJoCo as shown in Table I. With several more parallel executions possible across multiple threads, we managed to execute 1B simulation steps in less than 3 hours in RaiSim on a PC.

### C. Environment Setup

Authors of [15] and [12] provide competitive baselines. We extend their approach to constrained policy optimization utilizing some of the hyperparameters used in their work. In this section we describe the setup of the ANYmal RL environment for the task of tracking user-generated reference base velocity commands.

*1) Observation Space:* In order to be extendable to the physical robot, the observation space chosen for the ANYmal environment needs to be accessible through on-board sensors and state-estimators. In this regard, the 109-dimensional state vector for the RL environment is defined as $\left\{b_h, \mathcal{O}, v_{base}, \omega_{base}, \mathcal{J}_t, \mathcal{J}^{des}_{t-1,t-2,t-3,t-4}, \dot{\mathcal{J}}_{t,t-1,t-2}, \mathcal{V}_{base}\right\}$ where $b_h$ is the robot base height, $\mathcal{O}$ is the base orientation, $v_{base}$ is the linear velocity in base frame, $\omega_{base}$ is the angular velocity in base frame, $\mathcal{J}_t$ is the joint position at time $t$, $\mathcal{J}^{des}_t$ is the policy output at time $t$, $\dot{\mathcal{J}}_t$ is the joint velocity at time $t$ and $\mathcal{V}_{base}$ is the user-generated desired base velocity expressed in base frame.

*2) Action Space:* The control policy outputs a 12-dimensional action vector comprising of $\{\mathcal{J}^{des}_t\}$. The desired joint positions are forwarded as an input to the approximated actuator network which outputs the torques for each of the joints for the ANYmal quadruped. These torques, clipped between $[-35\ Nm, 35\ Nm]$, are then directly applied to the joints.

*3) Network Architecture:* Since our work focuses on the constrained RL formulation, we decided to use the same network architecture implemented in [15], which had been already demonstrated to perform well.

*4) Reward Terms:* The reward terms are shown in Table II. These terms are multiplied by coefficients which are scaled further to increase the difficulty for the RL agent as training progresses.

*5) $\rho$-Constraint Costs:* These costs are directly added to the reward function, and are as shown in Table III.

TABLE I: Training time required for executing 10M simulation steps using 12 parallel environment runs tested on a PC housing an Intel i7-8700K and an Nvidia RTX 2080Ti.

| | **RaiSim** | **PyBullet** | **MuJoCo** |
|---|---|---|---|
| Training Time (seconds) | 1031.6403 | 2043.9825 | 1820.8244 |

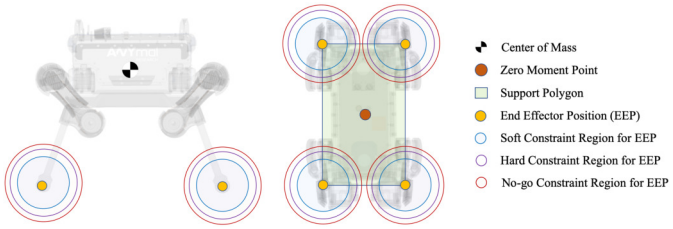



Fig. 3: Side-view and top-view of the ANYmal quadruped schematics representing some of the parameters used for constraint costs.

*6) $\kappa$-Constraint Costs:* These hard constraints are directly introduced in the aCPPO formulation as part of the expected discount cost return $J_{C_i}$ for cost term $i$. We introduce cost terms that account for stability constraints such as ZMP, as shown in Fig. 3, that have been used extensively for optimal control problem formulations. The $\kappa$-constraint cost terms for the aCPPO are shown in table IV. We do not use the ZMP term in hCPPO since the violation of ZMP can be caused due to external perturbations and discarding policies based on ZMP violations implies that we cannot perform state recovery.

In order to encourage recovery into a stable state upon violation of $\kappa$-constraints, we introduce an additive reward term for each of the constraints in case the robot state shifts back to obeying these constraints upon violations.

*7) $\eta$-Constraint Costs:* For cases when the control policy executes actions that cause the robot to land in unstable and unrecoverable states, we introduce $\eta$-constraints. Upon violations of these constraints we terminate the training episode, and disregard the training steps that may have been explored between $\kappa$-constraint and $\eta$-constraint violations and add a negative terminal reward to the last training step in the reformatted episode samples. The intuition behind this was to limit updates through explorations in regions that, apart from violating constraints, do not contribute to learning a feasible and desired locomotion behavior. The constraint terms are shown in table V. When any of the expressions evaluate to true, the training episode is terminated.

## III. Training

An overview of the training and validation process employed for our RL task is represented in Fig. 1.

TABLE II: Reward terms for the MDP formulation. Here $K$ refers to the logistic kernel defined as $K(x) = (e^x + 2 + e^{-x})^{-1}$, $\mathcal{V}^{lin}_{base}$ is the desired linear velocity in base frame, $\tau$ is the joint torque, $\mathcal{V}^{ang}_{base}$ is the desired angular velocity in base frame, $v^{foot}_{world,t}$ is the foot velocity in world frame at time $t$, and $\mathcal{O}^{base}_{x,y,z}$ is the base orientation along the $x, y, z$ axes.

| Term | Expression |
|---|---|
| Linear Velocity | $K\left(v_{base} - \mathcal{V}^{lin}_{base}\right)$ |
| Torque | $\|\tau\|^2$ |
| Angular Velocity | $K\left(\omega_{base} - \mathcal{V}^{ang}_{base}\right)$ |
| Foot Acceleration | $\|\|v^{foot}_{world,t} - v^{foot}_{world,t-1}\|\|^2$ |
| Foot Slip | $\|\|v^{foot}_{world}\|\|^2$ |
| Smoothness | $\|\|\mathcal{J}_t - \mathcal{J}_{t-1}\|\|^2$ |
| Orientation | $\|\|\mathcal{O}^{base}_{x,y,z} - \{0,0,\mathcal{O}^{base}_{z}\}\|\|^2$ |

### A. Generation of Reference Trajectories

We used a whole-body trotting controller to generate reference trajectories sampled at 400 Hz using the Gazebo simulator. The trajectories were represented as state-action $(s, a^*)$ pairs as detailed in Section II-C.

### B. Guided Policy Updates

Our training method alternates between supervised learning and reinforcement learning as presented in Alg. 1. The policy updates through supervised learning ensure that the policy search is directed towards a desired behavior. We trained the policy using the mean-squared-error loss between $a^*$ and $\pi_\theta(s)$ minimized using the Adam [34] optimizer.

While performing experiments, we observed that the policy action entropy had to be reduced after each session of supervised learning, else the policy search still preferred exploration. In fact, during some training experiments we found that, without precise reward function tuning, not reducing entropy caused the RL agent to converge at a local minimum. We empirically determined the reduction in entropy after each successive update.

### C. Constrained Policy Optimization

During policy exploration and optimization, we use the reward and cost functions described in the previous section to train the RL agent. We also introduce the following schemes to make our controller robust to unaccountable factors.

TABLE III: Cost terms for $\rho$-constraints. Here $\dot{\mathcal{J}}^{limit_\rho}$ refers to the joint speed limit for $\rho$-constraints, $\ddot{\mathcal{J}}^{limit_\rho}$ is the joint acceleration limit for $\rho$-constraints, $f_{h,i}$ is the height of end-effector $i$, $f^{des_\rho}_h$ is the desired end-effector height, $v_{f,i}$ is the velocity of the end-effector $i$, $f_i$ is the position of end-effector $i$, $\mathcal{R}^\rho_i$ is the corresponding feasible end-effector region for $\rho$-constraints, and $f^0_i$ is the base position for end-effector $i$ for a given joint configuration.

| Term | Expression |
|---|---|
| Joint Speed | $\left\|\max(\|\dot{\mathcal{J}}_t\| - \dot{\mathcal{J}}^{limit_\rho}, 0)\right\|^2$ |
| Joint Acceleration | $\left\|\max(\|\ddot{\mathcal{J}}_t\| - \ddot{\mathcal{J}}^{limit_\rho}, 0)\right\|^2$ |
| Foot Clearance | $\Sigma_i \left(f_{h,i} - f^{des_\rho}_h\right)^2 \left\|v_{f,i}\right\|^2$ |
| Foot Eligible Region | $\text{bool}\left(f_i \notin \mathcal{R}^\rho_i\right) \times \left\|f^0_i - f_i\right\|^2$ |

TABLE IV: Cost terms for $\kappa$-constraints. Here $\dot{\mathcal{J}}^{limit_\kappa}$ refers to the joint speed limit for $\kappa$-constraints, $\ddot{\mathcal{J}}^{limit_\kappa}$ is the joint acceleration limit for $\kappa$-constraints, $\mathcal{R}^\kappa_i$ is the corresponding feasible end-effector region for $\kappa$-constraints, $u$ is the ZMP, $\mathcal{S}$ is the region of support polygon with vertices given by the feet in contact with the ground, C is the center of mass of the quadruped, and $F_{f_i}$ is the contact force at foot $i$. The foot contacts cost term ensures that if 2 feet are in contact with the ground, they are not on the same side.

| Term | Expression |
|---|---|
| Joint Speed | $\left\|\max(\|\dot{\mathcal{J}}_t\| - \dot{\mathcal{J}}^{limit_\kappa}, 0)\right\|^2$ |
| Joint Acceleration | $\left\|\max(\|\ddot{\mathcal{J}}_t\| - \ddot{\mathcal{J}}^{limit_\kappa}, 0)\right\|^2$ |
| Foot Eligible Region | $\text{bool}\left(f_i \notin \mathcal{R}^\kappa_i\right) \times \left\|f^0_i - f_i\right\|^2$ |
| ZMP | $\text{bool}\left(u \notin \mathcal{S}\right) \times \|u - \text{C}\|^2$ |
| Foot Contacts | $\text{bool}(\Sigma_i(F_{f_i} > 0) < 3$ & $((F_{f_{lf}} > 0$ & $F_{f_{lh}} > 0)$ or $(F_{f_{rf}} > 0$ & $F_{f_{rh}} > 0)))$ |

TABLE V: $\eta$-constraint terms. Here $\dot{\mathcal{J}}^{limit_\eta}$ refers to the joint speed limit for $\eta$-constraints, $\ddot{\mathcal{J}}^{limit_\eta}$ is the joint acceleration limit for $\eta$-constraints, $\mathcal{R}_i^\eta$ is the corresponding feasible end-effector region for $\eta$-constraints and $u^\eta$ is the maximum allowed ZMP distance from the center of mass, C is the center of mass of the quadruped, and the minimum foot contacts have been set to 1 to ensure the control policy does not generate a behavior such as pronking.

| Term | Expression |
|---|---|
| Joint Speed | $\text{bool}(\dot{\mathcal{J}} > \dot{\mathcal{J}}^{limit_\eta})$ |
| Joint Acceleration | $\text{bool}(\ddot{\mathcal{J}} > \ddot{\mathcal{J}}^{limit_\eta})$ |
| Foot Eligible Region | $\text{bool}\left(f_i \notin \mathcal{R}_i^\eta\right)$ |
| ZMP | $\text{bool}\left(\|u - \text{C}\| > u^\eta\right)$ |
| Foot Contacts | $\text{bool}\left(\sum_i \text{bool}(F_{f_i} > 0) < 2\right)$ |

*1) Adding Noise to Observations and Actions:* We add Gaussian noise to the state and action vector [29] to account for sensor noise and inaccurate actuation. The standard deviation vector for the observation space is given as $s_c\{0.02, 0.1, [0.05]_3, [0.07]_3, [0.02]_{12}, [0.0]_{48}, [0.05]_{36}, [0.0]_3\}$, and for the action space, it is given as $s_c\{[0.04]_{12}\}$, where $s_c \in [0,1]$ is a scaling term increased over the training period.

*2) Changing Gravity:* We randomly sample acceleration due to gravity between $[0.95g, 1.05g]$, where $g = 9.81$ m/s$^2$ to emulate inertial scaling.

*3) Actuator Torque Scaling:* We randomly scale the output torque of our actuator network with the scaling coefficient $s_t \in [0.5, 2.0]$ to account for differences between the real actuators and the approximated model.

*4) Changing Link Mass and Size:* To ensure the training does not converge to a local minimum, we scale the mass and size of each of the links by coefficients $s_m^{link} \in [0.93, 1.07]$, and $s_l^{link} \in [0.97, 1.05]$ respectively.

*5) Adding Actuator Damping:* We emulate actuator damping by changing the output of the control policy using a complementary filter given as $\mathcal{J}_t^{des'} = K_{damp}\mathcal{J}_t^{des} + (1 - K_{damp})\mathcal{J}_{t-1}^{des'}$ where the gain $K_{damp}$ is randomized between $[1 - (s_c/4), 1]$.

*6) Changing Simulation Step Time:* For the possibility of execution on soft real time systems, we randomly set the step times of the control loop between $[2.25, 2.75]$ ms. During experiments we observed that the control policy even worked when the control frequency was changed from 400Hz to 200Hz.

## IV. Results and Discussion

We performed all training and experiments on commodity hardware; an Intel i7-8700K and Nvidia RTX 2080Ti. For training a control policy using our method, we required 450M simulation samples for about 240 policy optimization iterations using aCPPO, requiring less than 2 hours for a RaiSim based simulation. We then performed 36 policy iterations over 20M simulation samples using hCPPO. The authors of [15] required more than 7B simulation steps for convergence.

Having obtained a visually stable behavior in RaiSim, we tested the control policy in Gazebo simulation of ANYmal which consisted of an analytical actuator model. We then tested the control policy on the physical system. In our RL training, we do not use Gazebo because the compute time required for each simulation step is significantly larger than for RaiSim.

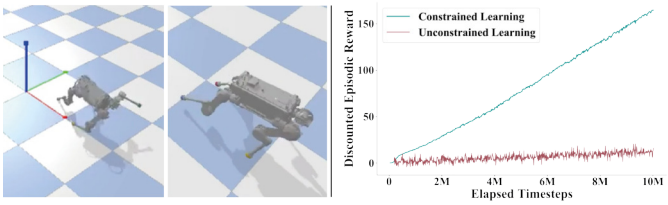



Fig. 4: Comparing differences in learnt behavior for constrained and unconstrained learning approaches. *left:* the unconstrained learning approach fails to develop an optimal strategy to track the desired base velocity commands even after 10M steps, *center:* the constrained learning approach limits the exploration of the policy within desired regions thereby directing policy optimization towards a preferred optimum, *right:* the average discounted episodic reward curves observed during training.

During our experiments with unconstrained learning methods, we observed that a constrained learning approach significantly directed policy convergence. We trained a control policy for a simple task of tracking forward base velocity of 0.7 m/s and observed that when we introduced even a basic constraint such as limits on end-effector positions with respect to the nominal stance, the policy trained using our constrained proximal policy optimization method performed much better than unconstrained proximal policy optimization, as is represented in Fig. 4. The reward for both the approaches was defined using the logistic kernel as $K(v_{base} - 0.7)$ scaled by a constant. We used the eligible foot regions defined in Table IV for constrained learning.

Moreover, for tracking base velocity commands, we required a minimum of about 2B simulation samples with precise reward function tuning to obtain a control policy similar to the trot controller. We changed the reward coefficients, increasing it for the torque required and decreasing for foot slip, foot clearance and smoothness empirically over at least 20 trials to get such a behavior. Without reward tuning, we obtained inefficient locomotion strategies such as pronking. This was, however, not the case with GCPO. Moreover, introducing GPUs in our approach helped us reduce the required training samples from approximately 1.6B, in the case of only CPPO, to 470M for GCPO.

The velocity tracking results obtained with our trained control policy on the physical robot system, outdoors on uneven terrain, are as shown in Fig. 5. It is important to note that the physical system comprised of additional sensor modules, amounting to approximately 10% of the robot mass, which had not been included in the simulations during training. We compared the results with the model-based trot controller we used for GPUs and observed that in most cases our controller tracks the velocity commands better than the trot controller as is evident from the tracking error plots. Here, we show that our policy closely tracks the velocity commands on the physical system despite having been trained on a simulator making this a successful sim-to-real policy transfer. Figure 6 represents the sum of the magnitude of torques measured at each joint for measured forward base velocities. We show that our controller, trained using GCPO, requires significantly lesser torque than the model-based trot controller. At most of the measured forward base velocities, the total joint torque measured for our controller is 20 Nm less than for the trot controller.

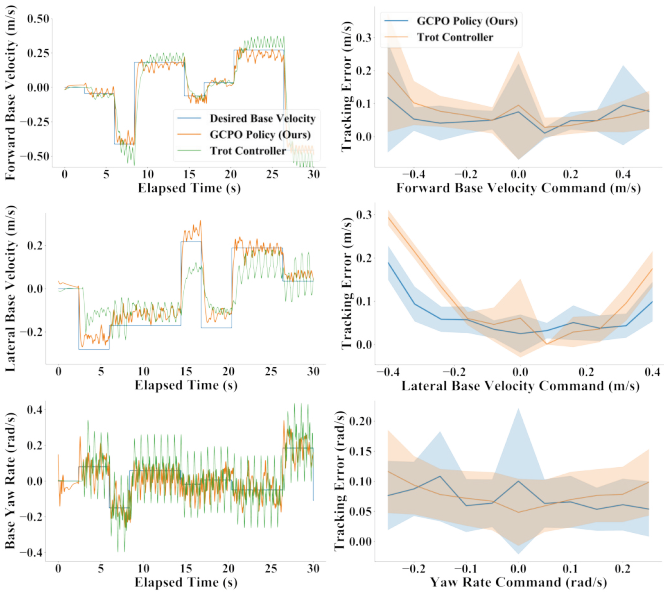



Fig. 5: Velocity tracking results obtained on the physical quadruped system. *Left:* base velocities measured for our trained control policy and model-based trot controller for the same sequence of velocity commands, *right:* the mean tracking error observed for given base velocity commands. The confidence bands represent the standard deviation of tracking error.

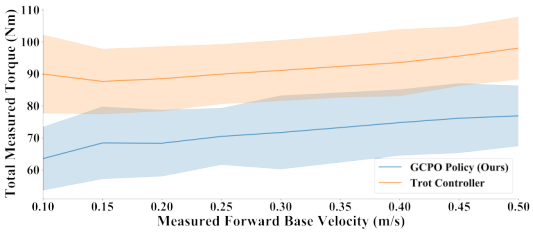



Fig. 6: The sum of the magnitude of joint torques measured on the physical system for each of the joints plotted against the measured forward base velocity. The confidence bands represent the standard deviation of torques measured. The controllers operate at 400 Hz.

As shown in Fig. 7, the density of joint velocity and joint accelerations for measured forward base velocity commands is very high below 5 rad/s and 50 rad/s$^2$ for the physical robot, while being even lower for RaiSim. In our experiments, for 5 minutes of data sampled at 400 Hz, we observed that the hard constraints on the physical system were only violated for 0.0196% and 0.0490% of times for joint velocities and joint accelerations respectively. For RaiSim, these values were 0.00119% and 0.00178%. As represented in Fig. 8 for the unconstrained approach, we measured these values in RaiSim and computed them to be 1.1710% and 1.2511%, 3 orders of magnitude larger than for our GCPO controller measured in RaiSim. Moreover, the density of the joint velocities and accelerations is significantly higher near the limits than for our constrained approach. It is important to note, however, that the behavior of the unconstrained control policies can be changed by significant re-tuning of the reward function. Furthermore, our control policy always maintains at least 2 stance legs during locomotion on flat terrain.

Exceeding our expectations, our GCPO controller was able to track the forward base velocity commands, on the physical

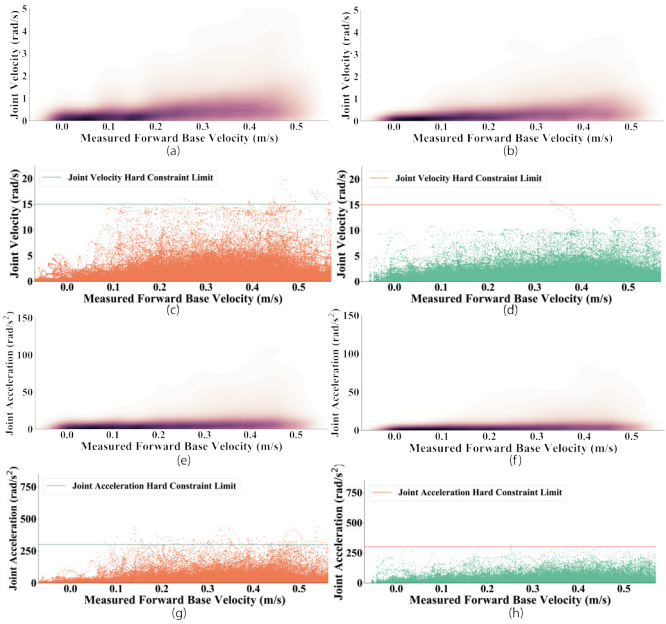



Fig. 7: Joint velocity and acceleration plots for each of the joint on the ANYmal quadruped plotted against measured forward base velocity. *(a)* the kernel density estimate (KDE) for the joint velocities measured on the physical system plotted using the parameters detailed in [35], *(b)* KDE for joint velocities measured in RaiSim, *(c)* scatter plot of the measured joint velocities on the physical system with hard constraint limit set to 15 rad/s, *(d)* scatter plot of the measured joint velocities in RaiSim, *(e)* KDE for joint accelerations measured on the physical system, *(f)* KDE for joint accelerations measured in RaiSim, *(g):* scatter plot of the measured joint accelerations on the physical system with hard constraint limit set to 300 rad/s$^2$, *(h)* scatter plot of the measured joint accelerations in RaiSim.

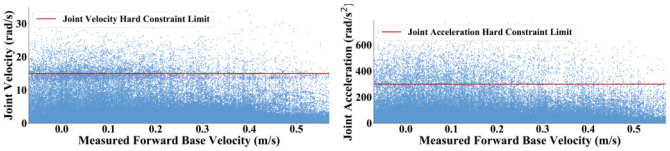



Fig. 8: *Left:* Joint velocity scatter plots obtained for every joint on the quadruped for an unconstrained locomotion behavior measured in RaiSim, *right:* joint acceleration scatter plots.

system, even when we introduced delays into the control execution with an RMS tracking error of 0.1736 m/s for following a base velocity command of 0.5 m/s for 10 s. These delays were randomly sampled from a uniform distribution between $[0, 17.5]$ ms. The controller became unstable when we sampled delays from distributions with upper limit greater than 20 ms. In most soft-realtime control systems, these delays are introduced by low-level hardware communication interfaces, and such a robust controller is certainly desirable.

We observed our controller's response to external perturbations by applying forces with magnitude ranging from 50 N to 120 N for duration between 1 s to 5 s to the robot base in Gazebo. We observed that our controller responded to these external perturbations by moving in the direction opposite to that of the applied force, ensuring it's stability. The controller was able to respond to external forces of up to 100 N for 1 s duration applied to the base horizontally. We also tested the controller's response on the physical system.

We emulated a weak actuator, to test the case where an

actuator becomes damaged, by reducing the position tracking gain of the weak actuator to 24. The position gain for all other actuators was set to 48. We observed that our controller was still able to track the base velocity commands with an RMS tracking error of 0.1975 m/s for following a base velocity command of 0.5 m/s for 10 s.

Furthermore, despite significant inertial scaling, our control policy, without any parameter re-tuning, still managed to track velocity commands even when we set the acceleration due to gravity to 1.62 m/s$^2$ in RaiSim. We observed an RMS tracking error of 0.2214 m/s for following a base velocity command of 0.5 m/s for 10 s.

## V. Conclusion

We presented an RL training method for quadrupedal locomotion which considers safety-critical constraints in its problem formulation and further encourages system recovery into stable states upon constraint violations. We used reference trajectories, obtained using a trot controller, to perform GPUs in order to direct policy optimization. Our experiments demonstrate that our RL method offers a robust controller which requires significantly lesser torque for execution compared with a model-based controller. Furthermore, it is important to note that constrained policy optimization does not necessitate use of GPUs. In our work, CPPO can be used even without GPUs, but have been introduced to limit exploration and hence reduce the samples required for convergence.

As part of future research, we aim to demonstrate the applicability of our method to more complex environments, and to extend our method to use perception data as a means to generalizing to a wide variety of challenging terrains.

## Acknowledgement

Siddhant would like to thank Dr. Jemin Hwangbo from RSL, ETH Zurich for his valuable inputs. We would also like to thank Luigi Campanaro and Benoit Casseau for helping us with our experiments.

## References

[1] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[2] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet, "Sample-efficient batch reinforcement learning for dialogue management optimization," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 7, no. 3, p. 7, 2011.

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

[6] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 38–44.

[7] C. Chow and D. Jacobson, "Studies of human locomotion via optimal programming," *Mathematical Biosciences*, vol. 10, no. 3-4, pp. 239–306, 1971.

[8] A. E. Bryson, *Applied optimal control: optimization, estimation and control*. Routledge, 2018.

[9] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[10] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 558–564.

[11] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[12] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

[13] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 41, 2017.

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

[15] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[16] F. Zhang, J. Leitner, M. Milford, and P. Corke, "Modular deep q networks for sim-to-real transfer of visuo-motor policies," *arXiv preprint arXiv:1610.06781*, 2016.

[17] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.

[18] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," *arXiv preprint arXiv:1812.02341*, 2018.

[19] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[20] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[21] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, 2016, pp. 49–58.

[22] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, 2018.

[23] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[24] K. Doya, K. Samejima, K.-i. Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," *Neural computation*, vol. 14, no. 6, pp. 1347–1369, 2002.

[25] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 22–31.

[26] M. L. Puterman, *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[27] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[28] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[29] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

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

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

[32] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[33] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.

# 4

# Paper II
# Rapid Stability Margin Estimation for Contact-Rich Locomotion

**This work was presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2021.**

# Manuscript Preface

This work introduces a deep neural network parameterization for fast estimation of quadrupedal robot stability. The improved computational performance makes the proposed approach suitable for training deep RL locomotion policies and performing real-time trajectory optimization.

The experimental results obtained after introducing the rapid stability margin estimation approach for trajectory optimization and deep reinforcement learning are detailed in Section V. Section V-A presents the reliability and accuracy of the stability margin estimation. Figure 1 and Figure 6 illustrate the stability margin estimated for varying base and left-front foot positions respectively. The color maps indicate the value of the stability margin at each of the sampled points on the terrain. Figure 5 provides the snapshots of a simulated quadrupedal traversal over uneven terrain using trajectory optimization. The color map represents the quadruped's perceived elevation of the terrain. Section V-C details upon introducing stability margin in the reward function of an RL formulation.

# Rapid Stability Margin Estimation for Contact-Rich Locomotion

Romeo Orsolino, Siddhant Gangapurwala, Oliwier Melon,
Mathieu Geisert, Ioannis Havoutis and Maurice Fallon

***Abstract*— The efficient evaluation the dynamic stability of legged robots on non-coplanar terrains is important when developing motion planning and control policies. The inference time of this measure has a strong influence on how fast a robot can react to unexpected events, plan its future footsteps or its body trajectory. Existing approaches suitable for real-time decision making are either limited to flat ground or to quasi-static locomotion. Furthermore, joint-space feasibility constraints are usually not considered in receding-horizon planning as their high dimensionality prohibits this.**

**In this paper we propose the usage of a stability criterion for dynamic locomotion on rough terrain based on the Feasible Region (FR) and the Instantaneous Capture Point (ICP) and we leverage a Neural Network (NN) to quickly estimate it.**

**We show that our network achieves satisfactory accuracy with respect to its analytical counterpart with a speed up of three orders-of-magnitude. It also enables the evaluation of the stability margin's gradient. We demonstrate this *learned stability margin* in two diverse applications - Reinforcement Learning (RL) and nonlinear Trajectory Optimization (TO) for legged robots. We demonstrate on a full-sized quadruped robot that the network enables the computation of physically-realizable Center of Mass (CoM) trajectories and foothold locations satisfying friction constraints and joint-torque limits in a receding-horizon fashion and on non-coplanar terrains.**

## I. Introduction

Dynamic locomotion of legged robots on rough terrains is hard to achieve because of the high dimensionality of multi-limbed floating-base systems and the need for fast decisions about the footsteps placement and base trajectory. Typical model-based approaches consist of optimal control and TO formulations which either focus on quasi-static locomotion or use simplified dynamics' models to reduce the dimensionality of the system. These simplified models capture the main dynamics of the robotic platform but fail to grasp important feasibility constraints of legged robots such as the kinematic (leg workspace and joint-velocity) and dynamic (*e.g.,* friction cones and joint-torque) constraints. Instead of being used in an anticipative capacity, these limits are therefore usually only verified at the control level and can saturate in case of violation. However, these physical and hardware limits become significant when the robot negotiates challenging environments and gets closer to its performance envelope. When complex configurations are required, joint-level feasibility constraints cannot be neglected at the planning level [1].

This work was supported by the the UKRI/EPSRC RAIN [EP/R026084/1] and ORCA [EP/R026173/1] Hubs and the EU H2020 Projects MEMMO and THING, the EPSRC grant 'Robust Legged Locomotion' [EP/S002383/1] and a Royal Society University Research Fellowship (Fallon). This work was conducted as part of ANYmal Research, a community to advance legged robotics. This work was part of the Human-Machine Collaboration Programme, supported by a gift from Amazon Web Services. The authors are with Dynamic Robots Systems (DRS) group, Oxford Robotics Institute, University of Oxford, UK. email: rorsolino, siddhant, omelon, mathieu, ioannis, mfallon@robots.ox.ac.uk

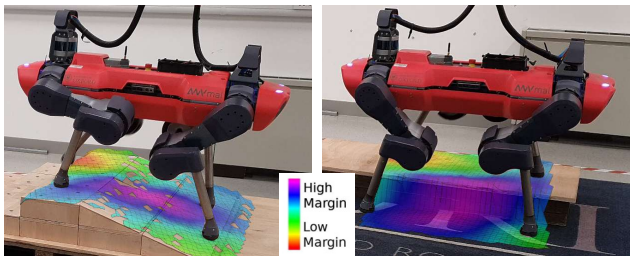



Fig. 1. The colormaps represent the value of the learned stability margin for different horizontal positions of the robot's base. High margin values (purple) correspond to $0.07\,\mathrm{m}$ whereas the low values (red) correspond to $-0.17\,\mathrm{m}$.

One approach to tackle the computational complexity of floating base systems is to project the joint-space constraints into lower-dimensional spaces, typically at the centroidal level (2-D or 6-D). Such projections are low-dimensional and preserve the descriptiveness of the original space but they are often expensive to compute thus defeating their purpose.

Along a different track of research, data-driven methods attempt to move the computational burden offline, thus interpreting the online decision process as a mere input-output relationship between state and the learned behavior. In this framework, RL methods have recently gained interest due to their reactiveness which allows them to retain stability on non-flat terrains [2], [3], [4]. Unfortunately, these approaches require a significant amount of offline data generation and training and are sensitive to reward function tuning.

### A. Contributions

In this paper, we demonstrate how machine learning can allow fast evaluation of feasibility constraints (contact friction constraints and joint-torque limits) projected onto a 2-D space. The obtained network is a mathematical tool suitable for the *online* generation of dynamic locomotion in presence of non-coplanar contacts in different control settings. Our contributions can be summarized as follows:

- We leverage data-driven methods to evaluate dynamic stability of legged robots in the presence of non-coplanar contacts. For a small loss of accuracy, we gain a three-order-of-magnitude speed-up in the evaluation of the *stability margin* and the possibility to compute its gradient, which would otherwise be only possible by finite-difference of the analytical computation.

- we showcase applications of the mentioned tool in both TO and RL. We validate our approach in a physics-based simulator over multiple gaits (walk and trot) and on the hardware ANYmal C quadruped robot [5] in presence of non-coplanar scenarios.

## II. Related Works

The fast assessment of the stability of a legged robot is a core issue in motion planning and control, especially for multi-contact scenarios [6], [7]. The fast binary assessment of static stability can be performed by means of Linear Programming (LP) [7] or Incremental Projection (IP) algorithms [6]. Optimal control and TO approaches, however, require a quantitative–and *non-binary*–measure of robustness [8].

Only a few works have focused on dynamic stability on complex terrains, mostly through the generalization of the *capturability analysis* [9] to height variations [10] or to the 3-D space of the CoM position [11], [12]. Other approaches have instead focused on the generalization of the Zero Moment Point (ZMP) onto the 6-D centroidal space [13], [14]. Some of them have also included the possibility to synthesize body trajectories that avoid joint-torque limits [15]. All of them, however, assume the contact locations to be predetermined and they are therefore not suitable for footstep planning on rough terrains.

2-D projections of such sets are possible [16], which enables sampling-based footstep planning on rough terrains [17]. Footstep planning has also been performed through A* based algorithms [18], mixed-integer convex programming [19] or LP [20]. Such methods, however, they all consider predefined base trajectories. Simultaneous optimization of base trajectories and contact locations on non-coplanar terrains has been achieved using mixed-integer programming [21] and nonlinear TO [22] but their computation time does not allow these strategies to be deployed in a receding-horizon fashion.

Approaches based on machine learning, on the other hand, have demonstrated their ability to generate dynamic locomotion behaviors on both flat grounds [23] and rough terrains [2]. These approaches, however, require long offline training phases and accurate reward function tuning. Tsounis *et al.* [3] leverage a model-based feasibility method [24] to guarantee the feasibility of their policy and to discard physically-unrealizable trajectories. Our RL application presented in this paper, presents similarities with this work in the sense that we also leverage a dense reward function to improve the sampling efficiency of our training phase; the reward function is in our case, however, an estimate of the analytical feasibility measure learned by means of a NN. Learning of the robot's centroidal dynamics has already been proposed to speed up motion planners on rough terrain [25], [26]. However, to the best of our knowledge, it is the first time that a learned dynamic feasibility constraint is demonstrated on the hardware for multiple gaits and multiple terrain.

On a similar line to our approach, Carpentier *et al.* [27] proposed to learn feasibility constraints in order to overcome the computation burden and enable online planning on complex terrain. Differently from our work, however, they focused on kinematic limits and learned a probability density function of the CoM positions whereas we focus on friction constraints and joint-torque limits. Besides that, we demonstrate the applicability of our method for simultaneous optimization of base trajectories and contact locations in two different settings: nonlinear TO and RL.

## III. Background

For completeness, we summarize in the following the steps required to compute the feasible region [17] and the ICP. Based on these quantities we then define *analytical* stability margin, a mathematical tool which we employ in this paper to quantify the dynamic stability of legged robots during motion over rough terrain. In Sec. IV we go on to explain how we use a data-driven method to offset the computational complexity of this analytical approach.

### A. Feasible Region

Bretl *et al.* [6] demonstrated how the support region differs from the convex hull of contacts points when the robot establishes non-coplanar contacts with the environment. The authors demonstrated how the IP algorithm can be employed to compute the support region and to efficiently test the stability of the robot [6], [28]. Subsequently, this method was extended to account for joint-torque limits [17] and to dynamic conditions such as underactuated support configurations (single and double point contacts) and external wrenches [1].

A summary of this improved strategy is given in Alg. 1. The algorithm iteratively solves a LP whose optimization variables are the CoM projection $\mathbf{c}_{xy} \in \mathbb{R}^2$, the contact forces $\mathbf{f}_i \in \mathbb{R}^3$ and the contact torques $\mathbf{w}_i \in \mathbb{R}^2$. The LP seeks the furthest point $\mathbf{c}^*_{xy}$ along the $\mathbf{a}_j \in \mathbb{R}^2$ direction on the $(x, y)$ plane that satisfies dynamic equilibrium (III.a) while the contact forces of each end-effector $i$ in contact with the

**Algorithm 1** IP algorithm.

**Input:** $\boldsymbol{\lambda} \in \mathbb{R}^{47}$
**Result:** feasible region $\mathcal{Y}_{fr}$
**Initialization**: $\mathcal{Y}_{outer}$, $\mathcal{Y}_{inner}$, get $\mathbf{q}$ given $\boldsymbol{\lambda}$ through IK.
**while** $area(\mathcal{Y}_{outer}) - area(\mathcal{Y}_{inner}) > \epsilon$ **do**
I) compute the edges $\mathbf{e}_j$ of $\mathcal{Y}_{inner}$
II) pick $\mathbf{a}_j$ cutting off the largest fraction of $\mathcal{Y}_{outer}$
III) solve the LP:

$$\mathbf{c}^*_{xy} = \underset{\mathbf{c}_{xy}, \mathbf{f}_i, \mathbf{w}_i}{\operatorname{argmax}} \quad \mathbf{a}_j^T \mathbf{c}_{xy}$$

such that :

$$\text{dynamic equilibrium} \qquad \text{(III.a)}$$
$$\mathbf{f}_i \in \mathcal{C}_i, \quad \forall i = \{1, 2, \dots N_c\} \qquad \text{(III.b)}$$
$$\mathbf{f}_i \in \mathcal{F}_i, \quad \forall i = \{1, 2, \dots N_c\} \qquad \text{(III.c)}$$
$$\mathbf{w}_i \in \mathcal{B}_i, \quad \forall i = \{1, 2, \dots N_c\} \qquad \text{(III.d)}$$

IV) update the outer approximation $\mathcal{Y}_{outer}$
V) update the inner approximation $\mathcal{Y}_{inner}$
**end while**

environment lie within their friction cone $\mathcal{C}_i$ (III.b):

$$\mathcal{C}_i = \left\{ \mathbf{f}_i \in \mathbb{R}^3 \quad | \quad ||(\mathbb{1} - \mathbf{n}_i \mathbf{n}_i^T)\mathbf{f}_i||_2) \leq \mu \mathbf{n}_i^T \mathbf{f}_i \right\} \tag{1}$$

and force polytope $\mathcal{F}_i$ (III.c):

$$\begin{aligned} \mathcal{F}_i = \Big\{ \mathbf{f}_i \in \mathbb{R}^3 | \quad \exists \boldsymbol{\tau}_i \in \mathbb{R}^{n_a} s.t. \quad \mathbf{M}_{bi}^T \dot{\boldsymbol{\nu}}_b + \mathbf{M}_i \ddot{\mathbf{q}}_i + \\ \mathbf{c}(\mathbf{q}_i, \dot{\mathbf{q}}_i) + \mathbf{g}(\mathbf{q}_i) = \boldsymbol{\tau}_i + \mathbf{J}(\mathbf{q}_i)^T \mathbf{f}_i, \quad \underline{\boldsymbol{\tau}}_i \leq \boldsymbol{\tau}_i \leq \overline{\boldsymbol{\tau}}_i \Big\} \end{aligned} \tag{2}$$

and while the contact torques $\mathbf{w}_i$ satisfy the box constraint $\mathcal{B}_i$ (III.d):

$$\mathcal{B}_i = \left\{ \mathbf{w}_i \in \mathbb{R}^2 \quad | \quad \underline{\mathbf{w}}_i \leq \mathbf{w}_i \leq \overline{\mathbf{w}}_i \right\} \tag{3}$$

$\mu \in \mathbb{R}$ represents the friction coefficient between the robot's foot and the ground (assumed here to be the same for all contacts). The vectors $\mathbf{q}_i, \dot{\mathbf{q}}_i$ and $\ddot{\mathbf{q}}_i \in \mathbb{R}^{n_a}$ represent the joint position, velocity and acceleration of the $i^{th}$ limb of the articulated robot. $\boldsymbol{\nu}_b = [\dot{\mathbf{c}}^T, \boldsymbol{\omega}^T]^T \in \mathbb{R}^6$ represents the generalized base velocity. The matrices $\mathbf{M}_{bi}, \mathbf{M}_i$ and $\mathbf{J}(\mathbf{q}_i)$ represent, respectively, the inertia of the base, the inertia of the legs, and the Jacobian. The vectors $\underline{\boldsymbol{\tau}}_i, \overline{\boldsymbol{\tau}}_i \in \mathbb{R}^{n_a}$ represent the minimum and maximum joint-torque limits of the $n_a$ actuators of one individual leg $i$. $\mathbf{w}_i = [w_x, w_y]^T \in \mathbb{R}^2$ is an infinitesimal contact torque (decoupled from the contact forces) having maximum value of $\overline{\mathbf{w}}_i = [w_M, w_M]^T \in \mathbb{R}^2$ and minimum value of $\underline{\mathbf{w}}_i = -\overline{\mathbf{w}}_i$ such that $w_M \in \mathbb{R} \to 0$. The torques $\mathbf{w}_i$ enable the computation of the feasible region also in presence of single or double point contacts [1].

The set of all variables required to quantify the robot's instantaneous dynamic balance are the following:

$$\begin{aligned} \boldsymbol{\lambda} = ( \underbrace{\mathbf{e}_g}_{\mathcal{B}}, \underbrace{\dot{\mathbf{c}}, \ddot{\mathbf{c}}, \dot{\boldsymbol{\omega}}, \mathbf{f}_{ext}, \boldsymbol{\tau}_{ext}, \mathbf{p}_1, \dots \mathbf{p}_{N_c}}_{\mathcal{H}}, \\ \mu, c_1, \dots c_{N_c}, \underbrace{\mathbf{n}_1, \dots \mathbf{n}_{N_c}}_{\mathcal{H}}) \in \mathbb{R}^{47} \end{aligned} \tag{4}$$

$\mathbf{e}_g \in \mathbb{R}^3$ is a unit vector representing the gravity axis in the base frame $\mathcal{B}$, *i.e.,* the third row of the robot's base orientation matrix $\mathbf{R} \in SO(3)$. $\dot{\mathbf{c}}, \ddot{\mathbf{c}} \in \mathbb{R}^3$ are the linear velocity and acceleration of the robot's CoM whereas $\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$ is the angular acceleration of the robot's base. $\mathbf{f}_{ext}, \boldsymbol{\tau}_{ext} \in \mathbb{R}^3$ compose the wrench of an external disturbance applied on the base of the robot (if any). $\mathbf{p}_i \in \mathbb{R}^3$, $\mathbf{n}_i \in \mathbb{R}^3$ and $c_i \in \{0, 1\}$ for $i = \{1, \dots N_c\}$ represent the feet positions, the contact normals and the contact state, respectively. Each of these quantities provided in the *horizontal* frame[1] as indicated in (4). It is important to notice that an implicit information about the robot's CoM position $\mathbf{c}$ is encoded in $\boldsymbol{\lambda}$ thanks to the definition of the feet positions $\mathbf{p}_i$ with respect to $\mathcal{H}$. The joint-space variables required in (2), but missing from (4), can be determined from $\boldsymbol{\lambda}$ through Inverse Kinematics (IK). For articulated robots with non-redundant limbs, such as ANYmal C, the IK problem can be efficiently solved analytically.

[1]The horizontal frame has been also called with multiple other names in the literature such as *control*, *local* or *yaw* frame [29]. It differs from the *global* frame because of its different location and yaw angle. It differs from the robot's *base* frame for its different roll and pitch angles.

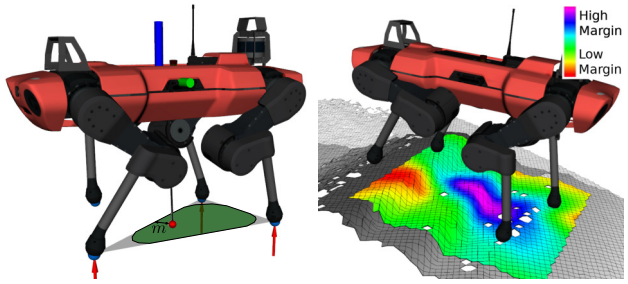



Fig. 2. Left: we define the stability margin as the perpendicular distance from the contour of the feasible region (green region) and the ICP (red sphere). Right: colormap representing the values of the learned stability margin where purple is the highest and red is the lowest.

At each iteration of Alg. 1 a new vertex $\mathbf{c}_{xy}^*$ is added to the inner approximation $\mathcal{Y}_{inner}$ and a new edge orthogonal to $\mathbf{a}_j$ and passing through $\mathbf{c}_{xy}^*$ is added to $\mathcal{Y}_{outer}$ [6].

Once $\mathcal{Y}_{inner}$ and $\mathcal{Y}_{outer}$ have converged to the desired tolerance $\epsilon$, we obtain the feasible region $\mathcal{Y}_{fr}$ as the *halfspace-description* of $\mathcal{Y}_{inner}$, consisting of a number $N_e \in \mathbb{N}$ of edges $\mathbf{e}_j \in \mathbb{R}^2$ with $j = 1, \dots, N_e$.

### B. Capture Point

The ICP $\boldsymbol{\xi} \in \mathbb{R}^2$ has been defined by Pratt *et al.* [9] as the point on the flat ground where a robot, given its instantaneous linear momentum, should place its foot to come to a complete stop:

$$\boldsymbol{\xi} = \mathbf{c}^{xy} + \frac{\dot{\mathbf{c}}^{xy}}{\omega} \tag{5}$$

where $\omega = \sqrt{g/c_z}$ is the natural frequency of the Linear Inverted Pendulum (LIP) model while $\mathbf{c}^{xy}$ and $\dot{\mathbf{c}}^{xy}$ are the projection on the horizontal $(x, y)$ plane of the CoM position and velocity. The ICP $\boldsymbol{\xi}$ can also be considered as the point where the CoM projection is converging to [9]. We employ it as a ground reference point to determine the instantaneous stability of our system: the robot is stable if: $\boldsymbol{\xi} \in \mathcal{Y}_{fr}$.

### C. Analytic Stability Margin

The feasible region is a convex set by construction [17]; we can compute the normals $\mathbf{n}_{e,j} \in \mathbb{R}^2$ to its edges and the offset terms $a_j \in \mathbb{R}$ such that $\mathbf{n}_{e,j} \cdot \mathbf{p} - a_j = 0$ if $\mathbf{p} \in \mathbb{R}^2$ is a point that lies along $\mathbf{e}_j$. The normals can be stacked into a matrix $\mathbf{N}$:

$$\mathbf{N} = \begin{bmatrix} \mathbf{n}_{e,1}^T & a_1 \\ \vdots & \vdots \\ \mathbf{n}_{e,N_e}^T & a_{N_e} \end{bmatrix} \in \mathbb{R}^{N_e \times 3} \tag{6}$$

so that the signed distances $\mathbf{d} \in \mathbb{R}^{N_e}$ between a 2-D point $\mathbf{p}$ and all the edges of $\mathcal{Y}_{fr}$ can be computed as: $\mathbf{d} = \mathbf{N} \cdot [\mathbf{p}^T \quad 1]^T$. If all the normals $\mathbf{n}_i$ are pointing inwards then all the elements of $\mathbf{d}$ will be positive if $\mathbf{p}$ lies within $\mathcal{Y}_{fr}$. We can then define the *analytic* stability margin $m_a$ (see Fig. III-B) as the minimum distance between the ICP $\boldsymbol{\xi}$ and the edges of the feasible region:

$$m_a = \min(\mathbf{N} \cdot [\boldsymbol{\xi}^T \quad 1]^T) \tag{7}$$

This will result into a positive value if $\boldsymbol{\xi}$ lies inside the feasible region and in a null or negative value otherwise. In Sec. IV we will explain how we used a NN to estimate this quantity from data.

## IV. Learned stability margin

The algorithm presented in the previous section depends on the centroidal and contacts state of the robot (from which also the required joint-space states needed to determine the force polytopes can be inferred).

The vector $\boldsymbol{\lambda} \in \mathbb{R}^{47}$ represents the input state of a Multi-Layer Perceptron (MLP) that we employed to estimate the output of the IP algorithm that we described in Section III-C. The learned stability margin $m_l$ can thus be obtained as:

$$m_l = MLP(\boldsymbol{\lambda}) \tag{8}$$

### A. Learning

We followed the following procedure to train the network:

*1) Network Architecture:* The dense feed-forward network used as the function approximator of the stability margin comprises the 47-dimensional input $\boldsymbol{\lambda}$, 1-dimensional output $m_l$, and 3 hidden layers (256 nodes in the first layer and 128 nodes in the other two layers). We use the softsign activation function to introduce non-linearity in the network.

*2) Generating Training Data:* We use the parameters represented in Table I as input to the network. In order to generate the training set we sample these parameters from the distributions shown in Table I and use the IP algorithm to compute the analytical stability margin $m_a$ for each set of the parameters. Note that whereas we do account for angular accelerations of the robot's base, the base angular velocity $\boldsymbol{\omega}$ is the only dynamic parameter that is not accounted for in our strategy, it does not belong to input vector $\boldsymbol{\lambda}$ and therefore no random data generation is required for this variable.

We noted that the usage of a uniform distribution for the feet locations and for the base velocity and acceleration ($x, y$ components) was particularly useful to make sure that the accuracy of the trained network would not quickly deteriorate whenever the robot deviated significantly from its nominal configuration (in which all the joint-positions are equidistant from their kinematic end-stops) or from static conditions.

*3) Supervised Learning:* We then trained the network by generating a dataset of $50 \cdot 10^6$ samples using a supervised learning approach employing the Adam optimization strategy [30] for backpropagation.

A graphic representation of the estimates returned by the network is given in Fig. 2 (right). In addition to the substantial speed up in the inference time, the so obtained MLP also enables partial derivative computation of the output (the stability margin) with respect to the input state. As we will in next Section, this leads to the possibility to directly employ the MLP in nonlinear optimization problems whose variables depend on the elements of $\boldsymbol{\lambda}$.

### B. Nonlinear Trajectory Optimization

To demonstrate the applicability of the learned stability margin $m_l$ in a motion planning setting, we have formulated a nonlinear TO problem which leverages the proposed MLP to enable the concurrent optimization of the robot's base trajectory (pose, linear and angular velocities) and contact locations on arbitrary terrains in a receding-horizon fashion:

TABLE I

Network input parameters and the sampling distributions used for generating training data.

| Parameter | Distribution |
|---|---|
| Feet in contact | $X \sim U(0, 3)$ |
| Base roll and pitch angles | $\alpha, \beta \sim U(-\pi/15, \pi/15)$ |
| COM linear velocity | $\dot{x}, \dot{y} \sim U(-1.25, 1.25)$<br>$\dot{z} \sim \mathcal{N}(0, 0.2)$ |
| COM linear acceleration | $\ddot{x}, \ddot{y} \sim U(-2.5, 2.5)$<br>$\ddot{z} \sim U(-1.25, 1.25)$ |
| Base angular velocity | – |
| Base angular acceleration | $\dot{\omega}_x, \dot{\omega}_y \sim U(-1.0, 1.0)$<br>$\dot{\omega}_z \sim \mathcal{N}(0, 0.5)$ |
| Friction coefficient | $\mu \sim U(0.2, 0.9)$ |
| $i-th$ foot deviation from nominal position | $p_i^x \sim U(-0.35, 0.35)$<br>$p_i^y, p_i^z \sim U(-0.2, 0.2)$ |
| Contact normal rotations (about gravity axis) | $\alpha \sim \mathcal{N}(0, \pi/8)$<br>$\beta \sim \mathcal{N}(0, \pi/8)$ |
| External force | $F_x, F_y, F_z \sim \mathcal{N}(0, 50)$ |
| External torque | $\tau_x, \tau_y, \tau_z \sim \mathcal{N}(0, 25)$ |

$$\underset{\mathbf{x}_b(t_k), \boldsymbol{\nu}_b(t_k), \mathbf{p}_i(t_j)}{\operatorname{argmin}} \quad \underbrace{w \left|\left| \frac{\partial h_t}{\partial \mathbf{p}_i^{xy}(t_j)} \right|\right|_2^2}_{\mathcal{C}}$$

such that :

$$\begin{aligned}
MLP(\boldsymbol{\lambda}(t_k)) \geq & m_{min} && \text{(minimum stability margin)} \\
\mathbf{p}_i(t_j) \leq & \mathcal{R}(\mathbf{x}_b) && \text{(foot range of motion)} \\
p_i^z(t_j) = & h_t(\mathbf{p}_i^{xy}) && \text{(terrain height)} \\
\mathbf{x}_b(t_0) = & \mathbf{x}_{t_0,b}, \quad \boldsymbol{\nu}_b(t_0) = \boldsymbol{\nu}_{t_0,b} && \text{(initial state)} \\
\mathbf{x}_b(t_N) = & \mathbf{x}_{T,b}, \quad \boldsymbol{\nu}_b(t_N) = \boldsymbol{\nu}_{T,b} && \text{(target state)}
\end{aligned} \tag{9}$$

where $\mathbf{x}_b = [\mathbf{c}, \mathbf{R}] \in SE(3)$ is the pose of the robot's base link and $\mathbf{R} \in SO(3)$ is the base orientation. The subscript $i$ represents the end-effector indices $\{1, 2, \dots N_c\}$ (where $N_c$ is the number of contacts) whereas the timestamps $t_k$ (with $k = 1, \dots, N$) represent the time instants of the base discretization nodes, equally spaced along the optimization horizon. For the feet variables, rather than using a uniformly spaced discretization like the base variables, an event-based parametrization has been performed, only setting a foot node at each touch down instant $t_j$ and keeping that variable fixed throughout the stance phase duration ($j = 1, \dots N_{s,i}$ where $N_{s,i}$ is the predefined number of steps of the $i^{th}$ foot over the horizon $T$). The problem has been formulated using third-order polynomials to ensure continuity between the nodes. The feet sequence here is predefined and so is the duration of each stance and swing phase. The vertical coordinate of the

footholds $p_i^z$ is set equals to $h_t$, as enforced by the *terrain height* constraint. The cost $\mathcal{C}$ finds a foothold on the 2.5-D map that minimizes the terrain slope which enables the robot to avoid inclined surfaces and edges as it navigates over rough terrain.

The *minimum stability margin* constraint is used to enforce dynamic feasibility of the trajectory all along the optimization horizon. The value of $m_{min}$ is gait dependent: in the trot gait, for example, the feasible region is a 1-D segment and the maximum achievable value will thus be zero (when the capture point $\boldsymbol{\xi}$ belongs to the segment). A minimum margin value $m_{min} = -0.05$, for example, will guarantee that the capture point does not diverge during the optimization horizon and that, consequently, the robot does not fall. This constraint is computed using the learned stability margin $m_l$ from the dynamic state of the robot (4) at each base's discretization nodes. As mentioned above, the derivative of $m_l$ with respect to the optimization variables, which is necessary for the solution of our TO problem, can be obtained as a function of the of the vector $\partial m_l / \partial \boldsymbol{\lambda}$, by backpropagation of the MLP. Consider, the CoM position variable $\mathbf{c}$, expressed in (9) with respect to the world frame $\mathcal{W}$. Although $\mathbf{c}$ is not part of the MLP's input state $\boldsymbol{\lambda}$, the partial derivative $(\partial m / \partial \mathbf{c})$ can still be obtained as a function of the feet positions $\mathbf{p}$ expressed in the horizontal frame $\mathcal{H}$ (which are part of $\boldsymbol{\lambda}$):

$$\frac{\partial m}{\partial \mathbf{c}} = \sum_{i=1}^{4} \left( \frac{\partial m}{\partial \mathbf{p}_i} \underbrace{\frac{\partial \mathbf{p}_i}{\partial \mathbf{c}}}_{=-1} \right) = -\sum_{i=1}^{4} \frac{\partial m}{\partial \mathbf{p}_i} \tag{10}$$

Because, by definition, the horizontal frame is located in the CoM, a translation of $\mathbf{c}$ will result in a translation of $\mathbf{p}$ of equal amplitude in the opposite direction, which gives $\partial \mathbf{p} / \partial \mathbf{c} = -1$. Moreover, considering that we do not know a priori which foot affects the margin $m$ for a given motion, we need therefore to sum up the contributions of each leg.

Note that the *minimum stability margin* is the sole term in the proposed TO problem that enforces dynamic consistency. Because the MLP has already learned offline the relationship between task space variables and feasible state of the robot, it is not necessary to include here the contact forces as optimization variables or the feasibility constraints already included in Alg. 1 such as friction cones and force polytopes.

### C. Reinforcement Learning

The proposed learned stability margin can also be used to obtain a control policy trained using model-free approaches such as RL [4], [31].

Most of the existing RL approaches for the complex task of legged locomotion are sample inefficient and require significant amount of simulated runs. Therefore, it becomes necessary to implement efficient RL environment setups in order to minimize computation time of each simulation step thereby reducing the total training time. However, in order to obtain a RL policy which exhibits the desired locomotion behavior, significant platform-specific engineering is often necessary. This is usually done by tuning a dense reward function: utilizing the stability margin $m_a$ as a metric of the robot's performance in the reward function enables the use of such a dense function. Additionally, learning the stability margin estimate $m_l$ allows for the necessary reduction of the stability inference by a considerable amount. In Sec. V-C we show simulation results of how a RL policy, extensively described in [4], can benefit from the leverage of the learned stability margin.

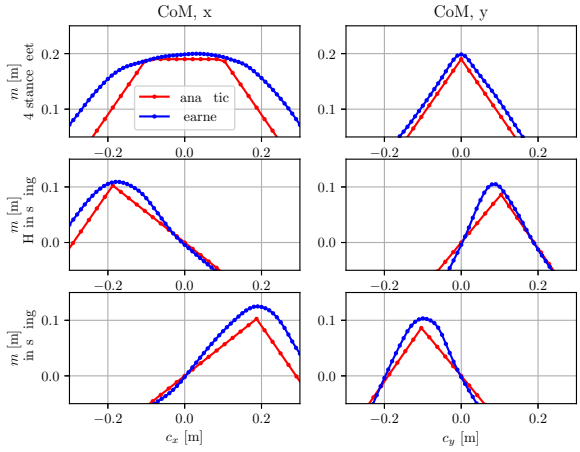



Fig. 3. Comparison of the learned stability margin $m_l$ (blue) and its analytical counterpart $m_a$ (red). Variation of the stability margin w.r.t.$x$ (left) and $y$ (right) coordinates of the CoM for a quadruple (top) and triple (middle and lower) stance configuration. The learned margin tends to divert from the analytical margin far away from the training distribution (see Tab. I) and near the discontinuities of the analytical margin's gradient (see Fig. 4), which is a necessary trade-off to achieve a smooth $m_l$ function.

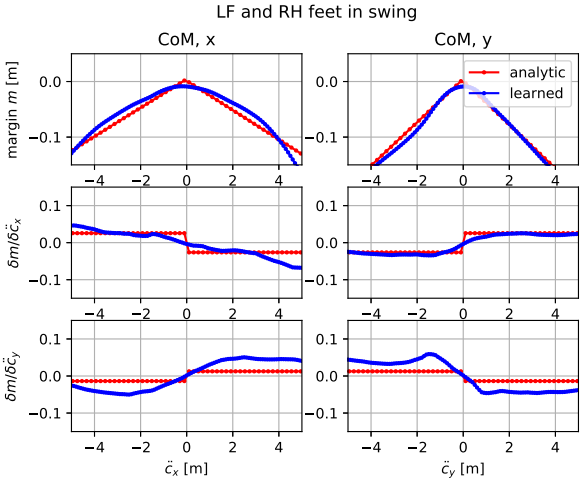



Fig. 4. Comparison of the stability margin (top) and its partial derivatives (middle and bottom): the analytical and learned partial derivatives were obtained using finite differences and backpropagation, respectively. Variation of the stability margin (top) and of its partial derivatives $\partial m / \partial \ddot{c}_x$ (middle) and $\partial m / \partial \ddot{c}_y$ (bottom) with respect to $x$ (left) and $y$ (right) coordinates of the CoM linear acceleration for a double stance configuration.

## V. Experimental Results

The learned stability margin $m_l$ can be evaluated in about $10\,\mu s$ thus offering a significant computational speed-up



Fig. 5. Snapshots of a Gazebo simulation where the ANYmal C robot walks over obstacles of 6, 12 and 18 cm while replanning its base pose, its linear and angular velocity and contact locations at 1.5 Hz adapting to the latest height map using the TO formulation presented in Sec. IV-B.

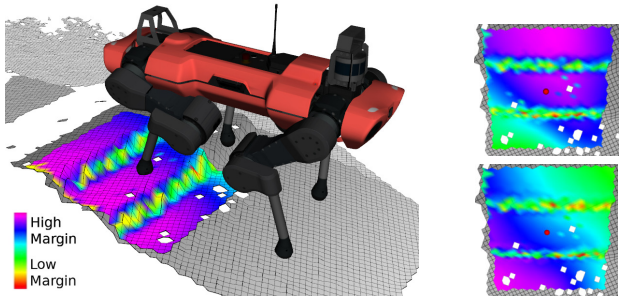



Fig. 6. Values of the learned stability margin when we vary the position of the left-front (LF) leg at different time of the walking gait cycle. Left: at full stance; top right: during swing of the right-front (RF) leg; bottom right: during swing of the left-hind (LH) leg. The prevalent purple color indicates that the motion planner has selected a good foothold as the stability margin value remains above zero throughout the whole gait cycle.

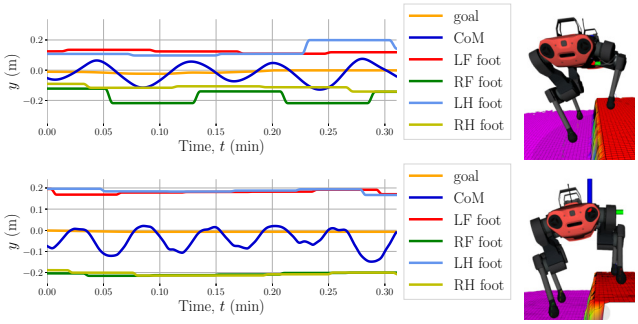



Fig. 7. Comparison of receding-horizon motion planning on a sagittal step of 18 cm using the SRB model (top) and the proposed feasible region (bottom). Using the feasible region results in wider stance configurations without needing of additional costs/reward functions and it pushes the CoM further away from the most retracted legs, in order to alleviate the load on those joints when close to their torque limits.

compared to the analytical counterpart. In the next Sections we analyse the performances of the network in terms of reliability and accuracy. We then report the experimental results obtained in simulation and on the hardware using the nonlinear TO formulation and the RL policy described above.

### A. Reliability and Accuracy

We measured reliability as the amount of true positives and true negatives that were estimated by the network when assessing the stability of the robot's state as a binary problem, considering a state to be stable when $m_l \geq 0$. We evaluated the model's predictability based on the range of samples within two times the standard deviation of the dataset generated using the parameters sampled from the distributions given in Tab. I. Over a batch of $2.5 \cdot 10^4$ samples we obtained a true positive rate of $0.970$ and a true negative rate of $0.984$.

As regards the accuracy of the network, Fig. 3 shows a comparison between the analytical stability margin $m_a$ (red) and the learned stability margin $m_l$ (blue) in different stance configurations: four stance (top), triple stance with left-hind (LH) foot in swing (middle) and triple stance with right-front (RF) foot in swing (bottom). Figure 4 shows the evolution of the analytic (red) and learned (blue) stability margins (top) and partial derivatives(middle and bottom) over a wide range of CoM linear accelerations $\ddot{c}_x$ (left) and $\ddot{c}_y$ (right). The IP algorithm does not provide the partial derivatives of the analytical margin $m_a$, which we therefore computed by first-order central difference. This requires two calls to Alg. 1 resulting in a $20.0\,\mathrm{ms}$ computation time for each partial derivative; this leads to a $940.0\,\mathrm{ms}$ computation time for the vector of partial derivatives $\partial m_a / \partial \boldsymbol{\lambda}$ with respect to the full input state $\boldsymbol{\lambda} \in \mathbb{R}^{47}$, almost four orders of magnitude more than the time required to infer the learned counterpart $\partial m_l / \partial \boldsymbol{\lambda}$ (around $100\,\mu\mathrm{s}$).

The accuracy of the $m_l$ curves shown in Fig. 4 is critical for its successful deployment in real-world applications. In particular, the derivatives with respect to the base acceleration are relevant for TO strategies. Although the accuracy error deteriorates as the state of the robot moves away from the training range (see Tab. I), we can see that the prediction error of $m_l$ is always below $1.0\,\mathrm{cm}$ for CoM accelerations within the range of: $\ddot{c}_x \in [-2.0, +2.0]\ \mathrm{m/sec^2}$ and $\ddot{c}_y \in [-2.0, +2.0]\ \mathrm{m/sec^2}$.

Fig. 1 shows the value of the learned stability margin when varying the base position. As expected, the stability margin is larger near the centre of the support polygon but the largest margin does not necessarily correspond to this point since the margin used also takes into account the joint-torque constraints. Similarly, Fig. 6 shows the value of the learned margin when changing the position of the left front foot, at different time instants of the gait cycle. Since the used stability criterion takes into account the contact normals $\mathbf{n}_i$, the positions very close to the edges of steps are already naturally avoided without need to add any cost.

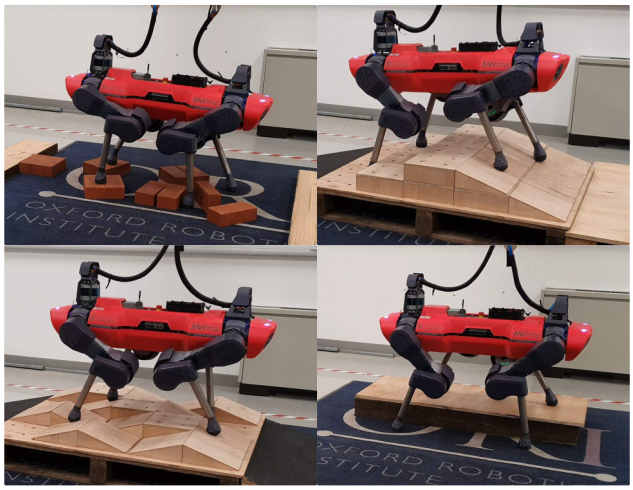


Fig. 8. Oxford's ANYmal C robot walks over bricks (top left), steps (top right), inclines (bottom left) and side steps (bottom right) using the receding-horizon nonlinear TO formulation proposed in Sec. IV-B using the learned stability margin as many feasibility criterion.

The most stable foot position change along the gait cycle, so the best contact position can be selected by computing the margin over the whole cycle. Next section shows this can be achieved through trajectory optimization.

### *B. Nonlinear Trajectory Optimization*

To demonstrate the applicability of the proposed network TO, we tested the formulation described in Section IV-B using the ANYmal C robot in simulation (Gazebo) and experimentally. We used a time horizon $T$ of $3.0\,\mathrm{s}$, a base trajectory discretization time of $\Delta T = 0.2\,\mathrm{s}$ and the gait parameters corresponding to a walk gait (with feet trajectory nodes discretized at touch down and lift-off). This resulted in an overall problem of 200 optimization variables and 500 constraints. Convergence took about 9 iterations using the Interior Point nonlinear solver IPOPT [32].

On a 4-core/8-thread Intel Xeon(R) CPU E3-1505M v6 @ $3.00\,\mathrm{GHz}$ computer this corresponded to about $70\,\mathrm{ms/iter}$. On the robot this fell to $40\,\mathrm{ms/iter}$. We ran the motion planner in a receding-horizon fashion at the frequency of $1.0\,\mathrm{Hz}$. As can be seen in Fig. 5 and 8, the motion planner enables the ANYmal C robot to adapt its body trajectory and footsteps location to the incoming terrain information from a 2.5-D height map with slopes and obstacles between 6 and $18\,\mathrm{cm}$.

The benefit of measuring the stability of the robot using the feasible region is particularly apparent in load-intensive tasks or in scenarios that require the robot to take on complex body configurations. In Fig. 7, for example, the ANYmal C is asked to walk along a side step, causing its left feet to be on a surface $18\,\mathrm{cm}$ above its right feet. In this case any simplified model, such as the Single Rigid Body (SRB) used to generate the motion shown in Fig. 7 (top), would require the CoM to be in the middle between the left and right feet, exactly as if they were all on the same flat surface. The feasible region (as in Alg. 1), instead, considers the joint-torque limitations of the robot and, for this reason, brings the CoM closer to the most extended legs which are in a more favourable configuration and therefore capable of carrying more body-weight (Fig. 7, bottom). Compared to the motion planning formulation based on the SRB model [33], the feasible region also leads the robot to walk with a wider stance without the need of any additional cost.

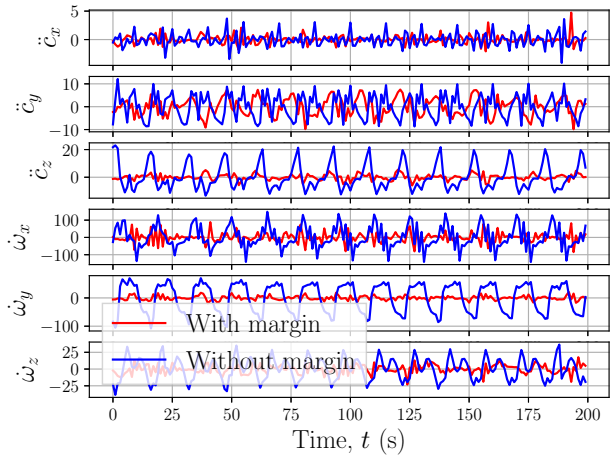



Fig. 9. Base acceleration trajectories resulting from two different RL policies for quadrupedal running trained without (blue) and with (red) a reward function maximising the learned stability margin $m_l$. This maximisation term results in smaller base accelerations.

### *C. Reinforcement Learning*

The proposed learned stability margin $m_l$ was also used in [4] to train a terrain-aware footstep planning policy, a domain adaptive tracking policy, and also an emergency recovery policy. This work proposed the use of $m_l$ as a dense reward function in order to assess the policies performance for dynamic locomotion over uneven terrain. The work further compared the influence of the stability margin on the robot's stance; a higher coefficient used with the stability margin reward term resulted in a behaviour with wider stance, whereas a smaller coefficient resulted in a preference for narrower stance during stable phases of motion while switching to wider stance during unstable phases.

In our experiments, we observed that introducing the stability metric in the reward function enabled us to obtain smoother motion trajectories. We trained two RL policies which mapped the robot state information to desired joint state of the ANYmal B robot for the task of following a reference base velocity command. We trained a RL policy using a simple reward function given by, $L(8.0 \times (0.75 - \dot{x}))$ where $L(x)$ is the logistic kernel given by $L(x) = (e^{-x} + 2 + e^{x})^{-1}$ and $\dot{x}$ is the heading velocity of the base represented in the base frame. We compared the learned behaviour of this policy with that of another RL policy obtained using a reward function comprising the previously introduced logistic kernel term, further augmented using the stability margin such that the RL policy attempts to track the desired base reference velocity while also maximizing the stability margin. As represented in the plot in Fig. 9, it is evident that the policy obtained by introducing the stability margin in the reward term results in smoother base motions

and smaller base accelerations (red) compared to the case without margin maximization (blue). Furthermore, similarly to what observed for the TO application (see Fig. 7), we observed a preference for wider stance in the case of the policy augmented with the stability margin. This is a direct result of the stance allowing for a larger support region thereby maximising the stability margin.

## VI. Conclusion and future works

In this paper, we have proposed the signed distance between the ICP and the closest edge of the feasible region [17], [1] as a measure of dynamic stability for legged robots in contact-rich scenarios and we have explained how to leverage a NN to overcome the computation time limitations of this measure. We have then reported the simulation and hardware results on the full-sized ANYmal C robot of two distinct applications in the fields of nonlinear TO and RL. In the TO setting, in particular, we have demonstrated how the learned stability margin can lead not only to the *online* concurrent planning of base trajectories and contact locations but also to the optimization of the body posture in order to avoid unnecessary violation of the robot's joint-torque limits.

We hope to advance this approach by including the learning of the joint-position limits [1] in such a way to embed into the TO and RL settings an accurate description of the body workspace.

## References


[1] A. Abdalla, M. Focchi, R. Orsolino, and C. Semini, "An Efficient Paradigm for Feasibility Guarantees in Legged Locomotion," 2019. [Online]. Available: http://arxiv.org/abs/2011.07967

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020. [Online]. Available: https://robotics.sciencemag.org/content/5/47/eabc5986

[3] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[4] S. Gangapurwala, M. Geisert, , R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-Aware Legged Locomotion using Reinforcement Learning and Optimal Control," 2020. [Online]. Available: http://arxiv.org/abs/2012.03094

[5] [Online]. Available: https://www.anybotics.com/anymal-legged-robot/

[6] T. Bretl and S. Lall, "Testing Static Equilibrium for Legged Robots," *IEEE Transactions on Robotics (TRO)*, vol. 24, no. 4, pp. 794–807, 2008. [Online]. Available: http://dx.doi.org/10.1109/TRO.2008.2001360

[7] A. Del Prete, S. Tonneau, and N. Mansard, "Fast algorithms to test robust static equilibrium for legged robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[8] N. Giftsun, A. Del Prete, and F. Lamiraux, "Robustness to Inertial Parameter Errors for Legged Robots Balancing on Level Ground," in *International Conference on Informatics in Control, Automation and Robotics (ICINCO 2017)*, Madrid, Spain, July 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01533136

[9] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 200–207.

[10] O. E. Ramos and K. Hauser, "Generalizations of the capture point to nonlinear center of mass paths and uneven terrain," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 851–858.

[11] S. Caron, A. Escande, L. Lanari, and B. Mallein, "Capturability-based Analysis, Optimization and Control of 3D Bipedal Walking," 2018. [Online]. Available: https://arxiv.org/pdf/1801.07022.pdf

[12] A. Del Prete, S. Tonneau, and N. Mansard, "Zero Step Capturability for Legged Robots in Multicontact," *IEEE Transactions on Robotics (TRO)*, vol. 34, no. 4, pp. 1021–1034, 2018.

[13] H. Dai, "Robust multi-contact dynamical motion planning using contact wrench set," Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[14] S. Caron, Q.-C. Pham, and Y. Nakamura, "Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics," in *RSS*, no. 8, 2015.

[15] R. Orsolino, M. Focchi, C. Mastalli, H.Dai, D. G. Caldwell, and C. Semini, "Application of Wrench based Feasibility Analysis to the Online Trajectory Optimization of Legged Robots," *IEEE Robotics and Automation Letters (RA-L)*, pp. 3363–3370, 2018.

[16] S. Caron, Q.-C. Pham, and Y. Nakamura, "ZMP support areas for multi-contact mobility under frictional constraints," *Transactions on Robotics (TRO)*, vol. 33, pp. 67–80, 2017.

[17] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, and C. Semini, "Feasible Region: an Actuation-Aware Extension of the Support Region," *IEEE Transactions on Robotics (TRO)*, 2020.

[18] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 9–16.

[19] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 279–286.

[20] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. TaÃ¯x, and A. Del Prete, "SL1M: Sparse L1-norm Minimization for contact planning on uneven terrain," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6604–6610.

[21] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernando-Lopez, and C. Semini, "Simultaneous Contact, Gait and Motion Planning for Robust Multi-Legged Locomotion via Mixed-Integer Convex Optimization," in *IEEE Robotics and Automation Letters (RA-L)*, 2018.

[22] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems through Phase-based End-Effector Parameterization," *IEEE Robotics and Automation Letters (RA-L)*, pp. 1560–1567, 2018.

[23] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau5872

[24] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier, and M. Taïx, "C-CROC: Continuous and Convex Resolution of Centroidal Dynamic Trajectories for Legged Robots in Multicontact Scenarios," *IEEE Transactions on Robotics (TRO)*, pp. 1–16, 2020.

[25] T. Kwon, Y. Lee, and M. Van De Panne, "Fast and Flexible Multi-legged Locomotion using Learned Centroidal Dynamics," *ACM Transactions on Graphics*, vol. 39, 07 2020.

[26] J. Viereck and L. Righetti, "Learning a Centroidal Motion Planner for Legged Locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, November 2021. [Online]. Available: https://arxiv.org/pdf/2011.02818.pdf

[27] J. Carpentier, R. Budhiraja, and N. Mansard, "Learning Feasibility Constraints for Multi-contact Locomotion of Legged Robots," in *RSS*, 2017.

[28] T. Bretl and S. Lall, "A fast and adaptive test of static equilibrium for legged robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 1109 – 1116.

[29] C. Clark, "Autonomous Robot Locomotion," 2011. [Online]. Available: https://www.cs.princeton.edu/courses/archive/fall11/cos495/COS495-Lecture3-RobotMotion.pdf

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Real-Time Trajectory Adaptation for Quadrupedal Locomotion using Deep Reinforcement Learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[32] A. Waetcher, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006. [Online]. Available: https://doi.org/10.1007/s10107-004-0559-y

[33] O. Melon, R. Orsolino, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Receding-Horizon Perceptive Trajectory Optimization for Dynamic Legged Locomotion with Learned Initialization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

# 5

# Paper III
# Real-Time Trajectory Adaptation for Quadrupedal Locomotion using Deep Reinforcement Learning

**This work was presented at the IEEE International Conference on Robotics and Automation (ICRA) 2021.**

# Manuscript Preface

This paper introduces a control architecture to perform online adaptation of motion plans generated offline using a trajectory optimization solver. This trajectory adaptation allows for addressing issues related to online inaccurate whole-body motion tracking.

The work utilizes TOWR [Winkler et al., 2018] to generate long-horizon base and feet motion plans for a quadrupedal system. TOWR employs a phase-based parameterization of feet motions wherein each phase (swing or stance) is represented by either a constant value or a sequence of cubic polynomials. During trajectory optimization, the duration of these phases is modified to allow for generation of arbitrary gait patterns. The foot placement timings are therefore generated directly through TOWR. A constraint on the total duration of each of the feet motions is also introduced into the optimization problem to ensure the feet motions end at a specified total time. In this work, we also predefine a finite number of phases for each of the feet. This depends upon the length of the trajectory. For example, the motion plans evaluated in Section V-B use 4 swing phases for each of the feet to traverse a distance of 2 m.

The work presents a neural network parameterized real-time trajectory adaptation module. This neural network corrective policy is trained using deep RL. The reward function employed in the RL environment comprises several reward terms that encourage accurate tracking of reference motion plans while exhibiting smooth and stable locomotion behavior. To address inaccurate tracking, the *trajectory tracking* reward term penalizes deviations from the reference motion plan. A *short-horizon goal* term is introduced to ensure the residual injected by the corrective policy still directs the quadruped towards the desired goal in case of external perturbations. To obtain a desirable behavior feasible for execution on the physical system, the reward function includes penalties on feet velocities and accelerations, joint velocities and accelerations, and deviations in current action with respect to the previous. This discourages aggressive behaviors.

Some of the previously mentioned reward terms were utilized in the work of *Hwangbo et al.* to obtain dynamic and robust locomotion policy for the complex ANYmal B [Hutter et al., 2016] quadruped [Hwangbo et al., 2019]. The reward function also includes penalty on *torque* utilization, *feet slippage*, and *feet clearance*. These terms reduce the overall energy consumption while discouraging dragging of feet along the ground which may result in significant wear-and-tear on the real robot. The reward expressions are provided in Table I and the RL training details are provided in Section IV of the attached manuscript.

Section V details upon the experimental results comparing open-loop trajectory tracking with the proposed trajectory adaptation approach. Table II provides the success rates observed for simulated trajectory tracking: without feedback, with blind feedback, and with perceptive feedback while traversing 3 types of terrains: flat, modular pallet, and stacked pallet. In case of blind feedback, the corrective policy only has access to proprioceptive state information and is blind to the terrain. In contrast, the perceptive policy includes a terrain representation in its observation space. These results are detailed in Section V-A. Additional hardware evaluation is presented in Section V-B.

# Real-Time Trajectory Adaptation for Quadrupedal Locomotion using Deep Reinforcement Learning

Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon and Ioannis Havoutis

***Abstract*— We present a control architecture for real-time adaptation and tracking of trajectories generated using a terrain-aware trajectory optimization solver. This approach enables us to circumvent the computationally exhaustive task of online trajectory optimization, and further introduces a control solution robust to systems modeled with approximated dynamics. We train a policy using deep reinforcement learning (RL) to introduce additive deviations to a reference trajectory in order to generate a feedback-based trajectory tracking system for a quadrupedal robot. We train this policy across a multitude of simulated terrains and ensure its generality by introducing training methods that avoid overfitting and convergence towards local optima. Additionally, in order to capture terrain information, we include a latent representation of the height maps in the observation space of the RL environment as a form of exteroceptive feedback. We test the performance of our trained policy by tracking the corrected *set points* using a model-based whole-body controller and compare it with the tracking behavior obtained without the corrective feedback in several simulation environments, and show that introducing the corrective feedback results in increase of the success rate from 72.7% to 92.4% for tracking precomputed dynamic long horizon trajectories on flat terrain and from 47.5% to 80.3% on a complex modular uneven terrain. We also show successful transfer of our training approach to the real physical system and further present cogent arguments in support of our framework.**

## I. Introduction

Legged locomotion has largely been approached using model-based control methods such as short-horizon motion planning (*e.g.,* one gait cycle ahead) which can be performed online for predefined gaits to optimize target footholds and center of mass (CoM) trajectories in a model predictive control (MPC) manner [1], [2]. Such a method enables humanoid and quadrupedal robot systems to instantly recover from unexpected disturbances or to blindly adapt to rough terrains [3]. In contrast, long-horizon nonlinear trajectory optimization can be performed efficiently to optimize for base trajectories (*i.e.,* base pose and twist), target footholds, feet trajectories, contact forces and timings [4]. Despite its minimal parameterization, the computational performance of such methods are not yet suitable for fast online motion planning. In this regard, much of the research focuses on either improving the convergence rate of this formulation by exploiting learnt initial guesses [5], [6], [7] or by introducing feasibility constraints in order to account for the dynamic limits of the actual hardware [8]. In both these cases, the successful execution of the pre-determined motion plan relies on a whole-body controller (WBC) used to compute the feed-forward joint torques, target joint velocities and target joint positions in order to track the generated reference trajectories online.

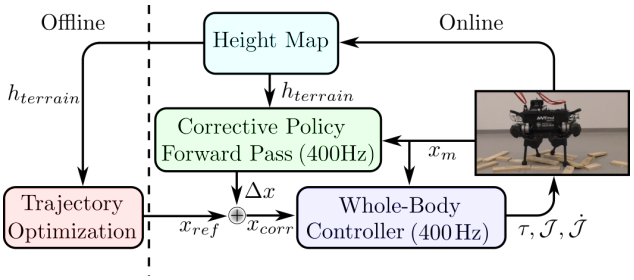



Fig. 1. Block diagram of the trajectory generation and tracking framework. Experiments were performed using the ANYmal B quadruped. The accompanying video can be found here: https://youtu.be/Ve4SDl1wI9s.

While long-horizon approaches have shown impressive results in controlled environments [4], their usage outdoors has been very limited mainly due to the slow response time associated with these methods making them unreliable in presence of unexpected disturbances, perturbations and inaccurate whole-body tracking.

Recent work on learning based approaches for quadrupedal locomotion [9], [10], [11] has shown great promise for development of robust and dynamic model-free data-driven control techniques which directly map sensory information into desired robot states enabling extremely fast control response. RL, in particular, has been further employed for locomotion over uneven terrain [12], [13] demonstrating impressive locomotion behaviour. However, in order to be used in safety-critical environments, the RL training setup necessitates constrained exploration and control behaviour [14] which requires significant platform-specific engineering thereby increasing the complexity of the RL training environment.

In this work, we consider the problem of tracking dynamic long horizon motion plans generated using a trajectory optimization solver, TOWR [4]. Unlike MPC-based short horizon motion plans which can be recomputed online to recover from perturbations, long horizon motion plans are computationally expensive for online replanning and require precise whole-body tracking. As a result, the performance of controllers employing such long horizon motion plans is often lower than that of MPC-based controllers.

This work was supported by the UKRI/EPSRC RAIN Hub [EP/R026084/1] and the EU H2020 Projects MEMMO and THING, the EPSRC grant 'Robust Legged Locomotion' [EP/S002383/1] and a Royal Society University Research Fellowship (Fallon). This work was conducted as part of ANYmal Research, a community to advance legged robotics. The authors are with Dynamic Robots Systems (DRS) group, Oxford Robotics Institute, University of Oxford, UK. Email: {siddhant, mathieu, rorsolino, mfallon, ioannis}@robots.ox.ac.uk

In this regard we present an approach to improve the performance of tracking long horizon motion plans by introducing a new control architecture which utilizes long horizon motion plans generated using TOWR, corrected online using an RL policy and then tracked by a model-based WBC. The WBC ensures that the necessary safety-critical constraints such as joint kinematic limits and peak joint torques are enforced during operation. The corrective RL policy generates additive deviations to the reference trajectories online with the objective of prioritizing stability over tracking extremely dynamic (and possibly infeasible) trajectories in addition to directing the robot towards the desired long horizon goal. We evaluate the performance of our control architecture using the ANYmal [15] quadruped, both, in simulation and on the physical robot.

## II. Related Work

A significant amount of research in RL and optimal control (OC) domains has focused on tackling shared objectives using a unified approach [16], [17] including contributions in the form of model-based RL [18], guided policy search (GPS) [19], [20] and even a reformulation of the stochastic optimal control problem in terms of KL divergence minimization [16].

There has also been work which directly builds on the baseline controller in order to learn a corrective control behavior. These corrective controllers have been employed for manipulation tasks [21] and also for legged locomotion [22]. In the case of [22], a model-based controller is implemented to efficiently solve the rigid body dynamics while the model-free data-driven controller is used to capture properties such as contacts and friction which are difficult to model.

Additionally, quasi-static control approaches such as model-based FreeGait [23] have been proposed for terrain-aware locomotion. In contrast, the trajectory optimization solver, TOWR, used in our work generates dynamic whole-body motion plans. Furthermore, as opposed to recent work on RL for dynamic locomotion over challenging [13] and uneven terrain [24] which focus on short horizon planning and control, in this work, we address the issues with tracking dynamic long horizon plans by performing online trajectory adaptation while tracking reference trajectories generated offline.

## III. Overview

We use a trajectory optimization technique to generate long-horizon base and end-effector task space motion plans. Then, during the online execution of such trajectories, we use the feedback generated using the corrective RL policy for a history of tracking errors to modify the base and end-effector motion plans which are tracked using a WBC. The WBC finally generates the desired joint position, velocity and feedforward torque commands which are then tracked by the actuators using a PD controller. This motion generation scheme is illustrated in Fig. 1.

### A. Trajectory Optimization

In this work, we utilize the trajectory optimizaztion solver TOWR [4] together with the smoothing costs described in [6] to generate base and end-effector (feet) motion plans. The TOWR framework formulates the locomotion problem of legged robots as a non-linear optimization problem which can generate dynamic trajectories for locomotion over complex 3D terrains. The problem is discretized into a numerically-solvable formulation using a direct-collocation transcription method which is then solved using an interior point approach [25].

To stabilize the behavior of the optimizer on uneven terrains, we consider the friction cone axis aligns with the gravity axis. This limits TOWR, which uses a local optimization approach, from considerable divergence during optimization around regions with high deviations in terrain slope.

### B. Whole Body Control

In order to track the motion plan *set points*, we use the whole-body controller based on the work of [3] which employs a hierarchical optimization approach to compute the feed-forward joint torques. It is important to note that, while the trajectory optimization computes contact forces for generating the trajectories, in our setup, these forces are only used to enforce a feasible motion at the trajectory generation stage and are not used by the whole-body tracking controller. Instead, the WBC takes the feet and base trajectories as inputs and recomputes the contact forces using the full dynamics model of the robot.

To improve the robustness during online motion plan tracking, we use the approach detailed in [26] to adapt the PD gains of the actuators and the friction coefficient used in the whole-body controller for a limb whose end-effector slips while in contact with the ground.

The WBC forwards the desired joint states and the feed-forward joint torque in addition to the PD gains (dependent on the foot contact states: support, swing or slip) to the actuator for low-level joint state tracking.

### C. Online Trajectory Correction

We represent the online trajectory adaptation problem in the framework of a discrete time stochastic Markov decision process (MDP) defined as a tuple $(S, A, R, P, \mu)$, where $S$ represents a set of states, $A$ a set of actions, $R : S \times A \times A \to \mathbb{R}$ the reward function, $P : S \times A \times S \to [0,1]$ the state transition probability, and $\mu$ the initial state distribution. We define a stationary policy $\pi : S \to \mathcal{P}(A)$ which, in our work, is approximated using a multi-layer perceptron (MLP), as a function mapping states to probability distributions over actions such that $\pi(a|s)$ denotes the probability of selecting action $a$ in state $s$. We represent the expected cumulative discounted return,

$$J(\pi) \doteq \mathop{\mathrm{E}}_{\mathcal{T}\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right], \tag{1}$$

where $\gamma \in [0,1)$ is the discount factor and $\mathcal{T}$ denotes a trajectory dependent on $\pi$. We employ a policy gradient based method, proximal policy optimization (PPO) [27] along with generalized advantage estimate (GAE) [28] so as to obtain a policy $\pi$ which maximizes $J(\pi)$.

As detailed in Section IV, we use the policy $\pi$ to map a set of robotic system state parameters to actions representing the desired corrective deviations to the reference trajectory at each time step. The modified set point is then tracked by the model-based whole-body controller. Both, the corrective feedback and the whole body control output are computed sequentially at 400 Hz.

## IV. Training

This section details upon the RL environment setup for training the corrective-tracking policy.

### A. Action Space

The corrective policy outputs a 36-dimensional action vector comprising of $\{\delta P_{base}, \delta \mathcal{O}_{base}, \delta v_{base}, \delta \omega_{base}, \delta P_{feet}, \delta v_{feet}\}$ where the $\{\delta P_{base}\} \in \mathbb{R}^3$ vector represents the deviation in base position, $\{\delta \mathcal{O}_{base}\} \in \mathbb{R}^3$ is the deviation in base orientation, $\{\delta v_{base}\} \in \mathbb{R}^3$ is the deviation in linear base velocity, $\{\delta \omega_{base}\} \in \mathbb{R}^3$ is the deviation in angular base velocity, $\{\delta P_{feet}\} \in \mathbb{R}^{12}$ represents the deviation in the 4 end-effector positions and $\{\delta v_{feet}\} \in \mathbb{R}^{12}$ represents the deviation in the 4 end-effector velocities. At each control step, we introduce these additive deviations to the desired set point obtained from the reference trajectory generated offline.

### B. Observation Space

The observation space in our training environment comprises of the proprioceptive sensory information which can be computed by sensors and state-estimators on the physical system. For terrain-aware corrective trajectory tracking, we also utilize an exteroceptive feedback in the form of an encoded representation of the terrain elevation.

*1) Proprioceptive Sensory Information:* The proprioceptive state information consists of a 486-dimensional vector defined as

$$\{\mathcal{C}^{m}_{t,t-4,t-8}, \mathcal{V}^{m}_{t,t-4,t-8}, \mathcal{C}^{traj}_{t+8,t+4,t+1,t,t-4,t-8}, \mathcal{V}^{traj}_{t+8,t+4,t+1,t,t-4,t-8}, \mathcal{C}^{corr}_{t-1,t-4,t-8}, \mathcal{V}^{corr}_{t-1,t-4,t-8}, \mathcal{C}^{traj}_{t+200}\}$$

where $\mathcal{C}_t$ refers to the generalized coordinates at time $t$, $\mathcal{V}_t$ refers to the generalized velocities at time $t$ and the superscripts $m$, $traj$ and $corr$ refer to the measured robot state, reference trajectory set point and the corrected trajectory set point respectively. The delay between $t+1$ and $t$ corresponds to 2.5 ms. All of the state parameters are represented in the robot's base frame. We introduce the short-horizon goal $\mathcal{C}^{traj}_{t+200}$ in order for the corrective policy to generate outputs which prioritize stable tracking of long-term trajectory plans as opposed to aggressive tracking of the reference trajectory.

*2) Exteroceptive State Representation:* We introduce latent encoding of the height maps in the state vector $s$. These height maps represent the terrain elevation local to the robot's base. We use $80 \times 80$-dimensional height maps, along the robot's heading and lateral axes respectively, corresponding to an area of $1.0 \times 1.0$ m$^2$ centered at the robot's base position. The elevation of the height maps is clipped between $[-2.0, 2.0]$ m.

TABLE I

Reward terms used in our MDP formulation. Here $\tau$ refers to the joint torque, $v^{foot}_{world,t}$ is the foot velocity in world frame at time $t$, $F_{foot}$ is the foot contact force, $\mathcal{J}_t$ is the joint position at time $t$, $\mathcal{O}^{base}_{x,y,z}$ is the base orientation along the $x,y,z$ axes, $f_{h,foot}$ represents the foot height, $f^{des}_h$ represents the desired foot clearance, $P^{m}_{base}$ represents the measured base position, $P^{traj}_{base}$ is the desired base position obtained from the reference trajectory and $P^{traj,H}_{base,t}$ represents the desired base position at the short-horizon (in our case at $t+200$) in the robot's base frame at time $t$.

| Term | Expression |
|---|---|
| Torque | $\|\tau\|^2$ |
| Foot Acceleration | $\|\|v^{foot}_{world,t} - v^{foot}_{world,t-1}\|\|^2$ |
| Foot Slip | $\|\|v^{foot}_{world}\|\|^2 \quad \forall\ F_{foot} > 0$ |
| Smoothness | $\|\|\mathcal{J}_t - \mathcal{J}_{t-1}\|\|^2$ |
| Orientation | $\|\|\mathcal{O}^{base}_{x,y,z} - \{0,0,\mathcal{O}^{base}_{z}\}\|\|^2$ |
| Joint Velocity | $\|\dot{\mathcal{J}}_t\|^2$ |
| Joint Acceleration | $\|\ddot{\mathcal{J}}_t\|^2$ |
| Foot Clearance | $\sum_{foot} \left(f_{h,foot} - f^{des}_h\right)^2 \left\|v^{foot}_{world}\right\|^2$ |
| Trajectory Tracking | $\|P^{m}_{base} - P^{traj}_{base}\|^2$ |
| Short Horizon Goal | $\sum\{P^{traj,H}_{base,t-1} - P^{traj,H}_{base,t}\}$ |

In order to perform sample-efficient RL [29] we reduce the dimensionality of our sparse height maps using a convolutional autoencoder network [30] to learn an encoding $h = f(x)$ such that the decoder, approximated as $g$, is able to regenerate the input $x \approx g(h)$. We then use the encoder network $f$ to encode our $80 \times 80$-dimensional height maps into a 60-dimensional representation.

The encoder comprises of 3 convolutional layers each with a kernel size of 3, and a depth mapping from 1 to 16 in the first layer, 16 to 24 in the next and 24 to 32 in the last layer. We append a 60-dimensional dense layer to output the encoded representation of the height maps. For training, we use a decoder network comprising of the deconvolutional layers with reversed depth mapping as that of the encoder convolutional layers to regenerate the original height map from the encoded vector. We train the auto-encoder to minimize the regeneration error using a dataset of procedurally generated height maps.

### C. Policy Network

For the combined proprioceptive and exteroceptive state representation, our network input is 546-dimensional, and the output is 36-dimensional. We use 3 hidden layers comprising of 1024, 584 and 256 nodes in the first, second and third layer respectively. Furthermore, we use the hyperbolic tangent non-linear activation for our policy network.

### D. Reward Signal

The reward terms used for training the corrective policy are shown in Table I (the notation is consistent with [14]).

One of the critical aspects of our task setup is to ensure the robot remains dynamically stable throughout the trajectory

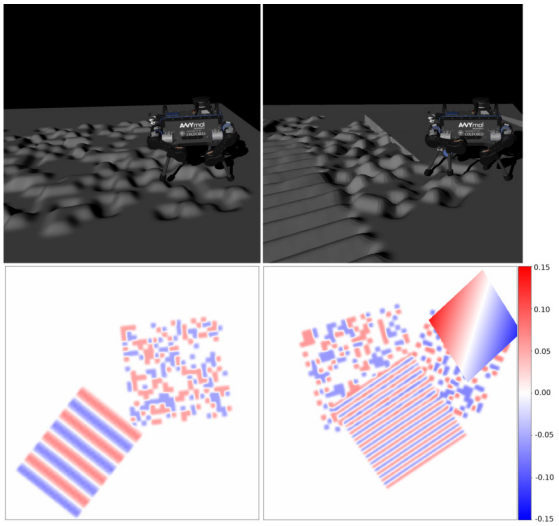


Fig. 2. *top:* Terrain visualization in the RaiSim simulator. *bottom:* The corresponding 16-bit PNG images representing the height maps. The colorbar represents the terrain elevation in m.

tracking duration. Despite significant research that has been performed to handle sparse rewards [31] in the RL setup, it is still desirable to have a dense reward curve. In this regard, we utilize a stability analysis method, detailed below, to generate a reward signal. To assess the stability of the robot we employ a criterion based on the feasible region [32] which depends on the endogenous and exogenous properties of the robot, and also on environmental conditions such as the friction coefficient and the surface normals.

We define the stability margin $m \in \mathbb{R}$ as the signed distance between the instantaneous capture point (ICP) [33] and the edges of the feasible region. The stability margin $m$ is positive when the ICP lies within the feasible region and negative if outside. We define the robot to be dynamically stable whenever $m > 0$. We thus introduce a reward term given by $\max(m, m_{max})$ so as to maximize the stability margin. For ANYmal, we used $m_{max} = 0.13$.

### *E. Rough Terrain Simulation*

We use the RaiSim [34] physics simulator to train our RL policy. In order to obtain a robust policy for terrain-aware reference trajectory tracking we train our RL policy across a multitude of procedurally generated terrains. We randomize our simulation environment and perform trajectory optimization over multiple terrains. This enables us to ensure that the trained RL policy generalizes over multiple terrains for different reference trajectories. For training our RL policy we used a total of 1000 different simulated terrains. We generate these terrains using 2-dimensional height maps as detailed in [24]. Examples of these height maps and the corresponding simulated terrains are represented in Fig. 2.

### *F. Policy Transfer to the Physical System*

In order to make the trained policy robust and generalizable to unaccountable factors we introduce several domain randomization methods during training. These are described below.

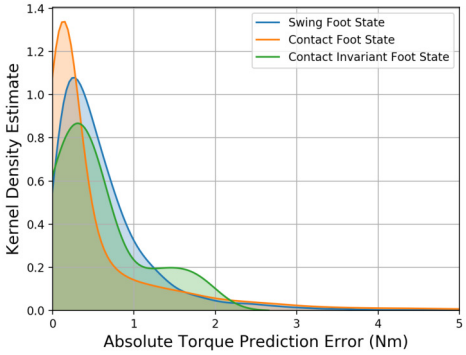



Fig. 3. The kernel density estimate plot representing the torque prediction error for different foot contact states.

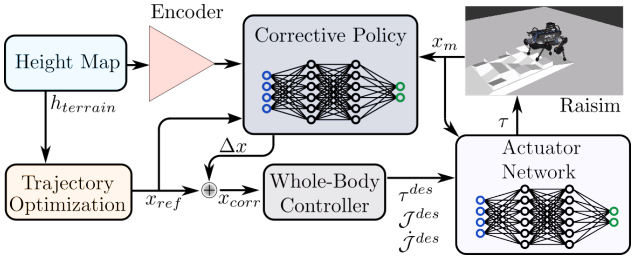



Fig. 4. Block diagram representing the modules that constitute the RL training environment of the corrective policy.

*1) Actuator Modelling:* As introduced in [9], we train an actuator network to model the actuation dynamics of the physical system. However, unlike in the case of [9], [14] where the actuators were modelled only for joint position targets with a predefined set of PD gains, in this work, we trained the actuator network using supervised learning to take as inputs the joint position and velocity targets along with the desired feed-forward torques. Since the whole-body controller utilizes different set of PD gains for joints of legs with foot in contact, swing and contact-invariant (slip) states, we further include the foot contact state information as an input to the approximated actuator model. The actuator network consists of an 18-dimensional input given by the vector

$$\{\delta\mathcal{J}^i_{t,t-4,t-8}, \delta\dot{\mathcal{J}}^i_{t-1,t-5,t-9}, \delta\tau^i_{t-1,t-5,t-9}, \dot{\mathcal{J}}^{des,i}_{t,t-4,t-8}, \tau^{des,i}_{t,t-4,t-8}, F^{state,f}_{t,t-4,t-8}\}$$

where $\delta\mathcal{J}^i_t$ represents the error between the measured and desired joint position for joint $i$ at time $t$, $\delta\tau^i_t$ represents the error between the measured and desired feed-forward joint torque for joint $i$ at time $t$, $\dot{\mathcal{J}}^{des,i}_t$ represents the desired joint velocity for joint $i$ at time $t$, $\tau^{des,i}_t$ represents the desired feed-forward joint torque for joint $i$ at time $t$ and $F^{state}_t$ is the contact state of foot $f$ at time $t$. The duration between $t+1$ and $t$ corresponds to 2.5 ms. The network contains 2 hidden layers with 48 nodes in each of the layers and a 1-dimensional output layer representing the estimate of the joint torques observed on the physical system.

TABLE II

SUCCESS RATES OBSERVED FOR TRACKING REFERENCE TRAJECTORIES WITH AND WITHOUT FEEDBACK FOR 1000 RUNS EACH OVER 3 DIFFERENT TERRAINS. THE INITIAL BASE POSITION ALONG THE HEADING AND LATERAL AXES IS RANDOMLY SHIFTED BY A MEAN OF 0 m AND A STANDARD DEVIATION OF 0.085 m

| | No Feedback | Blind Feedback | Perceptive Feedback |
|---|---|---|---|
| **Flat** | 72.7 | **94.6** | 92.4 |
| **Modular Pallet** | 47.5 | 59.3 | **80.3** |
| **Stacked Pallet** | 38.8 | 46.2 | **67.1** |

*2) Shifting Initial Position:* Upon environment resets, we randomly shift the base position along the horizontal axes with a mean of 0 m and a standard deviation of 0.03 m.

*3) Changing Gravity:* We uniformly sample acceleration due to gravity between $[0.95g, 1.05g]$, where $g = 9.81\,\mathrm{m{\cdot}s^{-2}}$ to emulate inertial scaling.

*4) Actuator Torque Scaling:* We randomly scale the output torque of our actuator network with the scaling coefficient $s_t \in [0.95, 1.05]$ to account for differences between the real actuators and the approximated model.

*5) Perturbing the Robot Base:* We apply external forces to the robot's base along its heading and lateral axes during trajectory tracking. This enables the policy to learn a recovery behavior so as to maximize the robot stability margin.

We sample the external forces from a normal distribution with a mean 0 N and a standard deviation of 15 N. We apply these forces for duration sampled randomly between $[1, 3]$ s.

## V. RESULTS

Training the terrain-aware corrective policy required 3.5B simulation steps with computation of 29.82M steps per hour on a standard desktop computer. Additionally, we trained a blind corrective policy for which we do not include the exteroceptive feedback in the observation space in less than 250M simulation steps for locomotion over flat ground. We tested its performance in simulation and on the physical system.

### A. Simulation results

For evaluation of the corrective policy we compare the locomotion behaviors obtained by the baseline motion generation strategy (offline trajectory optimization with WBC) with the same control architecture enhanced by our proposed online corrective RL policy. Table II represents the success rate, defined as the percentage of experiments that lead the robot to perform the whole trajectory without eventually falling down, observed over 1000 runs each for tracking pre-computed reference trajectories over 3 different terrains - flat, modular pallet and stacked pallet. In this experiment, we randomly shifted the initial base position of the robot along the heading and lateral axes by a mean of 0 m and a standard deviation of 0.08 m. The run was considered a success if the robot managed to reach the trajectory horizon with a base tracking error of less than 0.05 m along the horizontal axes and remained at the goal for 3 seconds without resulting in termination. We considered the run a failure if any contact-pairs apart from the feet and ground were detected. In case of flat terrain, we observed that most of the failures occurred during high base-position tracking error phases where the WBC aggressively attempted to correct the base position tracking error without recomputing plans for feet motions. For the case of the corrective policy, we observed trajectory tracking with extended feet position directly corresponding to a higher stability margin. This behaviour is evident from Fig. 7 which represents the 2nd order regressive approximation of the modified trajectory and measured feet position along the x and y axis of the world frame for 1000 runs of the robot tracking a 1.5 m long pre-computed trajectory with actuation torque output randomly scaled between $[0.975, 1.02]$.

For the case of locomotion over modular pallet, we observed failures mainly due to foot slip. The addition of the corrective policy resulted in a recovery behaviour which we also observed during tests on the physical robot as shown in Fig. 5. In the case of the stacked pallet problem, we observed that the physical dimension of the robot was a limiting factor.

We further conducted experiments to test the ability of the corrective policy to perform state recovery. We used an offline pre-computed trajectory 1000 times over different terrains. The terrains were generated as an undulated environment where each bump and gap (see Fig. 2) has a different height sampled from a random uniform distribution in the range $[-h_{max}, h_{max}]$ where $h_{max}$ represents the maximum terrain height. In all cases the WBC is required to track a reference base and feet trajectory corresponding to 16 steps, for a total distance of 1.5 m and a duration of 3.5 s. This motion plan is optimized offline for flat ground and is therefore not aware of the undulated terrain. Fig. 6 (left) represents the success rate observed for $h_{max}$ ranging between $[0.0, 0.05]$ m.

Figure 6 (middle) represents the success-rate obtained during the execution of a predefined trajectory of 3.5 s on flat terrain while perturbing the robot with a horizontal external force for a duration of sampled in the range $[1.5, 2.5]$ s. The intensity of this force is gradually increased with a step of 5 N in the range of $[0, 25]$ N. Also the start time of application of the external disturbance since the beginning of the run was randomly sampled in the range $[1.75, 2.25]$ s. Each success rate was computed using 1000 runs. The blind corrective policy performed better than the perceptive policy. This was largely due to the blind policy having significantly more experience to disturbances over flat terrain during training.

Figure 6 (right) shows the success rate obtained when we scaled the actuation torque generated by the actuator network by a scaling factor between $[0.85, 1.15]$ while tracking a pre-computed reference trajectory on flat-terrain for 1000 runs each. The effective joint torque was limited between $[-40, 40]$N·m and the initial robot base position was randomly shifted along the heading and lateral axes by a mean of 0 m and a standard deviation of 0.08 m. We observed that reducing the joint torque results in WBC failing to effectively track the desired set points whereas increasing the torque results in aggressive tracking and overshooting eventually resulting in failure.

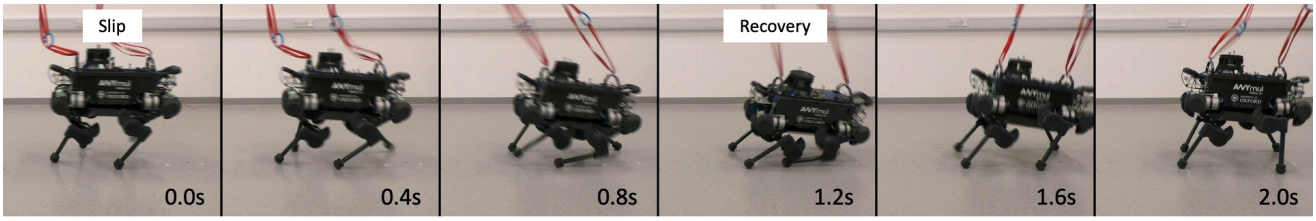



Fig. 5. Time frame of the robot recovering with the corrective controller.

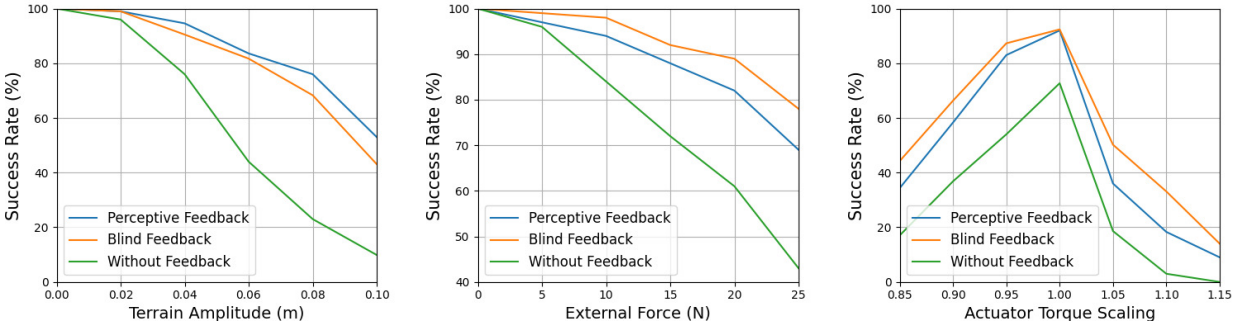



Fig. 6. ***left:*** Success-rate observed for tracking pre-computed trajectory on undulated terrains with varying height for 1000 runs each. ***middle:*** Success rate obtained for locomotion on flat ground in presence of unexpected external forces. ***right:*** Success rate observed for tracking reference trajectory on flat ground for different actuation torque scaling.

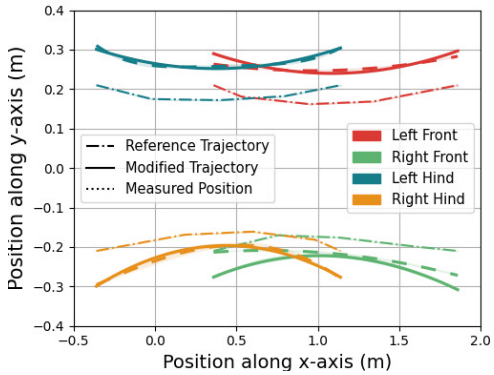



Fig. 7. Comparison of the feet position computed using a trajectory optimization solver with the 2nd order regressive approximation of the trajectory modified online by a corrective RL policy.

### *B. Hardware results*

We transferred the blind policy to the ANYmal robot and tested its performance for tracking a pre-computed reference trajectory of 3.5 s on flat ground. The reference trajectory used for this first test is highly dynamic wherein the robot travels 2.0 m in 3.5 s and achieves a maximum velocity of $0.7\,\mathrm{m{\cdot}s^{-1}}$ with significant acceleration and deceleration along the motion plan. For the case without feedback tracking, we observed a success rate of 33.33% over 15 trials. With the introduction of blind feedback, we observed a success rate of 60.0%. Note that the poor performance associated with tracking dynamic long horizon motion plans is due to accumulation of tracking errors which the whole-body controller then aggressively attempts to correct. For the case of less dynamic trajectory tracking wherein the robot travels 1.5 m in 3.5 s, we observed a success rate of 80.0% and 100.0% for open-loop and feedback-based tracking respectively. We further tested the performance of our policy to recover from an undesirable state by introducing perturbations. We tested the ANYmal robot walking over wooden tiles using the corrective policy. In this experiment, we observed that without feedback, the WBC failed to track the reference trajectory for each of the 5 trials. In comparison, for corrective tracking, we observed a success rate of 40% with most failures occurring due to foot slippage. We further introduced an unexpected wooden block along the robot's path and observed no success without feedback. We observed a success rate of 66.67% using the RL policy with most of the failures occurring due leg blockage. Furthermore, Fig. 5 presents an example of a trial that almost fails due to foot slippage but eventually succeeds to recover.

## VI. Conclusion and Future Work

We presented a control architecture to perform online corrective trajectory tracking. We observed a higher success rate for the case of tracking trajectories with feedback in both simulation and on the real robot. Despite having been tested with motion plans generated offline, our control architecture can be extended for use with motion planners which optimize trajectories online in an MPC manner. Additionally, we observed that executing the policy on the real robot requires a WBC which is robust to actuation dynamics. Therefore, our future work includes training an RL policy which also outputs actuator tracking gains. We also aim to obtain a perceptive policy which functions with asynchronous sensory feedback for execution on the physical system.

## REFERENCES


[1] G. Bledt and S. Kim, "Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 6316–6323.

[2] O. Villarreal, V. Barasuol, P. Wensing, and C. Semini, "MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion," in *Proc. IEEE International Conference on Robotics Automation (ICRA)*, Paris, France, May 2020.

[3] C. Dario Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3359–3365.

[4] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, July 2018.

[5] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2986–2993.

[6] O. Melon, M. Geisert, D. A. Surovik, I. Havoutis, and M. Fallon, "Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations," 2019.

[7] D. Surovik, O. Melon, M. Geisert, M. Fallon, and I. Havoutis, "Learning an Expert Skill-Space for Replanning Dynamic Quadruped Locomotion over Obstacles," in *2020 Conference on Robot Learning (CoRL)*, Nov. 2020.

[8] A. Bratta, R. Orsolino, M. Focchi, , V. Barasuol, G. G. Muscolo, and C. Semini, "On the Hardware Feasibility of Nonlinear Trajectory Optimization for Legged Locomotion based on a Simplified Dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[9] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[10] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

[11] A. L. Mitchell, M. Engelcke, O. P. Jones, D. Surovik, S. Gangapurwala, O. Melon, I. Havoutis, and I. Posner, "First steps: Latent-space control with semantic constraints for quadruped locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5343–5350.

[12] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[13] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020. [Online]. Available: https://robotics.sciencemag.org/content/5/47/eabc5986

[14] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *arXiv preprint arXiv:2002.09676*, 2020.

[15] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 38–44.

[16] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[17] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems Magazine*, vol. 32, no. 6, pp. 76–105, 2012.

[18] K. Doya, K. Samejima, K.-i. Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," *Neural computation*, vol. 14, no. 6, pp. 1347–1369, 2002.

[19] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1–9. [Online]. Available: http://proceedings.mlr.press/v28/levine13.html

[20] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 528–535.

[21] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 6023–6029.

[22] Z. Xie, G. Berseth, P. Clary, J. W. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," *CoRR*, vol. abs/1803.05580, 2018. [Online]. Available: http://arxiv.org/abs/1803.05580

[23] P. Fankhauser, C. Dario Bellicoso, C. Gehring, R. Dubé, A. Gawel, and M. Hutter, "Free gait — an architecture for the versatile control of legged robots," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 1052–1058.

[24] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," 2020.

[25] A. Wächter and L. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, 01 2004.

[26] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, "Dynamic locomotion on slippery ground," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4170–4176, Oct 2019.

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

[28] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015.

[29] S. S. Du, A. Krishnamurthy, N. Jiang, A. Agarwal, M. Dudík, and J. Langford, "Provably efficient rl with rich observations via latent state decoding," *arXiv preprint arXiv:1901.09018*, 2019.

[30] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[31] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," *arXiv preprint arXiv:1610.04286*, 2016.

[32] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, and C. Semini, "Feasible Region: an Actuation-Aware Extension of the Support Region," *IEEE Transactions on Robotics (TRO)*, 2020.

[33] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture Point: A Step toward Humanoid Push Recovery," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, Dec 2006, pp. 200–207.

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

# 6

# Paper IV
# RLOC: Terrain-Aware Quadrupedal Locomotion using Reinforcement Learning and Optimal Control

# Manuscript Preface

This paper introduces a modular reinforcement learning and optimal control based terrain-aware quadrupedal locomotion framework. The framework comprises terrain elevation encoder, footstep planner, motion controller, domain adaptive tracker, and recovery controller. The proposed control architecture, as illustrated in Figure 3, is also demonstrated to perform zero-shot transfer to a domain described by quadruped not introduced during training. This transferability is evaluated in Section V-B. As shown in Figure 11, the transfer robustness depends upon how dynamic the motions of the quadruped are. This relates to higher tracking errors for larger velocity commands. Table VIII further highlights the effectiveness of introducing dynamics randomization during training of the footstep planning policy and the domain adaptive tracking policy. The proposed training approach enables better transfer performance and has been detailed in Section V-B and Section V-D.

With regards to traversal over uneven terrain, the footstep planner utilizes the embedded robo-centric terrain elevation and robot state information to generate desired footholds. These feet positions are then tracked by a motion controller [Bellicoso et al., 2018b] which plans the base and feet motions, and further generates low-level actuation commands. The feet contact scheduler is directly incorporated in the motion controller and is dependent upon the user-defined gait pattern. This scheduling has been detailed in Section III-B. While terrain-adaptive phase (swing or stance) duration is more suitable to perform complex maneuvers over uneven terrain, this remains part of future work which investigates generation of arbitrary gait patterns online. This is similar to the work presented in Chapter 5.

Since the proposed modular framework comprises several components, in preliminary experiments it was observed that training them together in a multi-agent RL formulation required significant training time.

The multi-agent RL formulation was also unsuitable for training on a normal desktop PC, as used in this work, often resulting in out-of-memory issues. This allowed for execution of only few parallel training environments which additionally resulted in convergence towards local optimum wherein the control framework was unable to locomote over uneven terrain. The work therefore investigates individual training of each of these components. However, as part of future work, fine-tuning these components after they have been individually trained will also be investigated. This will be done in a multi-agent RL formulation.

The experimental evaluation detailed in Section V shows dynamic terrain-aware locomotion capability. Figure 14 presents the ability of the control framework to track velocity commands of up to $0.5\,\mathrm{m\,s^{-1}}$ over uneven terrain. This has been further pushed to $0.65\,\mathrm{m\,s^{-1}}$ in additional evaluation on the physical systems. Detailed analysis on the proposed framework and ablation studies are provided in Section V.

# RLOC: Terrain-Aware Legged Locomotion using Reinforcement Learning and Optimal Control

Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon and Ioannis Havoutis

***Abstract*—We present a unified model-based and data-driven approach for quadrupedal planning and control to achieve dynamic locomotion over uneven terrain. We utilize on-board proprioceptive and exteroceptive feedback to map sensory information and desired base velocity commands into footstep plans using a reinforcement learning (RL) policy trained in simulation over a wide range of procedurally generated terrains. When ran online, the system tracks the generated footstep plans using a model-based controller. We evaluate the robustness of our method over a wide variety of complex terrains. It exhibits behaviors which prioritize stability over aggressive locomotion. Additionally, we introduce two ancillary RL policies for corrective whole-body motion tracking and recovery control. These policies account for changes in physical parameters and external perturbations. We train and evaluate our framework on a complex quadrupedal system, ANYmal version B, and demonstrate transferability to a larger and heavier robot, ANYmal C, without requiring retraining.**

***Index Terms*—Deep Learning in Robotics and Automation, AI-Based Methods, Legged Robots, Robust/Adaptive Control of Robotic Systems**

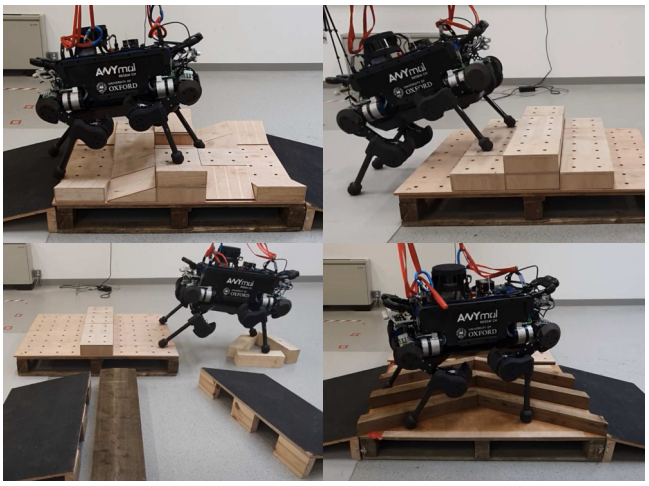


Fig. 1: ANYmal B quadruped used to perform experiments on different terrains using our RLOC framework. Overview video can be found at: https://youtu.be/EAt0QjuO2cE.

## I. Introduction

Legged robot locomotion research has been driven by the goal of developing robust and efficient control solutions with the potential to exhibit a level of agility that matches their biological counterparts. Their advanced mobility is characterized by the ability to form support contacts with the environment at discontinuous locations as required. This makes such systems suitable for traversal over terrains that are inaccessible for wheeled or tracked robots of equivalent size. This maneuverability, however, comes at the expense of increased control complexity. Such systems often need to maintain dynamic stability, select appropriate contact sequences, and actively balance during locomotion.

In this work, we design, train and evaluate a control framework for legged locomotion over uneven terrain using the *ANYbotics ANYmal B [1]* and *ANYmal C* [2] quadrupedal robots. ANYmal version B is a 30 kg rugged robot developed with a focus on high mobility and the ability to perform dynamic locomotion behaviors for rough terrain operations. It features a large workspace which enables a wide range of leg motions and is suitable for a broad domain of applications, for example, inspection tasks on offshore platforms [3] and sewer systems [4]. Figure 1 shows the ANYmal B robot used for testing our proposed framework on different uneven terrains. We additionally highlight the generality of our control framework by testing our approach with the newer ANYmal, version C [2] –a 50 kg quadrupedal robot with a similar kinematic structure but different dynamic characteristics– and by evaluating its performance *without retraining* any of our planning and control policies. Experiments performed with ANYmal C are shown in Fig. 2.

The control of such underactuated robotic systems is an active area of research.

Approaches based on whole-body trajectory optimization such as contact-implicit optimization (CIO) [5] can generate complex multi-contact motions but often remain too computationally extensive to be used online. More recent works based on differential dynamic programming (DDP) algorithms [6], [7] enable efficient computation of whole-body motions suitable for online execution. These methods, however, are still under development, and their effectiveness on physical systems and complex environments is yet to be demonstrated.

A rich body of work has focused on using simplified dynamic models which capture only the main dynamics of the system so as to achieve a balance of computational complexity [8]. Such models are also known as *templates* and are typically lower dimensional than the full (whole-body) model, and also linear with respect to the states of the

This work was supported by the UKRI/EPSRC RAIN Hub [EP/R026084/1] and the EU H2020 Projects MEMMO and THING, the EPSRC grant 'Robust Legged Locomotion' [EP/S002383/1] and a Royal Society University Research Fellowship (Fallon). This work was conducted as part of ANYmal Research, a community to advance legged robotics. The authors are with Dynamic Robots Systems (DRS) group, Oxford Robotics Institute, University of Oxford, UK. Email: {siddhant, mathieu, rorsolino, mfallon, ioannis}@robots.ox.ac.uk



Fig. 2: ANYmal C quadruped used to perform experiments on different terrains outdoors, on grassy and muddy terrain also including slopes, and indoors, similar to the experiments performed using ANYmal B. The policies trained using ANYmal B were transferred to ANYmal C without retraining. The video clips of the demonstrations are provided in Movie S6.

system; the mechanism that relates them back to the whole-body model is instead usually called the *anchor*. Examples of templates include the linear inverted pendulum (LIP) [9] and the spring-loaded inverted pendulum (SLIP) [10], and their derived variants [11].

For the development of robust model-based controllers, the synthesis of stability criteria based on ground reference points (2D points on a flat ground plane) such as the zero moment point (ZMP) [12] and the instantaneous capture point (ICP) [13] is also introduced into the control problem. The templates can be seen as a way to constrain the admissible domain of such ground reference points. In this regard, the balancing problem for legged robots can be translated into an inclusion problem that requires the reference point to belong to its admissible region, i.e. the *support region*, for the robot to be stable. In this way a measure of robustness, the stability *margin*, can be defined as the distance between the reference point and the edges of its admissible region.

Often, simplified models and ground reference points are limited to flat terrain and their descriptive accuracy deteriorates when applied to environments with complex geometry. For such *multi-contact* conditions, the centroidal dynamics model [14] is widely used. It represents the system in a compact manner such that it focuses on the inputs (contact wrenches) and the outputs (center of mass linear and angular accelerations) in the task space of the robot discarding the intermediate joint-space physical quantities. In this case, the reference point for evaluating stability can be described by a 6-dimensional wrench and, therefore, for evaluation of stability, templates that illustrate suitable 6D admissible regions need to be defined [15], [16]. Despite their low dimensionality, such simplified models are still typically too numerically inefficient to be employed in the robot's real-time control loop. An alternative approach is to evaluate the stability of the robot directly using the contact forces [17], [18]. This method, however, necessitates introduction of the contact force variables in the locomotion problem thereby adding to computational delays which result in reduced reactive control.

Recent research on model-predictive control (MPC) based strategies have demonstrated perceptive locomotion over uneven terrain [19], [20]. These controllers, however, often employ system model approximations, due to which, the potential of the physical robot is not fully exploited. Moreover, these methods require exhaustive tuning of planning and control parameters, requiring considerable development time.

Alternatively, an entirely different approach has focused on investigating solutions which can perform a direct mapping between sensory information to desired control signals with minimal computational overhead. Model-free data-driven techniques such as RL have introduced promising substitutes to model-based control methods, and achieved highly dynamic and robust locomotion skills on complex legged robot systems such as ANYmal B [21], [22], [23], Laikago [24] and Cassie [25]. The major challenge of using RL lies in the fact that training these methods is extremely sample inefficient especially for high-dimensional problems such as legged locomotion. When applied to a robot such as ANYmal B, the vast exploration space associated with an RL policy, not only necessitates considerable amount of training samples, but also implies that without precise reward function tuning, the RL agent could converge towards an undesired control behavior. As an alternative, a guided policy exploration approach, such as guided policy search (GPS) [26], which relies on utilizing model-based control solutions in conjunction with data-driven policy optimization can be employed to direct policy convergence and increase sample efficiency.

The possibility of damaging the robotic system itself and the environment of operation during policy exploration necessitates the use of simulation platforms. Ease of access to physics simulators, and the ability to execute simulations much faster than real-time, makes this approach appealing for training RL policies. However, the inaccuracy associated with the complex dynamics that are hard to model in simulation often affect the performance of RL policies when deployed onto real systems. Although there have been works where training was done on the real system itself [27], [28], this has been appropriate for simpler and inherently stable robots which cannot be employed for training complex quadrupedal systems such as the ANYmal robots. To address the reality gap problem, more accurate models of the training environment

can be directly introduced into the simulators. For example, *Hwangbo et al.* introduced an actuator network [21] to model the dynamics of the physical actuators and further performed ablation studies to justify the necessity for accurate models to achieve simulation-to-real transfer. Moreover, for a training environment, it is very likely that not all the dynamics can be precisely modeled. As a solution, domain randomization [29] techniques have been used to train policies which are robust to unaccountable factors [21], [23].

Most of the above mentioned RL research focuses on quadrupedal locomotion on flat ground, overlooking the primary purpose of quadrupedal systems – mobility over rough terrain. *Peng et al.* presented an approach for terrain-adaptive locomotion using RL [30], however, this work was limited to 2D character animation. *Tsounis et al.* presented an approach for terrain-aware gait planning and control using RL for the ANYmal B quadruped [31]. *Tsounis et al.* used a hierarchical approach to plan gaits suitable for navigation to a desired goal position, and then performed whole body control, both using RL. However, different gait planning policies were trained for different terrains, while sim-to-real transfer was not addressed. *Lee et al.* presented an RL based approach for quadrupedal locomotion over challenging terrains [32], however, this work only focused on use of proprioceptive feedback for locomotion without explicitly processing terrain information.

We propose a unified RL and optimal control (OC) based terrain-aware legged locomotion framework as illustrated in Fig. 3. We train a footstep planning RL policy which maps the robot state and terrain information to desired feet positions. In order to track these positions, we extend the *dynamic gaits* motion controller of *Bellicoso et al.* [33] to enable foothold tracking over non-flat terrain. This forms the core of our locomotion framework. In order to account for changes in physical parameters and actuation dynamics, we train a domain adaptive tracking policy which introduces corrective feed-forward joint torques for accurate tracking of the trajectories generated by the motion controller. Furthermore, to enable recovery to a stable state upon perturbations or unexpected events like foot slippage, we train an RL policy to generate low-level joint position commands such that during unstable motions, the RL policy stabilizes the robot and then the control switches back to the footstep planning and tracking modules. A video overview of our proposed framework is presented in Movie 1.

Our main contributions with this work include: (1) A training approach for obtaining a perceptive footstep planning policy which employs a model-based dynamic motion controller. This setup enables us to generate valid dynamic motion plans while ensuring the obtained policy adapts to and exploits the domain largely described by the characteristics of the motion controller. Furthermore, approximating the perceptive footstep planning policy using a neural network implies that we can directly map the exteroceptive and proprioceptive robot state information to desired feet positions getting rid of the need for the computationally exhaustive task of combinatorial search among a large space of feasible footstep plans. Additionally, the model-based control architecture ensures that safety-critical constraints such as joint kinematic limits and

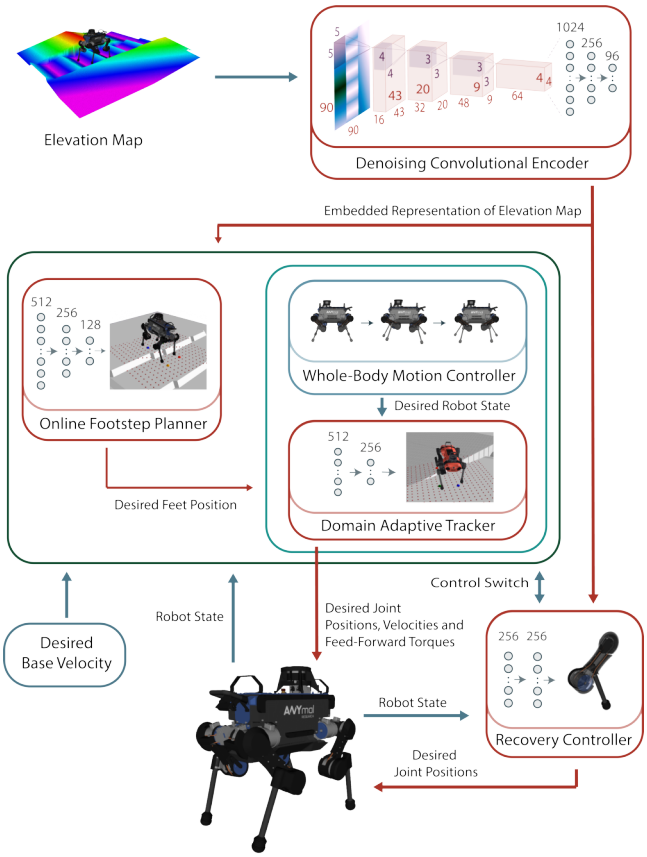



Fig. 3: Overview of the proposed control framework - RLOC. We use an online footstep planning RL policy to generate desired feet positions which are tracked using a model-based whole-body motion controller. In order to perform accurate tracking even in case of changes in robot dynamics, we introduce a domain adaptive tracking policy to generate feed-forward corrective torques. Furthermore, in order to exhibit a quick response to external perturbations, we introduce a recovery control policy which generates low-level joint position commands.

peak joint torques are still enforced by the low-level controller. (2) Obtaining a domain adaptive tracking policy which can be utilized to address system modeling inaccuracies during whole-body motion tracking. This allows us to use a model-based controller on a robot where the modeled characteristics differ from the actual kinematic and dynamic properties of the system. This further enables us to perform domain randomization while training the footstep planning policy. (3) Realizing a modular framework which combines RL and OC based approaches to perform zero-shot generalization to a domain not explicitly modeled during training. This lets us transfer our framework to the physical ANYmal C despite having been trained in a simulator with ANYmal B.

Our framework allows for high-level tracking of reference base velocity commands over complex uneven terrain, enabling the robot to be operated either by a user (through a remote control) or a higher-level goal planner. We evaluate the

performance of RLOC on a wide range of simulated terrains and present the results for the ANYmal B and C quadrupeds. We further transfer RLOC on to the real robots.

## II. Preliminaries

### A. System Model

A general quadrupedal robotic system can be modeled as a floating base $B$ with four actuated limbs attached to it. The robot state can be described w.r.t. a global inertial reference frame $W$ where the base position is expressed as ${}_W r_{WB} \in \mathbb{R}^3$, and the orientation, $\mathrm{q}_{WB} \in SO(3)$, is represented using Hamiltonian unit quaternions with a corresponding rotation matrix $\mathbf{R}_{WB} \in SO(3)$. We assume that the $z$-axis of $W$, $\mathbf{e}_z^W$, aligns with the gravity axis. The angular positions of the rotational joints in each of the limbs are described as a vector $\mathrm{q}_j \in \mathbb{R}^{n_j}$. For the ANYmal robots, $n_j = 12$. The linear and angular velocities of the base w.r.t. the global frame are written as ${}_W \mathrm{v}_{WB} \in \mathbb{R}^3$ and ${}_W \omega_{WB} \in \mathbb{R}^3$ respectively. The generalized coordinates and velocities are thus stacked as vectors q and u where

$$\mathrm{q} = \begin{bmatrix} {}_W r_{WB} \\ \mathrm{q}_{WB} \\ \mathrm{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \quad \mathrm{u} = \begin{bmatrix} {}_W \mathrm{v}_{WB} \\ {}_W \omega_{WB} \\ \dot{\mathrm{q}}_j \end{bmatrix} \in \mathbb{R}^{6+n_j}. \quad (1)$$

We denote positions of the feet in the global reference frame as ${}_W r_{WF} \in \mathbb{R}^{3\times n_f}$, where $n_f = 4$ is the number of feet. We additionally define ${}_W r_{WF_{xy}} \in \mathbb{R}^{2\times n_f}$ as a vector of $x$ and $y$ components of the feet positions.

We introduce a horizontal frame $H$ such that ${}_W r_{WH} = {}_W r_{WB}$ and $\mathbf{R}_{WH} = \mathbf{R}_{WB_z}$, where $\mathbf{R}_{WB_z}$ is the $z$ decomposition of the base rotation matrix which can also be expressed as $\mathbf{R}_{WB} = \mathbf{R}_{WB_z}\mathbf{R}_{WB_y}\mathbf{R}_{WB_x}$. The frame $H$ can thus be interpreted as having the same translation as the robot base frame $B$ w.r.t. the global inertial reference frame, but rotation in the frame $H$ is only permissible along $\mathbf{e}_z^H$, implying that the gravity axis always aligns with the $z$-axis of frame $H$ and the yaw rotation of frame $H$ also always aligns with the yaw rotation of frame $B$. We then represent the local robocentric terrain elevation in the horizontal frame as $\mathbf{M}_H : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^{91\times 91}$, an elevation map corresponding to an area of $1.8 \times 1.8\,\mathrm{m}^2$ and a resolution of $0.02\,\mathrm{m}$.

The quadrupedal system is actuated using the joint control torques $\tau_j \in \mathbb{R}^{n_j}$. These torques are computed at the actuator level using an impedance control model written as

$$\tau_j = K_p(\mathrm{q}_j^* - \mathrm{q}_j) + K_d(\dot{\mathrm{q}}_j^* - \dot{\mathrm{q}}_j) + \tau_{j_{FF}} \quad (2)$$

where $K_p$ and $K_d$ refer to the position and velocity tracking gains respectively, $\mathrm{q}_j^*$ is the vector representing desired joint positions, $\dot{\mathrm{q}}_j^*$, the desired joint velocities, and $\tau_{j_{FF}}$ refers to the feed-forward joint torques.

Note that, for concise presentation, we will assume that all of the translations and rotations are measured and expressed in the global reference frame $W$ unless otherwise stated. In this regard, for the rest of the manuscript, the notation ${}_W r_{WB}$ will be shortened to $r_B$, $\mathbf{R}_{WB}$ to $\mathbf{R}_B$, and similarly for all the other introduced notations.

### B. Motion Parameterization

The motion of the quadrupedal system is expressed in terms of the center of mass (CoM) and feet motions. In consistency with the definitions introduced in [33], we represent the CoM motion plan in the form of a sequence of quintic splines where the $j$-th spline for the $x$ component is represented as

$$\begin{aligned} x(t) &= \alpha_{j5}^x t^5 + \alpha_{j4}^x t^4 + \alpha_{j3}^x t^3 + \alpha_{j2}^x t^2 + \alpha_{j1}^x t + \alpha_{j0}^x \\ &= [t^5 \quad t^4 \quad t^3 \quad t^2 \quad t \quad 1] \\ &\quad \cdot [\alpha_{j5}^x \quad \alpha_{j4}^x \quad \alpha_{j3}^x \quad \alpha_{j2}^x \quad \alpha_{j1}^x \quad \alpha_{j0}^x]^T \\ &= \eta^T(t)\alpha_j^x. \end{aligned} \quad (3)$$

The spline $j$ corresponds to a duration of $\Delta t_j$ such that $t \in [\bar{t}, \bar{t} + \Delta t_j]$. Here $\bar{t}$ refers to the total elapsed time of the motion described by $j-1$ splines. We therefore express the CoM position, $\mathrm{p}_{CoM} \in \mathbb{R}^3$, as $\mathrm{p}_{CoM} = \mathbf{K}(t)\alpha_j$ where

$$\mathbf{K}(t) = \begin{bmatrix} \eta^T(t) & 0 & 0 \\ 0 & \eta^T(t) & 0 \\ 0 & 0 & \eta^T(t) \end{bmatrix}, \quad (4)$$

and $\alpha_j = [\alpha_j^{xT} \quad \alpha_j^{yT} \quad \alpha_j^{zT}]^T$. The CoM velocity is thus represented as $\dot{\mathrm{p}}_{CoM} = \dot{\mathbf{K}}(t)\alpha_j$, and the acceleration as $\ddot{\mathrm{p}}_{CoM} = \ddot{\mathbf{K}}(t)\alpha_j$.

We parameterize the feet motions based on their contact state given by the vector $\mathrm{c}_F \in \{0, 0.5, 1\}^4$ where an open-contact is represented by 0 and close-contact (with zero foot velocity) by 1. Depending on the terrain, there remains a possibility of foot slippage i.e. a close-contact with non-zero foot velocity. This *contact-uncertainty* state is therefore denoted by 0.5. For the purpose of motion planning, we only consider the open (0) and close (1) contact states. The contact-uncertainty state is instead utilized for adapting the joint PD control gains during motion tracking. This is detailed in Section III-B.

We define the swing phase, $\Phi_{o,i}$, for foot $i \in \{0,1,2,3\}$ to correspond to the foot motion where $\mathrm{c}_{F,i} = 0$. Conversely, the stance phase, $\Phi_{c,i}$, corresponds to the motion where $\mathrm{c}_{F,i} = 1$. Both the swing and stance phases can be described by the tuple

$$\Phi = \langle r_F^l, r_F^d, r_F^h, \mathrm{c}_F, t_F^l, t_F^d, v_F^l, v_F^d \rangle \quad (5)$$

where $r_F^l$, $r_F^d$ and $r_F^h$ refer to the vectors describing the feet lift-off, touch-down and clearance positions respectively. The foot clearance position represents the maximum $r_{F_z,i}$ during the given phase. For $\Phi_{o,i}$, $t_{F,i}^l$ is the time elapsed since previous lift-off and $t_{F,i}^d$ is the time until touch-down. For $\Phi_{c,i}$, $t_{F,i}^l$ refers to the time until lift-off and $t_{F,i}^d$ is the time elapsed since previous touch-down. $v_F^l$ and $v_F^d$ refer to the feet lift-off and touch-down velocities respectively. Additionally, for $\Phi_{c,i}$, we consider $r_F^l = r_F^d = r_F^h$.

### C. Reinforcement Learning

The problem of sequential decision making can be formulated in the framework of a discrete time Markov decision process (MDP) [34]. An MDP is defined by the tuple -

$\langle S, A, R, P, \mu \rangle$, where $S$ represents a set of states, $A$ a set of actions, $R : S \times A \times S \to \mathbb{R}$ the reward function, $P : S \times A \times S \to [0, 1]$ the state transition probability, and $\mu$ the initial state distribution. In the context of deep RL, as utilized in our work, an RL agent described in this MDP framework interacts with an environment based on a policy $\pi : S \to \mathcal{P}(A)$ which is modeled as a neural network. This policy can be defined as a function mapping states to probability distributions over actions such that $\pi(a|s)$ denotes the probability of selecting action $a$ in state $s$. The objective of the RL agent is to then obtain a policy $\pi_\theta$ for parameters $\theta$ in order to maximize the expected cumulative discounted return,

$$J(\pi) \doteq \mathop{\mathrm{E}}_{\mathcal{T} \sim \pi_\theta} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right], \tag{6}$$

where $\gamma \in [0, 1)$ is the discount factor and $\mathcal{T}$ denotes a trajectory dependent on $\pi_\theta$.

## III. Methodology

We present a modular terrain-aware quadrupedal planning and control framework for locomotion over uneven terrain. We propose a footstep planner which utilizes the proprioceptive and exteroceptive information to generate the desired feet touch-down positions, $r_F^{d*}$, that are tracked during the immediate swing phases. The footstep planner is sequentially evaluated after each of the previous four swing phases are executed, completing a *gait stride*. The desired footholds are thus updated after each gait stride at the instance when each of the feet is in stance phase, $\mathrm{c}_F = \overrightarrow{1}_F$ where $\overrightarrow{1}_F \in \mathbb{R}^4$ is a vector of ones. The dynamic gaits motion controller, which serves as a motion planner and whole body controller, is then employed to generate and track the desired CoM and feet motion plans. The whole body tracking of motion plans occurs at 400 Hz, while the motion planning optimization is performed in parallel to the control thread and a new optimization problem is run everytime a solution is found.

The domain adaptive tracker, also executed at 400 Hz, introduces additive deviations to the feed-forward joint torques, $\tau_{j_{FF}}$, generated by the motion controller. These additive deviations, $\delta\tau_{j_{FF}}$, are included in the impedance controller shown in eq. 2. Finally, the recovery controller, running at 400 Hz, is activated to perform joint position control when a robot state is encountered such that the motion controller becomes unstable and is unable to execute a state recovery behavior.

We employ this modular control architecture for following reasons:

1) The individual trained modules can be precisely tuned to adapt to different tasks. For example, transferring the framework for use on a quadruped with significantly larger mass would only require adapting either the motion controller or the domain adaptive tracker.
2) In our preliminary experiments, we observed that training a terrain-aware locomotion policy which outputs desired joint position commands required comprehensive tuning of the RL environment parameters, without which, the network often failed to learn, resulting in a behavior where the robot would not respond to velocity commands, instead, would stand in place.
3) Distributing the planning and control tasks between the RL footstep planner and the model-based motion controller enabled us to obtain a footstep planner which was able to generalize to wide range of terrains in less than 32 h of total training. In contrast, the gait planner of DeepGait [31] required 82 h of training time for each of the terrains. Moreover, the gait controller of DeepGait required further 116 h of training time for each of the uneven terrains, and an additional 58 h of training for locomotion on flat-ground, totaling the training time to approximately 454 h for locomotion over three different terrains: Flat-World, Temple-Ascent, and Random-Stairs (see DeepGait [31]).

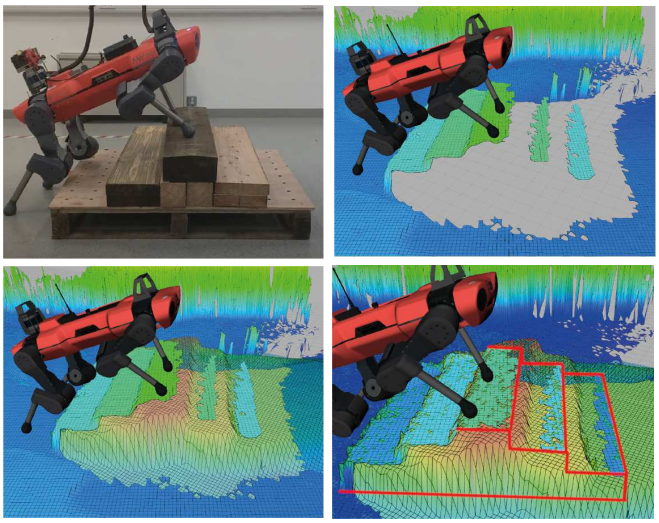


Fig. 4: **top:** Ascending stairs with ANYmal C (left) and the corresponding raw elevation map (right). The unfilled regions in the raw elevation map represent non-observable terrain information which could not be captured by the depth camera on the robot due to occlusions. **bottom:** Filtered elevation map obtained using nearest-neighbors interpolation (left) and the terrain edges (right).

An overview of the control framework is presented in Fig. 3. This section details upon each of the modules. We also describe the MDPs for the footstep planner, domain adaptive tracker and the recovery controller in this section. The detailed RL training setups are presented in Section IV.

### A. Elevation Map Encoding

We extract $\mathbf{M}_H$ using the *elevation mapping* framework presented in [35] and embed it to obtain a latent representation, $\mathrm{Z}_{\mathbf{M}_H} \in \mathbb{R}^{96}$, by employing an encoder network, $E_{\mathbf{M}_H}$, trained using a denoising convolutional auto-encoding strategy. Our motivation for performing this embedding (including the encoding approach) is based on the following reasons:

1) Three of the modules constituting our RLOC framework utilize exteroceptive terrain information. Pre-processing the elevation map to extract terrain features eliminates

repeated processing of the elevation map by each of the modules limiting further computational delays.

2) The denoising property of $E_{\mathbf{M}_H}$ is relevant to perform a sim-to-real transfer. The elevation map extracted on the physical system is often noisy and contains occlusions that cannot be precisely modeled. An example of this is shown in Fig. 4. A nearest-neighbors interpolation technique is employed to obtain a filtered elevation map which often results in distortions (e.g. the edges of the steps are blurred and shifted). The denoising auto-encoding strategy enables extraction of the most relevant terrain features like edges and slopes even with distorted terrain information.
3) A pre-trained terrain feature extractor implies that, during training of RL policies, the RL agent does not need to learn the parameters for the terrain feature extraction block thereby making the training faster while also adapting to a pre-trained encoder which can be shared with multiple modules.
4) Even though different strategies such as fast Fourier transform (FFT) can be employed to compress the elevation map, our work focuses on utilizing deep learning strategies for *feature extraction*. This enables us to adapt the previously trained network to a diverse class of terrains by retraining the encoder. Additionally, strategies such as transfer learning can be utilized to reduce training complexity. The feature extractor can also be retrained along with an RL policy to adapt to different tasks not introduced during prior training.

The architecture of the denoising convolutional auto-encoder is presented in Fig. 5. Note that, while extracting the elevation map, we clip $\mathbf{M}_H$ in the range $[-2,2]$ m. For network training and evaluation stability, we further normalize $\mathbf{M}_H$ by performing a scalar division by 2 to obtain $\mathbf{M}_H^n$ with elements in the range $[-1,1]$. As part of training, our objective is then to learn the weights of the convolutional encoder $E_{\mathbf{M}_H}$, such that

$$\mathrm{Z}_{\mathbf{M}_H} = E_{\mathbf{M}_H}(\mathbf{M}_H^n + \mathbf{N}_{\mathbf{M}}(\mathbf{M}_H^n)) \tag{7}$$

where $\mathbf{N}_{\mathbf{M}}$ represents the artificial noise introduced in $\mathbf{M}_H^n$, and

$$\hat{\mathbf{M}}_H^n = D_{\mathbf{M}_H}(\mathrm{Z}_{\mathbf{M}_H}) \tag{8}$$

where $D_{\mathbf{M}_H}$ is the decoder network which maps the latent embedding of $\mathbf{M}_H^n$ to its approximate reconstruction $\hat{\mathbf{M}}_H^n$. Both $E_{\mathbf{M}_H}$ and $D_{\mathbf{M}_H}$ are trained together using supervised learning to minimize the reconstruction loss

$$\mathcal{L}_{\mathbf{M}_H} = (\mathbf{M}_H^n - \hat{\mathbf{M}}_H^n)^2 . \tag{9}$$

The artificial noise, $\mathbf{N}_{\mathbf{M}} : \mathbb{R}^{91\times 91} \to \mathbb{R}^{91\times 91}$, is expressed as

$$\begin{aligned}\mathbf{N}_{\mathbf{M}}(\mathbf{M}) = n_{\mathcal{N}}\mathcal{N}_{91\times 91}(0,1) \\ + (\mathcal{I}(\mathbf{M}, p_{\mathcal{I}}) - \mathbf{M}) \\ + n_{\mathcal{C}}(\mathcal{C}(\mathbf{M}, k_b) - \mathbf{M})\end{aligned} \tag{10}$$

where $\mathbf{M} \in \mathbb{R}^{91\times 91}$ is a matrix similar to $\mathbf{M}_H^n$, $\mathcal{N}$ is the additive Gaussian noise, $\mathcal{I}$ is the impulse noise also know as salt-and-pepper noise, $p_{\mathcal{I}} \sim \mathcal{U}(0, 0.05)$ is the probability of introduced impulses sampled uniformly in the range $[0, 0.05]$, $\mathcal{C}$ is the 2D convolution function, and $k_b \in \mathbb{R}^{3\times 3}$ is the convolutional blurring kernel written as

$$k_b = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} . \tag{11}$$

$n_{\mathcal{N}} \sim \mathcal{U}(0, 0.05)$ and $n_{\mathcal{C}} \sim \mathcal{U}(0, 1)$ are the coefficients for weighting the noise functions.

To train the denoising convolutional autoencoder, we use our terrain generator tool, described in Section IV-A1, to generate $20 \times 20\,\mathrm{m}^2$ terrains, where each terrain is represented as $\mathbf{T} \in \mathbb{R}^{1001\times 1001}$. These terrains contain objects such as stairs, wave, bricks, planks and unstructured ground. We slice a $91 \times 91$ matrix from $\mathbf{T}$ at randomly selected positions and further use data-augmentation techniques such as rotation, mirroring and modifying contrast during training of the autoencoder online. This augmentation is done in parallel to the main training session in a separate thread. Figure 6 shows examples of regeneration of noisy terrain inputs using the trained autoencoder.

### B. Motion Controller

The *dynamic gaits* motion controller of *Bellicoso et al.* [33] generates CoM motion plans by solving a nonlinear optimization problem using the sequential quadratic programming (SQP) approach [36] for a time horizon $t_H^c$ which corresponds to the periodicity of the active locomotion gait. The SQP optimization parameters correspond to the spline coefficients $\alpha = [{\alpha_0}^T \cdots {\alpha_i}^T \cdots {\alpha_{n_s}}^T]$ where $n_s$ is the number of different support polygons on the optimization horizon. This formulation, where the SQP also optimizes for $\alpha_i^z$ as opposed to assuming a fixed base height to ease on the computational complexity, enables us to adapt the dynamic gaits motion controller for locomotion over uneven terrain. For stability, dynamic gaits uses constraints on the ZMP. It also penalizes deviations from a *path regularizer*, which represents an approximated CoM motion plan for a duration of $t_H^c$, to limit the drift of the robot base w.r.t. the reference footholds which may occur due to external perturbations and continuous updates of the motion plan. The path regularizer, $\mathcal{T}_R^c$, is computed based on the estimated footprint center and the reference base velocity command, $\mathbf{c}^*$. We introduce a control frame $C$ such that $r_C = r_B$, $\mathbf{R}_{C_z} = \mathbf{R}_{B_z}$, and $\mathbf{e}_z^C$ aligns with the normal of the estimated terrain plane $\mathbf{e}_z^T$. The velocity command $\mathbf{c}^*$ is therefore expressed in the control frame as

$$\mathbf{c}^* = [\mathrm{v}_x^* \mathbf{e}_x^C \quad \mathrm{v}_y^* \mathbf{e}_y^C \quad \omega_z^* \mathbf{e}_z^C]^T \tag{12}$$

where $\mathrm{v}_x^*$ and $\mathrm{v}_y^*$ are the desired heading and lateral velocities expressed along $\mathbf{e}_x^C$ and $\mathbf{e}_y^C$ respectively, and $\omega_z^*$ is the desired yaw rate around the vertical axis $\mathbf{e}_z^C$. We use a similar approach as presented in [33] to compute $\mathcal{T}_R^c$ and introduce the following adaptations in the CoM motion plan optimization problem: (1) the terrain plane is estimated by fitting a planar surface through the current and planned footholds as opposed to current and previous footholds (2) unlike dynamic gaits which sets the height of the path regularizer $\mathcal{T}_{R_z}^c$ at a fixed reference value

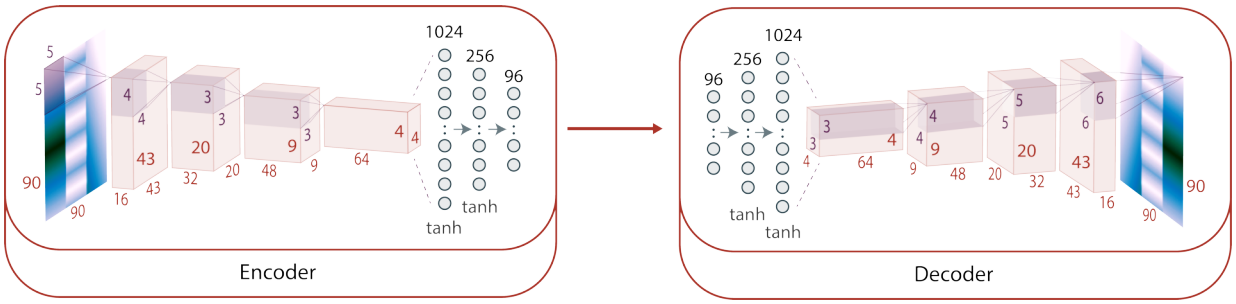



Fig. 5: Network architecture of the denoising convolutional autoencoder used to generate an embedded representation of the local elevation map. We use the *leaky ReLU* activation function for the convolutional layers and the *tan hyperbolic* activation function for the dense layers.

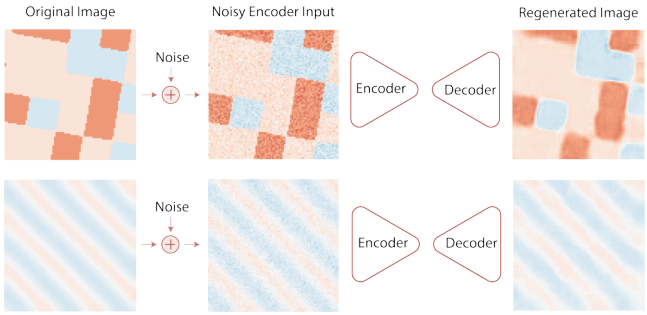



Fig. 6: Examples of regeneration of noisy images using our trained autoencoder - **top:** a patch from the bricks terrain and **bottom:** a patch from the sine wave terrain.

$h_{ref}$ with respect to the control frame, we adapt this value such that:

$$\mathcal{T}^c_{R_z} = \mathbf{M}_H(\mathcal{T}^c_{R_x}, \mathcal{T}^c_{R_y}) + h_{ref} - 0.125|\mathbf{R}_{C_y}|$$

where $\mathbf{R}_{C_y}$ is the angle of the terrain plane on the pitch axis of the robot. This modification allows us to increase the reachability of the feet on stairs and complex terrains. The support polygons (used in the dynamic constraints of the optimization) are defined by separate modules, the foothold planner and contact scheduler.

The foothold planning module in dynamic gaits is set up as a quadratic programming (QP) problem to obtain the $x$ and $y$ components of the desired footholds $r^*_{F_{xy}}$. The nominal footholds expressed in the horizontal reference frame ${}_H r^n_{HF_{xy}}$ corresponds to the projections of the robot hip locations measured and expressed in the horizontal frame along the vertical axis $\mathbf{e}^H_z$. The cost function of the QP problem then includes, among others, a penalty term for deviations from the nominal foothold locations. The foothold optimizer gives us the desired feet touch down positions, $r^{d*}_F$, where $r^{d*}_{F_z}$ is extracted from the terrain plane estimate, for each swing phase.

The contact scheduler module, manages the feet contact-events for a full stride of an active gait while also supporting smooth gait transitions giving us the lift-off, $t^l_F$, and touch-down, $t^d_F$, timings for the feet swing and stance phases based on the assumption that $v^l_F$ and $v^d_F$ are zeros. This module also allows to slow down the execution of the gait in case a touch-down event is not detected at the expected time. The $\Phi_o$ and $\Phi_c$ thus obtained can be used for generating motion plans for $r^{d*}_F$. Moreover, dynamic gaits considers a fixed foot clearance position $r^h_F$ for $\Phi_o$.

In our RLOC framework, the foothold optimizer is replaced by the RL footstep planner (described below) which generates the desired feet touch-down positions, $r^{FP}_F$, for locomotion over uneven terrain. Additionally, $r^h_{F,i}$ for swing phase $\Phi_{o,i}$ of foot $i$ is now computed based on the maximum terrain elevation observed between the line, $l^\Phi_{ld}$, connecting $r^l_{F_{xy}}$ and $r^{FP}_{F_{xy}}$, given by

$$r^h_{F_{z,i}} = \max(\mathbf{M}_H(X,Y)) + 0.05\,\text{m} \tag{13}$$

where $\mathbf{M}_H$ is the elevation map expressed in the global reference frame, $X$ and $Y$ are the vectors containing the $x$ and $y$ coordinates of $l^\Phi_{ld}$ sampled at a resolution of 0.02 m. The additional offset of 0.05 m to the foot clearance position is introduced to avoid collisions during inaccurate motion tracking. To limit uncertain behavior that may occur due to imprecise elevation map generation as a result of sensor noise and occlusions, $r^h_F$ is clipped to be less than $r^l_F + 0.25\,\text{m}$.

In contrast to dynamic gaits, which uses the estimate of the terrain plane to determine the orientation of the friction cone, we compute the terrain normal at each foot using the information from the elevation map. The normals are estimated using central finite difference at the feet positions. Since the information from the elevation map is not always reliable, we also add the following checks:

1) When $c_{F,i} = 1$, the foot height is close to the terrain elevation such that $|r^d_{F_z,i} - \mathbf{M}_H(r^d_{F_{xy},i})| < \varepsilon_d$ where $\varepsilon_d = 0.04\,\text{m}$.
2) The terrain elevation within a deviation of $\pm 4\,\text{cm}$ centered at $r^d_{F_{xy},i}$ is planar.
3) The estimated terrain normal at foot $i$, $\mathbf{e}^{T_i}_z$, stays in realistic bounds, i.e. $|cos^{-1}(\mathbf{e}^{T_i}_z \cdot \mathbf{e}^W_z)| < \pi/3\,\text{rad}$.

If any of these checks fail for a foot $i$, we fall back to a vertical estimate for the corresponding friction cone.

In the original work, the CoM and foothold motion optimizers are run every time a new solution is obtained. In our training setup, however, we use a fixed optimization frequency of 50 Hz which corresponds to a conservative lower bound of the planning frequency on the hardware. This allows us to compute plans at a frequency independent of the computation

time enabling execution of parallel simulations during training of our RL policies. For deployment, we switch back to the original variable optimization frequency approach. Moreover, during evaluation, we remove the foothold optimization module from the motion controller and only utilize the RL footstep planner executed after every gait stride.

The reference motion plans are tracked using a whole-body controller [37] based on a hierarchical QP optimization framework to optimize for the generalized accelerations $\dot{\mathrm{u}}$ and feet contact forces to generate the feed-forward torques $\tau_{j_{FF}}$. The impedance control torques are then computed by extracting the desired joint positions, $\mathrm{q}_j^*$, and joint velocities, $\dot{\mathrm{q}}_j^*$, from the feet motion plans using inverse kinematics.

The work of *Jenelten et al.* on locomotion over slippery surfaces [38] proposes the use of impedance gains depending on the feet contact-states. For a foot $i$, the gains are selected according to the rule

$$K_{p,d}^{\mathrm{c}_{F,i}=0.5} >> K_{p,d}^{\mathrm{c}_{F,i}=1} > K_{p,d}^{\mathrm{c}_{F,i}=0} \tag{14}$$

such that, in the contact-uncertainty state $\mathrm{c}_{F,i} = 0.5$, the high gains used for joints in the leg containing the foot $i$ prevent further leg slippage due to increased stiffness. For detailed description, we defer the reader to the original work. We use this approach of adaptive impedance for whole-body tracking of the reference motion plans in RLOC.

### C. Footstep Planner

Computing new set of desired feet positions after each gait stride, specifically after each of the previous four feet swing phases have been executed, allows us to formulate the problem of footstep planning as a sequential MDP. In particular, we represent the problem as an infinite-state space and infinite-action space MDP.

*1) State and Action:* We define the MDP state as a tuple $\mathbf{s}_f := \langle s_R, s_v, s_j, s_c, s_M \rangle$, and the action $\mathbf{a}_f := \langle {}_H r^*_{HF_{xy}} \rangle$. The state $\mathbf{s}_f \in \mathbb{R}^{204}$ comprises terms representing robot base orientation $s_R \in \mathbb{R}^3$, base linear and angular velocities $s_v \in \mathbb{R}^6$, sparse representation of history of joint positions and velocities $s_j \in \mathbb{R}^{96}$, desired base velocity $s_c \in \mathbb{R}^3$, and an embedded representation of the robocentric elevation map $s_M \in \mathbb{R}^{96}$. The exact definitions of the footstep planner (FP) state terms are provided in Table I. The action $\mathbf{a}_f \in \mathbb{R}^8$ comprises the desired foothold locations for each of the four feet represented in the horizontal frame. The $z$ components of the desired feet positions are directly extracted from $\mathbf{M}_H$. Therefore, the desired feet positions measured and expressed in horizontal frame can be transformed into the world frame representation using the transformation

$$r_{F,i}^{FP} = (\mathbf{R}_H)\,({}_H r^*_{HF,i}) \;+\; r_B \tag{15}$$

for foot $i$. The feet positions generated by the footstep planner, $r_F^{FP}$, are then forwarded to the dynamic gaits motion controller.

Note that, we introduce the history of joint states in the state space to infer the active locomotion gait. The different gaits result in different *stability margin* [39], [40], a metric introduced in the reward function to assess the dynamic stability of the robot, which implies that the desired feet positions generated by the footstep planner depend on the gait in use thereby necessitating the state space to contain information relating to the robot locomotion gait.

TABLE I: State term definitions for the footstep planner MDP. Here, $t$ refers to the current measurement and the duration between $t-1$ and $t$ corresponds to 2.5 ms.

| FP State |
|---|
| $s_R = \mathbf{e}_z^B$ |
| $s_v = [(\mathbf{R}_B^T \mathrm{v}_B)^T \quad (\mathbf{R}_B^T \omega_B)^T]^T$ |
| $s_j = [\mathrm{q}_{j_t}^T \quad \mathrm{q}_{j_{t-5}}^T \quad \mathrm{q}_{j_{t-10}}^T \quad \mathrm{q}_{j_{t-50}}^T \quad \dot{\mathrm{q}}_{j_t}^T \quad \dot{\mathrm{q}}_{j_{t-5}}^T \quad \dot{\mathrm{q}}_{j_{t-10}}^T \quad \dot{\mathrm{q}}_{j_{t-50}}^T]^T$ |
| $s_c = \mathbf{c}^*$ |
| $s_M = \mathrm{Z}_{\mathbf{M}_H}$ |

*2) Reward Function:* The reward function in the MDP formulation encourages stable dynamic locomotion over uneven terrain in order to track a desired base velocity command while minimizing energy consumption. During execution of the sequential footstep planning MDP in an episodic framework, the planning state transition, $\mathbf{P}_{f_n} := \langle \mathbf{s}_{f_n}, \mathbf{a}_{f_n}, \mathbf{s}_{f_{n+1}} \rangle$, is only observed after the motion plans for each of the feet are executed. The tracking of these motion plans is instead performed at each control step executed at 400 Hz. Therefore, each $\mathbf{P}_f$ in the footstep planning MDP comprises several control steps. The control behavior observed within these state transitions is still relevant to assess the performance of the footstep planner especially pertaining to velocity tracking and foot slippage. In this regard, our reward function includes a *running* reward term, $\mathbf{r}_F^{\mathrm{r}} \in \mathbb{R}$, accumulated over the control horizon for each $\mathbf{P}_f$, and a *final* reward term $\mathbf{r}_F^{\mathrm{f}} \in \mathbb{R}$, computed after a state transition is observed. The reward function can then be expressed as

$$\mathbf{r}_F = \mathbf{r}_F^{\mathrm{f}} + \frac{1}{(n+1)_t - (n_t+1)} \left( \sum_{t=n_t+1}^{(n+1)_t} \mathbf{r}_{Ft}^{\mathrm{r}} \right) \tag{16}$$

where $n_t$ corresponds to the control step at which state $\mathbf{s}_{f_n}$ is observed. The duration between $t = n_t + 1$ and $t = (n+1)_t$ is not fixed and depends upon the touch-down and lift-off timings. We express the running reward at control step $t$ as

$$\mathbf{r}_{Ft}^{\mathrm{r}} = 8\mathbf{r}_{vt} - 0.016 d_f \mathbf{r}_{\tau t} - 2.5 d_f \mathbf{r}_{\mu_t} + 4\mathbf{r}_{mt} \tag{17}$$

comprising the velocity tracking reward $\mathbf{r}_v$, control joint torque penalty $\mathbf{r}_\tau$, feet slippage penalty $\mathbf{r}_\mu$, and the stability margin reward $\mathbf{r}_m$. The weighting factor $d_f \in [0,1]$ corresponds to the training curriculum and is increased as training progresses to fine tune the behavior of the RL agent. We update the curriculum factor after each policy iteration by the rule $d_{f_{k+1}} \leftarrow d_{f_k}^{0.9986}$ where $k$ refers to the policy iterations and $d_{f_0} = 0.02$. The final reward is represented as

$$\mathbf{r}_F^{\mathrm{f}} = -2 d_f \mathbf{r}_n - 1 d_f \mathbf{r}_e - 2.5 d_f \mathbf{r}_\mu + 4\mathbf{r}_m + 2\mathbf{r}_h \tag{18}$$

comprising the penalty for deviation from nominal footholds $\mathbf{r}_n$, reward for distance from terrain edges $\mathbf{r}_e$, feet slippage penalty $\mathbf{r}_\mu$, the robot stability margin reward $\mathbf{r}_m$ based on [40], and the feet height reward relating to the closeness to the nominal feet height $\mathbf{r}_h$. The definitions for the reward terms are presented in Table II.

TABLE II: Reward term definitions for the footstep planner MDP. Here $K$ refers to the logistic kernel defined as $K(x) := (e^x + 2 + e^{-x})^{-1}$. $\mathcal{S}_m$ is the function mapping robot state information to the robot stability margin, $N_F$ represents the feet contact normals, $\overrightarrow{F}_B$ represents the external force on base, $\overrightarrow{\tau}_B$ is the external torque applied on base, $\mathbf{T}_\mu$ is the terrain friction coefficient, and $\mathbf{C_M}$ maps the foot position to a cost describing the distance of the foot from the terrain edges. The functions $\mathcal{S}_m$ and $\mathbf{C_M}$ are detailed in the Appendix. ${}_H r^n_{HF_z}$ is the $z$ component of the nominal feet positions.

| FP Rewards |
|---|
| $\mathbf{r}_v = K\left(5\|\mathbf{c}^* - \mathbf{c}\|^2\right)$ |
| $\mathbf{r}_\tau = \|\tau_j\|^2$ |
| $\mathbf{r}_\mu = \sum \|\dot{r}_{F,i}\|, \quad \forall i, \mathrm{c}_{F,i} = 1, i \in \{0,1,2,3\}$ |
| $\mathbf{r}_m = \mathcal{S}_m(\mathrm{q}, \mathrm{u}, \dot{\mathrm{u}}, \mathrm{r}_F, \mathrm{c}_F, N_F, \overrightarrow{F}_B, \overrightarrow{\tau}_B, \mathbf{T}_\mu)$ |
| $\mathbf{r}_n = \|{}_H r_{HF_{xy}} - {}_H r^n_{HF_{xy}}\|^2$ |
| $\mathbf{r}_e = \sum \mathbf{C_M}(r_{F_{xy},i}), \quad \forall i$ |
| $\mathbf{r}_h = K(8\|{}_H r_{HF_z} - {}_H r^n_{HF_z}\|^2)$ |

*3) Episode Termination Criteria:* We execute the footstep planning MDP in an episodic framework employing the following termination criteria while training the footstep planner:

- Angle between the terrain normal and base vertical axis exceeds a threshold, $|\cos^{-1}(\mathbf{e}^B_z \cdot \mathbf{e}^C_z)| > \pi/4$
- Any of the robot links except for the feet collide with the terrain.
- Robot self-collision

Upon termination, the robot state is reset and a new episode is executed.

*4) Policy Parameterization:* We parameterize the footstep planning policy as a bounded Gaussian distribution $\pi_{\theta_f}(\mathbf{a}|\mathbf{s}_f) := \tanh(\mathcal{N}(\mathbf{a}|\mu_{\theta_f}(\mathbf{s}_f), \sigma_{\theta_f}(\mathbf{s}_f)))$ where both the mean $\mu_{\theta_f}(\mathbf{s}_f)$ and the log standard deviations $\log(\sigma_{\theta_f}(\mathbf{s}_f))$ are state-dependent outputs of a neural network. The footstep planning network architecture representing the function mapping state to the action mean is shown in Fig. 7. The log standard deviation is extracted using a linear mapping of the latent vector obtained as an output of the last hidden layer. This state-dependent stochasticity is used to encourage exploration during training. We utilize a custom implementation of soft actor critic (SAC) [41], heavily based on *RLlib* [42], to train the footstep planning policy $\pi_{\theta_f}$ by performing entropy-regularized reinforcement learning. For a slow environment setup of footstep planning, which includes prediction of desired footholds and further includes tracking of these feet positions using the motion controller, SAC offers better sample efficiency in comparison to the widely used on-policy alternatives, trust region policy optimization (TRPO) [43] and proximal policy optimization (PPO) [44], and has further been demonstrated to perform efficiently on quadrupedal locomotion tasks [45]. The training is detailed in Section IV-A.

### D. Domain Adaptive Tracker

The model-based characteristics of the dynamic gaits motion controller implies that, in order to exhibit robust motion tracking, significant tuning of the controller parameters are required. This is especially relevant for computing $\tau_{j_{FF}}$ (see eq. 2) which is written as

$$\tau_{j_{FF}} = \mathrm{M}_j \dot{\mathrm{u}}^* + \mathrm{h}_j - \mathrm{J}^T_{sj} \lambda^* \tag{19}$$

where $\mathrm{M}_j$ represents the mass matrix relative to the joints, $\mathrm{h}_j$ comprises Coriolis, centrifugal and gravity terms, $\mathrm{J}_{sj}$ corresponds to the support Jacobian, $\dot{\mathrm{u}}^*$ is the desired generalized accelerations, and $\lambda^*$ represent the desired contact forces. This is detailed in [37]. Executing such a controller on a system where the robot dynamics differ from the modeled dynamics will result in failure to perform motion plan tracking. In this regard we introduce a domain adaptive tracker to generate corrective joint torques, $\delta\tau_j$, for impedance control which can then be rewritten as

$$\tau^c_j = K_p(\mathrm{q}^*_j - \mathrm{q}_j) + K_d(\dot{\mathrm{q}}^*_j - \dot{\mathrm{q}}_j) + \tau_{j_{FF}} + \delta\tau_j \ . \tag{20}$$

This reformulation offers us the ability to deploy the motion controller on systems with different dynamics without the need for retuning the control parameters. Our motivation for introducing the domain adaptive tracker is therefore two-fold:

1) When deployed on a physical system, RLOC can be made robust to system disturbances, encouraging accurate motion plan tracking even with change in dynamic properties, such as added mass, without the need for explicit modeling of system inaccuracies.
2) Introducing the domain adaptive tracker in an RL environment setup employing a model-based motion controller allows us to perform dynamics randomization without retuning the controller. This enables us to utilize the domain adaptive tracker for training the footstep planning policy with wide domain of kinematic and dynamics properties eventually letting us perform a zero-shot transfer of the footstep planner trained with ANYmal B to ANYmal C.

*1) State and Action:* We formulate the problem of domain adaptive tracking in the framework of MDP, such that, given the robot state and the history of tracking errors, the domain adaptive tracker (DAT) implicitly models the deviation in system dynamics from the modeled dynamics to generate corrective joint torques at each control step. We define the DAT state as

$$\mathbf{s}_d := \langle s_r, s_{\delta r}, s_{\mathbf{R}}, s_{\mathbf{R}^*}, s_{\mathbf{v}}, s_{\delta\mathbf{v}}, s_{\mathbf{j}}, s_F, s_\tau, s_M, s_b, s_{\mathbf{b}} \rangle \tag{21}$$

where $\mathbf{s}_d \in \mathbb{R}^{330}$ comprises vector representing the history of robot base position $s_r \in \mathbb{R}^6$, history of base position tracking error $s_{\delta r} \in \mathbb{R}^9$, history of robot orientation $s_{\mathbf{R}} \in \mathbb{R}^9$, history of desired orientation $s_{\mathbf{R}^*} \in \mathbb{R}^9$, history of robot velocities $s_{\mathbf{v}} \in \mathbb{R}^{18}$, history of velocity tracking errors $s_{\delta\mathbf{v}} \in \mathbb{R}^{18}$, history of joint state $s_{\mathbf{j}} \in \mathbb{R}^{72}$, history of desired feet positions and velocities $s_F \in \mathbb{R}^{72}$, previous corrective torques $s_\tau \in \mathbb{R}^{12}$, embedded elevation map $s_M \in \mathbb{R}^{96}$, robot base position relative to the elevation map update frame $s_b \in \mathbb{R}^3$, robot orientation in the elevation map update frame $s_{\mathbf{b}} \in \mathbb{R}^6$. It is important to note that, unlike in the case of footstep planning where each call to the policy occurs after each gait stride, the domain adaptive tracking policy is executed at 400 Hz. Since the perception

sensors on the robot are significantly slower, the elevation map on the physical robot cannot be updated at 400 Hz. In order to account for the asynchronous updates of the exteroceptive and proprioceptive feedback, we store the reference frame of the robot's base during each update of the elevation map and include the current state of the robot in this frame in the state tuple. The *elevation map update frame* is represented as $B_{\mathbf{M}_H}$. The definitions of the state terms are provided in Table III.

TABLE III: State term definitions for the domain adaptive tracking MDP. Here, subscript notation $t$ on a frame indicates the state of the frame at control step $t$, for example, $B_{t-4}$ represents the state of frame $B$ at control step $t-4$. If no subscript exists, the frame is considered to be in the current state $t$. ${}_B r^*_{BB_t}$ is the desired base position at control step $t$ expressed in the current measured base frame, $\hat{\mathbf{e}}^B_z$ the desired orientation, $r^*_F$ the desired feet positions, $\dot{r}^*_F$ the desired feet velocity, $\mathrm{v}^*_B$ desired linear velocity and $\omega^*_B$ is the desired angular velocity. The desired robot state at control step $t$ is extracted from the motion plans generated by dynamic gaits.

| DAT State |
|---|
| $s_r = [{}_B r^T_{BB_{t-4}} \quad {}_B r^T_{BB_{t-8}}]^T$ |
| $s_{\delta r} = [{}_B r^{*T}_{BB_t} \quad ({}_B r^*_{BB_{t-4}} - {}_B r_{BB_{t-4}})^T \quad ({}_B r^*_{BB_{t-8}} - {}_B r_{BB_{t-8}})^T]$ |
| $s_{\mathbf{R}} = [(\mathbf{e}^B_z)^T \quad (\mathbf{e}^{B_{t-4}}_z)^T \quad (\mathbf{e}^{B_{t-8}}_z)^T]^T$ |
| $s_{\mathbf{R}^*} = [(\hat{\mathbf{e}}^B_z)^T \quad (\hat{\mathbf{e}}^{B_{t-4}}_z)^T \quad (\hat{\mathbf{e}}^{B_{t-8}}_z)^T]^T$ |
| $s_{\mathbf{v}} = [(\mathbf{R}^T_B \mathrm{v}_B)^T \quad (\mathbf{R}^T_{B_{t-4}} \mathrm{v}_{B_{t-4}})^T \quad (\mathbf{R}^T_{B_{t-8}} \mathrm{v}_{B_{t-8}})^T$ $(\mathbf{R}^T_B \omega_B)^T \quad (\mathbf{R}^T_{B_{t-4}} \omega_{B_{t-4}})^T \quad (\mathbf{R}^T_{B_{t-8}} \omega_{B_{t-8}})^T]^T$ |
| $s_{\delta \mathbf{v}} = [(\mathbf{R}^T_B(\mathrm{v}^*_B - \mathrm{v}_B))^T \quad (\mathbf{R}^T_{B_{t-4}}(\mathrm{v}^*_{B_{t-4}} - \mathrm{v}_{B_{t-4}}))^T \quad (\mathbf{R}^T_{B_{t-8}}(\mathrm{v}^*_{B_{t-8}} - \mathrm{v}_{B_{t-8}}))^T$ $(\mathbf{R}^T_B(\omega^*_B - \omega_B))^T \quad (\mathbf{R}^T_{B_{t-4}}(\omega^*_{B_{t-4}} - \omega_{B_{t-4}}))^T \quad (\mathbf{R}^T_{B_{t-8}}(\omega^*_{B_{t-8}} - \omega_{B_{t-8}}))^T]^T$ |
| $s_{\mathbf{j}} = [\mathrm{q}^T_{j_t} \quad \mathrm{q}^T_{j_{t-4}} \quad \mathrm{q}^T_{j_{t-8}} \quad \dot{\mathrm{q}}^T_{j_t} \quad \dot{\mathrm{q}}^T_{j_{t-4}} \quad \dot{\mathrm{q}}^T_{j_{t-8}}]^T$ |
| $s_F = [({}_B r^*_{BF})^T \quad ({}_B r^*_{BF_{t-4}})^T \quad ({}_B r^*_{BF_{t-8}})^T$ $(\mathbf{R}^T_{B_t} \dot{r}^*_{F_t})^T \quad (\mathbf{R}^T_{B_{t-4}} \dot{r}^*_{F_{t-4}})^T \quad (\mathbf{R}^T_{B_{t-8}} \dot{r}^*_{F_{t-8}})^T]^T$ |
| $s_\tau = \delta \tau_{t-1}$ |
| $s_M = \mathrm{Z}_{\mathbf{M}_H}$ |
| $s_b = ({}_{B_{\mathbf{M}_H}} r_{(B_{\mathbf{M}_H} B)})$ |
| $s_{\mathbf{b}} = [(\mathbf{e}^{B_{\mathbf{M}_H}}_x)^T \quad (\mathbf{e}^{B_{\mathbf{M}_H}}_y)^T]^T$ |

The DAT action can be defined as $\mathbf{a}_d := \langle \delta \tau_j \rangle$, where $\delta \tau_j \in \mathbb{R}^{12}$ is the corrective joint torque introduced in the impedance controller as shown in eq. 20.

*2) Reward Function:* Domain adaptive tracking can be expressed as an imitation learning problem where the DAT policy tracks the reference motion plan by generating corrective feedforward joint torques. The reward function is then expressed in terms of tracking errors as

$$\mathbf{r}_D = \mathbf{r}_b + 0.125\mathbf{r}_\mathbf{v} + 1.5\mathbf{r}_j + 0.4\mathbf{r}_{\dot{j}} \tag{22}$$

where $\mathbf{r}_b$ promotes minimizing base position tracking errors, $\mathbf{r}_\mathbf{v}$ encourages base velocity tracking, $\mathbf{r}_j$ relates to joint position tracking, and $\mathbf{r}_{\dot{j}}$ is the joint velocity tracking reward. The reward definitions are given in Table IV.

TABLE IV: Reward term definitions for the domain adaptive tracking MDP.

| DAT Rewards |
|---|
| $\mathbf{r}_b = K(10 \Vert {}_B r^*_{BB_t} \Vert^2)$ |
| $\mathbf{r}_\mathbf{v} = K(6 \Vert \mathbf{R}^T_B(\mathrm{v}^*_B - \mathrm{v}_B) \Vert^2) + K(4 \Vert \mathbf{R}^T_B(\omega^*_B - \omega_B) \Vert^2)$ |
| $\mathbf{r}_j = K(20 \Vert \mathrm{q}^*_j - \mathrm{q}_j \Vert^2)$ |
| $\mathbf{r}_{\dot{j}} = K(8 \Vert \dot{\mathrm{q}}^*_j - \dot{\mathrm{q}}_j \Vert^2)$ |

*3) Episode Termination Criteria:* The termination criteria include:

- Any of the robot links except for the feet collide with the terrain.
- Robot self-collision.

*4) Policy Parameterization:* Unlike in the case of footstep planning which utilizes a stochastic policy, we parameterize the DAT policy in the form of a discrete probability distribution, wherein, for a particular action, $\mathbf{a}_{s_d} \in A$, the distribution $\pi_{\theta_d}(\mathbf{a}|\mathbf{s}_d) := \delta^{s_d}_n \ \forall \mathbf{a}_n \in A$ where $\delta$ refers to Kronecker delta. Since the probability is non-zero for only a particular action in a given state, we can represent the policy as a *deterministic* mapping between a state to action. We therefore rewrite $\pi_{\theta_d}(s) : S \rightarrow A$. We model the policy using a neural network. The architecture is presented in Fig. 7. Additionally, we clip the action in the range $[-40, 40]$ N m. We use the successor to the continuous action-space deep deterministic policy gradient (DDPG) [46] algorithm - twin delayed deep deterministic policy gradient (TD3) [47] to train our domain adaptive tracking policy using Gaussian noise regularization as opposed to time-correlated exploration through Ornstein-Uhlenbeck process. For this, we used the RLlib TD3 implementation. Note that, for the footstep planner, the stochasticity of the footstep planning policy was leveraged to perform exploration during training. In the case of the deterministic DAT policy, however, we perform exploration by introducing noise during execution of an action. We detail upon the training in Section IV-B.

### *E. Recovery Controller*

To address system disturbances and external perturbations, dynamic gaits additionally employs an inverted pendulum model based stabilizing cost term in its foothold optimizer. The cost, $w_s(\mathbf{c}^* - \mathbf{c}_{hip_i})\sqrt{h/g}$, for hip $i$ with height $h$ and acceleration due to gravity $g$, encourages the foothold optimizer to minimize the error between the desired velocity $\mathbf{c}^*$ and the hip velocity $\mathbf{c}_{hip_i}$. This results in generation of a desired foothold in the direction of the perturbation. Depending on the weighting $w_s$, the foothold optimizer either prioritizes reference velocity tracking or stability. This, however, relates to the motion controller *adapting* to the external disturbances as opposed to *resisting* them. Therefore, for perturbations of large magnitude, adapting to the disturbances by executing feet swing motions to track footholds in the direction of the perturbations may not be responsive enough, especially, considering the joint velocity limits utilized in the optimization problem. We therefore introduce the recovery controller for following reasons:

1) The motion controller executes motion plans to track the desired feet positions generated by the RL footstep planner. These feet positions are only updated after each gait stride. With system disturbances, the CoM position may drift while the desired feet positions are still not updated. This would result in the ZMP drifting significantly from the support region eventually resulting in failure if no recovery motion is executed.
2) Perturbations of large magnitude cannot be fully addressed by adapting the motion plans. The ability to resist these disturbances allows us to better address such perturbations and also reduce the velocity command tracking errors, while only adapting to the disturbances when the perturbations are extremely large.

In this regard, the recovery controller (RC), operating at 400 Hz, is activated when any of the empirically tuned criteria relating to the robot state, described by q and u, exceed the limits. These are provided in the Appendix. RC is deactivated and the control switches back to the motion controller when each of the robot state criteria lies within its deactivation limit (also provided in the Appendix).

We express the recovery control problem in the framework of MDP directly building upon our prior work [23].

*1) State and Action:* For the purpose of recovery control over uneven terrain, we augment the state space defined in [23] with additional terrain information. The RC state, $\mathbf{s}_r \in \mathbb{R}^{201}$, is defined as $\mathbf{s}_r := \langle s_R, s_v, s_\mathbf{j}, s_M, s_b, s_\mathbf{b}, s_c, \mathrm{q}_j^* \rangle$ and RC action, $\mathbf{a}_r \in \mathbb{R}^{12}$, is given by the tuple $\mathbf{a}_r := \langle \mathrm{q}_j^* \rangle$. The RC state term definitions are consistent with those for FP and DAT.

*2) Reward Function:* The objective of the recovery controller is to stabilize the robot when perturbed. In this regard, we aim to maximize the stability margin, while also penalizing foot slippage and energy consumption. We also introduce a velocity tracking reward, such that, when utilized with a high-level goal planner, the RC executes motions in the direction of the desired velocity. This is based on the assumption that the goal planner prioritizes locomotion around accessible regions thereby resulting in fewer failures. Additionally, we utilize the velocity tracking property of the recovery controller for training. This is described in Section IV-C. The RC reward function is then given by

$$\begin{aligned}\mathbf{r}_R = 10\mathbf{r}_m + 3.5\mathbf{r}_v - 0.008 d_r \mathbf{r}_\tau - 0.25 d_r \mathbf{r}_{fh} - 3 d_r \mathbf{r}_\mu \\ - 0.1 d_r \mathbf{r}_e - 0.4 d_r \mathbf{r}_{\dot{\mathrm{q}}} - 0.06 d_r \mathbf{r}_{\ddot{\mathrm{q}}} - 0.01 d_r \mathbf{r}_{\ddot{f}}\end{aligned} \tag{23}$$

where joint velocity reward term $\mathbf{r}_{\dot{\mathrm{q}}} = \|max(|\dot{\mathrm{q}}_j| - 10, 0)\|^2$, feet clearance reward term $\mathbf{r}_{fh} = \sum_i (\mathrm{r}_{F,i_z} - 0.085)^2 \; \forall i, \mathrm{c}_{F,i} = 0, i \in \{0,1,2,3\}$, joint acceleration reward term $\mathbf{r}_{\ddot{\mathrm{q}}} = \|max(|\ddot{\mathrm{q}}_j| - 75, 0)\|^2$, feet acceleration reward term $\mathbf{r}_{\ddot{f}} = \|\ddot{\mathrm{r}}_F\|^2$ and $d_r$ is the reward scaling term increased as training progresses, updated by the rule $d_{r_{k+1}} \leftarrow d_{r_k}^{0.996}$ where $k$ refers to the policy iterations and $d_{r_0} = 0.3$. The rest of the reward term definitions are consistent with FP.

*3) Episode Termination Criteria:* We utilize the same termination criteria as employed for DAT.

*4) Policy Parameterization:* We parameterize the recovery control policy as a Gaussian distribution $\pi_{\theta_r}(\mathbf{a}|\mathbf{s}_r) := \mathcal{N}(\mathbf{a}|\mu_{\theta_r}(\mathbf{s}_r), \sigma_{\theta_r})$. The mean $\mu_{\theta_r}(\mathbf{s}_r)$ is generated using a neural network as a function of the state, while the standard deviation $\sigma_{\theta_r}$ is independent of the state and is utilized during training for exploration. The recovery control network architecture representing the function mapping state to the action mean is shown in Fig. 7. We use the RLlib implementation of proximal policy optimization (PPO) [44] to train the recovery control policy. The training is detailed in Section IV-C.

## IV. Training and Evaluation

This section details upon the training, testing and deployment setups for each of the RL policies. We trained these policies using the *RaiSim* [48] physics simulator for its extremely fast contact dynamics computation. Moreover, RaiSim supports online reloading of heightmaps enabling the user to simulate terrains making it suitable for training terrain-aware RL policies. We defer the reader to our previous work [23] for a detailed analysis on the preference for RaiSim over other widely accessible simulation platforms. Figure 7 represents an overview of the training approach.

### A. Footstep Planner

*1) Terrain Generation:* Similar to [32], we introduce a large set of terrains, 10k in our case, where each terrain is represented as $\mathbf{T} \in \mathbb{R}^{1001 \times 1001}$, into our simulation environment to obtain robust policies which can generalize to a wide domain of terrains. Our terrain generation procedure is similar to the one introduced in [49], and is further detailed in the Appendix.

*2) Actuation Dynamics:* As introduced in [21], we train an actuator network to model the actuation dynamics of the physical system to reduce the reality gap by emulating the impedance controller response as observed on the real robot. However, unlike in the case of [21], [23] where the actuators were modeled only for joint position targets with a predefined set of PD gains given by

$$\tau_j = \mathbf{j}_G(\mathrm{q}_j^*, \mathrm{q}_j, \dot{\mathrm{q}}_j) \; , \tag{24}$$

in this work, we trained the actuator network using supervised learning to obtain the effective actuation torques

$$\tau_j = \mathbf{j}_R(\mathrm{q}_j^*, \mathrm{q}_j, \dot{\mathrm{q}}_j^*, \dot{\mathrm{q}}_j, \tau_{j_{FF}}, K_p, K_d) \; . \tag{25}$$

Since the motion controller utilizes different set of PD gains for joints of legs with foot in close, open and contact-uncertainty states, we further include the PD gains as an input to the approximated actuator model. Note that, even for the same contact states, the actuators located on the joints of the same limb can have different PD gains. The actuator network thus maps the input $j_s \in \mathbb{R}^{21}$ to an approximation of the joint torque, $\tau_j \in \mathbb{R}$, observed on the physical system. The network architecture and training details are provided in the Appendix.

*3) Velocity command generation:* In order to reduce computational overhead, our terrain generation tool does not include collision detection between objects. As a result, these terrains contain overlapping objects. In some cases, the terrains also have significant deviations in height in small regions which cannot be traversed due to the limited size of the robots. In this regard, we generate velocity commands with

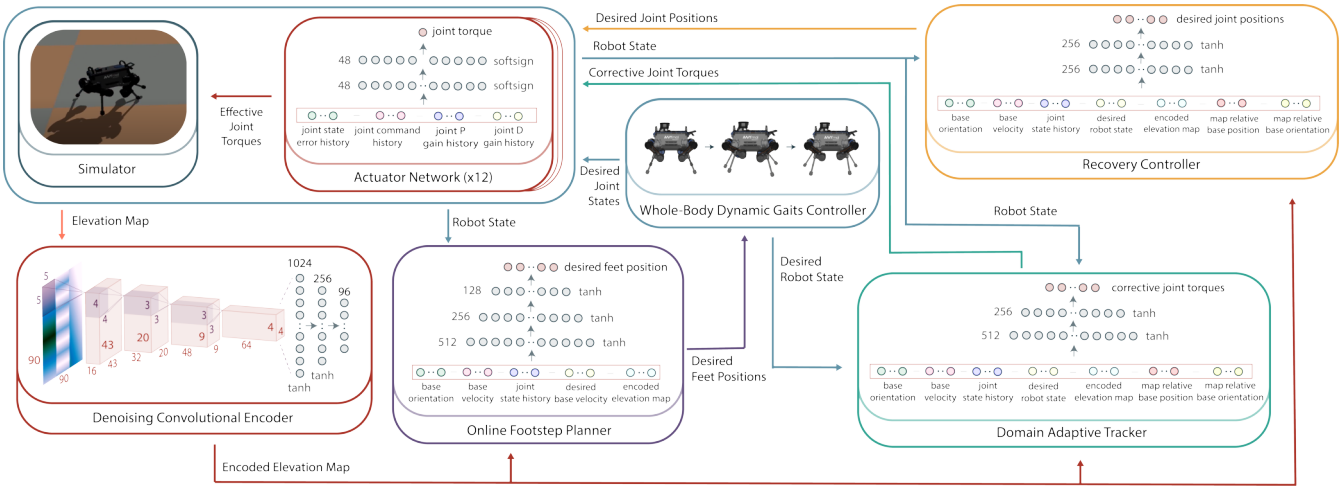



Fig. 7: Overview of the approach used to train the RL policies as part of our framework. We introduce the denoising convolutional encoder to embed the elevation map into a lower-dimensional representation. This, along with the robot states and desired velocity commands becomes the observation space for each of the RL policies. Since we train the RL policies in simulation, we use an actuator network to approximately model the actuation dynamics of the real robot to make sim-to-real transfer feasible.

the assumption that the robot tracks the velocity commands perfectly. We then check for height deviations in the terrain for the next 400 ms along the robot's expected trajectory. In case the height deviation exceeds the threshold of 0.4 m, we generate new velocity commands until the height deviation along the expected trajectory remains less than the threshold. This height deviation test is performed every 100 ms. The described velocity command generation approach significantly helps boost the performance of the footstep planning and domain adaptive tracking policies during training. Without introducing a strategic velocity command generator, we observed that our policies failed to converge often resulting in reward collapse.

*4) Domain randomization:* Our footstep planner is trained to adapt to the behavior of the motion controller which significantly depends on the dynamic properties of the system. In order to achieve a locomotion behavior robust to change in system properties, such as added mass, we further utilize domain randomization strategies in addition to terrain randomization. These are shown in Table V. Note that the sampling distributions presented in Table V relate to the limits which the motion controller is robust to. We observed that increasing the range of the distributions caused motion tracking failure. In this regard, we perform two training sessions for the footstep planner. Session 1 employs the distributions presented in Table V, while Session 2 utilizes the domain adaptive tracker to enhance the robustness of the motion controller and therefore employs the distributions presented in Table VI. Furthermore, since the perception sensors present on the real robot cannot be used to perceive the elevation along regions occluded by an obstacle, we apply blurring filter to the elevation map, as shown in eq. 10, to emulate interpolation around unobservable terrain features.

We also apply external perturbations along $\mathbf{e}_x^B$ and $\mathbf{e}_y^B$. However, since the motion controller is not robust to perturbations of high magnitude, we sample the external forces from a normal distribution $\mathcal{N}(0,10)\,\mathrm{N}$ clipped between $[-30,30]\,\mathrm{N}$ for a duration in the range $[1,4]\,\mathrm{s}$. We also perform gait randomization, where we select the trot gait with a probability of 0.7, crawl with a probability of 0.2, and amble & pace with a probability of 0.05 each.

TABLE V: Domain randomization properties utilized in Session 1 for training the footstep planning policy.

| Property | Sampling Distribution |
|---|---|
| Gravity, $g = 9.81\,\mathrm{m\,s^{-2}}$ | $\mathcal{U}(0.96g, 1.04g)$ |
| Actuation Torque Scaling | $\mathcal{U}(0.9, 1.1)$ |
| Robot Link Mass Scaling | $\mathcal{U}(0.93, 1.07)$ |
| Robot Link Length Scaling | $\mathcal{U}(0.97, 1.05)$ |

*5) Reinforcement Learning:* For Session 1, we employ an interactive direct policy learning approach to perform guided updates of our footstep planning policy, $\pi_{\theta_f}$, in order to increase sample efficiency during RL policy optimization. We initialize policy parameters, $\theta_f$, using Xavier initialization [50]. We consider a fixed-length episode of 1024 desired feet position samples, implying 1024 gait strides are executed in each episode. During training, we also perform locomotion over flat terrain. In this case, we use the motion controller to track the feet position generated by the policy and store it in a buffer $\mathcal{D}$. Additionally, we generate the desired feet position using the foothold optimizer of the Dynamic Gaits motion controller for the same robot state, $\mathbf{s}_f$, and add it to $\mathcal{D}$. After desired number of episodes used for locomotion over flat ground, we perform back-propagation through the footstep planning policy using the mean-squared error loss between the predicted and the expert samples. For uneven terrain, we utilize SAC to perform policy exploration so as to learn to generate feet position that avoid edges and slopes while ensuring the

stability of the robot. We alternate between policy optimization through exploration and the *interactive guided policy learning* during training. The ratio of episodes used to perform guided learning to the number of episodes used to perform policy exploration is reduced as training progresses. We refer to this approach as interactive guided policy optimization (IGPO). Algorithm 1 describes the IGPO approach used for training the footstep planning policy. It is important to note that since we use a curriculum based reward transition given by the factor $d_f$ as shown in eq. 17 and eq. 18, we continuously push new state transition tuples to the SAC replay buffer while discarding the older samples to ensure that the action-value function approximations utilized in SAC remain relevant to the latest objective weighted by $d_f$. Moreover, while performing guided learning, we set the target value of the state-dependent stochasticity based on $\sigma_{\theta_f}(\mathbf{s}_f)$ to zero.

**Algorithm 1** Interactive Guided Policy Optimization

**Input:** $\lambda_d$, $n_{eps}$, $l_{eps}$, $H$
*Initialize* $\mathcal{D} = \{\}$, $\alpha = 1$
**for** $k = 0, 1, 2 \ldots H$ **do**
  $\theta_f, \mathcal{D} = \text{GUIDEDLEARNING}(\theta_f, \mathcal{D}, \alpha n_{eps}, l_{eps})$
  $\theta_f = \text{POLICYOPTIMIZATION}(\theta_f, (1-\alpha) n_{eps}, l_{eps})$
  $\alpha = e^{\lambda_d t}$
**end for**
**function** GUIDEDLEARNING($\theta_f$, $\mathcal{D}$, $n_{eps}$, $l_{eps}$)
  **for** $i = 0$ **to** $n_{eps} - 1$ **do**
    **for** $t = 0$ **to** $l_{eps} - 1$ **do**
      Sample $a \sim \pi_{\theta_f}(a|s)$
      Generate $a^*$ using DYNAMICGAITS(s)
      $\mathcal{D} = \mathcal{D} \cup \{(s, a, a^*)\}$
      $s, r = \text{RLENVIRONMENTSTEP}(a)$
    **end for**
    Sample batch of $\{(s, a_j, a_j^*)\}$ from $\mathcal{D}$
    Update $\theta_f$ by minimizing $\sum_j ||a_j - a_j^*||^2$
  **end for**
  **return** $\theta_f$, $\mathcal{D}$
**end function**
**function** POLICYOPTIMIZATION($\theta_f$, $n_{eps}$, $l_{eps}$)
  **for** $i = 0$ **to** $n_{eps} - 1$ **do**
    $\tau = \{\}$
    **for** $t = 0$ **to** $l_{eps} - 1$ **do**
      Sample $a \sim \pi_{\theta_f}(a|s)$
      $s, r = \text{RLENVIRONMENTSTEP}(a)$
      $\tau = \tau \cup (s, a, r)$
    **end for**
    Update $\theta_f$ using SAC
  **end for**
  **return** $\theta_f$
**end function**

As training progresses, we increase the complexity of the RL environment by scaling the maximum terrain elevation. As introduced in Section IV-A1, we represent the elements of $\mathbf{T}$ in the range of $[0, h_{max}]$ where $h_{max} = 2\,\text{m}$. During training, we scale $h_{max}$ using $d_{hf}$ which is updated by the rule $d_{hf_{k+1}} \leftarrow d_{hf_k}^{0.9992}$ where $k$ refers to the policy iterations and $d_{hf_0} = 0.01$.

For Session 2, we do not perform guided learning and only utilize SAC to make footstep planning more robust to the changes in modeled system dynamics. We also set each of the curriculum factors to one. This additional training allows us to utilize the same footstep planner trained with ANYmal B for deployment on ANYmal C. The training hyperparameters and policy convergence iterations are provided in the Appendix.

*6) Testing:* For the purpose of evaluation, we utilize the stochastic policy in a deterministic manner. Therefore, we consider the $\mu_{\theta_f}(\mathbf{s}_f)$ as the desired action, ${}_H r^*_{HF_{xy}}$, and discard $\sigma_{\theta_f}(\mathbf{s}_f)$. We detail the test results in Section V-A.

*7) Deployment:* The observations utilized for the footstep planner are accessible on the physical robots. During deployment on the real robots, we observe a computational delay of approximately 35 ms during elevation map extraction and encoding. The elevation map processing is therefore executed in a parallel thread while the motion tracking controller runs in the main thread at 400 Hz. The robot maintains the previous desired footholds while the forward pass through the elevation map encoder and the footstep planning policy is performed, after which, the new motion plans for the updated footholds are generated and tracked. This methodology remains consistent for both ANYmal B and C. A forward pass through the FP policy network requires approximately 195 µs on both the robots.

### B. Domain Adaptive Tracker

We utilize the terrain generator, actuator network and velocity command generator introduced in Section IV-A for training the DAT policy. In addition, we use the footstep planning policy obtained after training Session 1 to train our domain adaptive tracking policy.

*1) Domain Randomization:* The randomization parameters and the corresponding sampling distributions representing the limits are provided in Table VI. During training of the DAT policy, the parameter randomization is approached using a curriculum learning setup where initial limits of the distributions are set to 1. The lower and upper limits are then linearly expanded after every policy iteration, $k$, based on the rule $l_{l,k} = max(\frac{k}{2000}(l_{min} - 1) + 1, l_{min})$ and $l_{u,k} = min(\frac{k}{2000}(l_{max} - 1) + 1, l_{max})$ respectively where the $l_{min}$ and $l_{max}$ are the lower and upper limits presented in Table VI. We also introduce external perturbations and gait randomization as done for the footstep planner.

TABLE VI: Domain randomization properties utilized for training the domain adaptive tracking policy. Here $\tau_j^*$ is the expected torque output of the impedance controller.

| Property | Sampling Distribution |
|---|---|
| Gravity, $g = 9.81\,\text{m}\,\text{s}^{-2}$ | $\mathcal{U}(0.96g, 1.04g)$ |
| Actuation Torque Scaling | $\mathcal{U}(0.85, 1.15)$ |
| Robot Link Mass Scaling | $\mathcal{U}(0.9, 1.10)$ |
| Robot Link Length Scaling | $\mathcal{U}(0.85, 1.15)$ |
| Actuation Damping Gain $G$ such that $\tau_{j,t} = G\tau_{j,t}^* + (1-G)\tau_{j,t-1}$ | $\mathcal{U}(0.8, 1)$ |

*2) Reinforcement Learning:* We employ TD3 for training the DAT policy allowing us to perform constrained exploration. We observed that the stochasticity associated with policies trained using SAC and PPO required exhaustive hyperparameter tuning without which the exploration resulted in introduction of large corrective torques which caused the whole-body controller to significantly drift from the generated motion plans causing aggressive recovery and eventual failure. This inhibited learning since the corrective torques mostly resulted in episode termination. We therefore used TD3 to initially perform constrained exploration using uncorrelated Gaussian sampling. During training, we scale the standard deviation of the standard Gaussian distribution according to the rule $d_{a,k+1} \leftarrow d_{a,k}^{0.9994}$ where $k$ refers to the elapsed policy iterations and the initial scaling $d_{a,0} = 0.01$. Additionally, we scale $\delta\tau_j$ by a factor of 40 before adding it to the feed-forward torque generated by the whole body controller.

As in the case of training the footstep planner, we use the same curriculum approach of scaling the maximum terrain elevation. We used the standard RLlib TD3 implementation for reinforcement learning. The hyperparameters are provided in Appendix.

*3) Testing:* Since we obtain a deterministic policy, we do not perform any modifications to the neural network and directly employ it as a function mapping state to action. The test results are presented in Section V-B.

*4) Deployment:* The DAT is executed along with the whole-body controller in the main control loop at 400 Hz. The generated corrective torques are then added to the output of the whole-body controller, after which the desired joint states are forwarded to the actuator. The DAT remains active when the recovery controller is deactivated. A forward pass through the DAT policy network requires approximately 66 µs.

### C. Recovery Controller

Our recovery controller training setup is directly built on top of our previous work [23]. However, unlike in the case of our previous work, we do not employ a guided policy optimization strategy, since the guided learning focused on velocity command tracking as opposed to recovery control, and instead utilize the training approach presented in [21] performing reinforcement learning using PPO.

For training the RC policy, we employ the terrain generator introduced in Section IV-A1. Since we perform joint position control with the recovery policy, we also use the actuator network to model the actuation dynamics for fixed $K_p = 50$ and $K_d = 0.1$ with zero velocity target. This is described in eq. 24. We defer the reader to the original work [21] for detailed description of the joint position target actuator network.

*1) Domain Randomization:* We use the same randomization strategy, including curriculum learning, as used for DAT for training the recovery controller.

*2) Reinforcement Learning:* Since our objective is to learn a recovery controller which can stabilize to perturbations in addition to performing state recovery, we introduce external perturbations in our training environment. However, in order to encompass a wide possibility of robot states when the recovery controller is activated, we initially train the RC policy to track reference velocity commands. We then transition from this objective to that of recovery control. For this, we initially learn to locomote over flat ground after which we introduce larger external perturbations while also scaling up the terrain elevation. The approach is detailed in Algorithm 2.

**Algorithm 2** Recovery Control Training Objective Transition

**Input:** $n_{eps}$, $l_{eps}$, $H$
*Initialize* $e_{max} = 1e-4$, $f_{max} = 0.01$, $t_{f_{max}} = 0.01$, $t_f = 0$, $t_a = 0$
**for** $k = 0, 1, 2 \ldots H$ **do**
  **for** $i = 0$ **to** $n_{eps} - 1$ **do**
    $t_{vel}, s = \text{RLENVIRONMENTRESET}(e_{max})$
    **for** $t = 0$ **to** $l_{eps} - 1$ **do**
      **if** $t >= t_{vel}$ **then**
        Generate new velocity command
        Sample $t_{vel} \sim \mathcal{U}(1,3)$
      **end if**
      Sample action $a \sim \pi_{\theta_r}(a|s)$
      **if** $t >= t_a$ and $t < t_a + t_f$ **then**
        Apply external force $f$
      **else if** $t > t_a + t_f$ **then**
        Sample external force, $f \sim \mathcal{N}(0, 45 f_{max})$
        Sample $t_a \sim \mathcal{U}(t, t+2)$ and $t_f \sim \mathcal{U}(0, 4 t_{f_{max}})$
      **end if**
      $s, r = \text{RLENVIRONMENTSTEP}(a)$
      **if** $s$ is a terminal state **then**
        $t_{vel}, s = \text{RLENVIRONMENTRESET}(e_{max})$
      **end if**
    **end for**
  **end for**
  Update $\pi_{\theta_r}$ using PPO
  $e_{max} \leftarrow e_{max}^{0.9996}$, $f_{max} \leftarrow f_{max}^{0.9984}$, $t_{f_{max}} \leftarrow t_{f_{max}}^{0.997}$
**end for**
**function** RLENVIRONMENTRESET($e_{max}$)
  Sample terrain and scale elevation by $e_{max}$
  Randomize robot pose
  Randomize $\mathrm{q}_j$ where $max(|\mathrm{q}_j - \mathrm{q}_j^n|) < 0.3$
  Randomize $\dot{\mathrm{q}}_j$ where $max(|\dot{\mathrm{q}}_j|) < 1.0$
  Update state $s$ and set $t_{vel} = 0$
  **return** $t_{vel}$, $s$
**end function**

*3) Testing:* As in the case of FP, we utilize the stochastic policy in a deterministic manner. In this regard, the action $\mathbf{a}_r = \mu_{\theta_r}(\mathbf{s}_r)$. The desired joint positions, $\mathrm{q}_j^* = \mathbf{a}_r$ are then forwarded to the actuator to generate the effective actuation torques. The results are presented in Section V-C.

*4) Deployment:* We query the robot state at each control step executed at 400 Hz to activate RC according to the limits described in Section III-E. When active, RC is run in the main control loop to generate the desired joint positions. While the RC remains active, the footstep planner, motion controller and domain adaptive tracker are deactivated. A forward pass through the RC policy network requires approximately 42 µs.

## V. Results and Discussion

This section details the results obtained for each of the RL policies introduced in our framework. Note that, unless otherwise stated, the results presented here relate to the ANYmal B quadruped.

### A. Footstep Planning

We evaluate the footstep planning policy, obtained after the 2 training sessions, by measuring its success rate when traversing different terrains. We set up our simulation platform to perform 1000 runs for each of the 3 footstep planners navigating using 2 different gaits - trot and crawl. The 3 planners are our RL footstep planner, a baseline perceptive planner and a baseline blind planner. In the case of the baseline planners, we use the original foothold optimizer of dynamic gaits to generate $r_F^d$ for tracking the desired base velocity commands. We further modify $r_{F_z}^d$ by extracting the terrain elevation at the touch-down location directly from $\mathbf{M}_H$ (including for the blind controller). For the case of perceptive controller, we additionally adjust $r_{F_{xy}}^d$ up to a radial deviation of 0.05 m so as to minimize the cost $\mathbf{C}_\mathbf{M}(\hat{r}_{F_{xy},i}) + 2.5\|\hat{r}_{F_{xy},i} - r_{F_{xy},i}^n\|^2$ for each of the feet. For this optimization, we compute the cost by sampling $\hat{r}_{F_{xy},i}$ along the polar radial coordinate with a resolution of 0.005 m and a polar angular resolution of $\pi/18$ rad. We test the performance over 1000 different terrains of varying dimensions for each of the 4 object types - stairs, bricks, wave and unstructured ground as shown in Fig. 8. Each of these objects are located at the center of a $5 \times 5\text{m}^2$ terrain, with the length of the object, along the direction of the robot's frontal axis, varying between 2.0 m to 3.6 m. We consider the following terrain properties.

1) The amplitude for the unstructured terrain is varied between $[0.0125, 0.025]$ m.
2) For stairs, the number of steps range between $[3, 8]$. The maximum terrain elevation is varied between $[0.25, 0.8]$ m using clipped normal distribution, $\mathcal{N}(0.3, 0.1)$ m.
3) The wave terrain is generated using a sine function along its length with an amplitude range of $[0.05, 0.1]$ m. The period of the sine function ranges between $\pi/2$ to $\pi$.
4) For brick terrain, the bricks are made of deformed $10 \times 10\text{cm}^2$ patches with extrusions $\{-h, 0, h\}$ where $h \in [0.02, 0.05]$ m.

The results obtained for traversal over these terrains are presented in Movie S1.

For each run, we set the robot at the origin and generate a forward velocity command of 0.3 m/s. We denote the run as a success if the robot traverses a distance of 4 m, and regard it as a failure if any of the FP termination criteria is observed. The success rates obtained for each of the experiments are represented in Table VII. Figure 9 presents the success rate for traversing each terrain for each of the tested controllers with two gaits.

The RL footstep planner was observed to exhibit preference for wider stance. This directly corresponds to a higher stability margin, a term introduced in the reward function during training. We measured ${}_H r_{HF}$ after every gait stride. For traversal over stairs, the absolute mean foot position in frame $H$ was 0.35 m and 0.20 m along its $x$ and $y$ axes, similar to the expected nominal stance value of 0.35 m and 0.20 m. However, in the case of unstructured terrain, we computed the mean feet position to be $(0.34, 0.23)$m, $(0.35, 0.21)$m for the wave terrain, and $(0.34, 0.22)$m for the bricks terrain.

Figure. 9e represents the success rate as a function of step height and depth evaluated using *kernel density estimation* (KDE).

The extrusions in the brick terrains often caused leg blockages. This was prominent for the blind planner. The ability of the perceptive planner to minimize distance from the edges in the terrain significantly reduced the problem with leg blockage. However, the difference in the feet height resulted in the robot toppling over in most cases. With the RL planner, we observed a wide stance and preference for foot placement along regions which minimized the deviation in feet height. For the case of the blind planner, we computed a mean deviation between the maximum and minimum foot height to be 0.074 m. For the perceptive planner the mean height deviation was 0.083 m, and 0.057 m with the RL planner. Figure 9f illustrates the success rate computed for different brick heights. In this case, the brick height only corresponds to the extrusion of the terrain

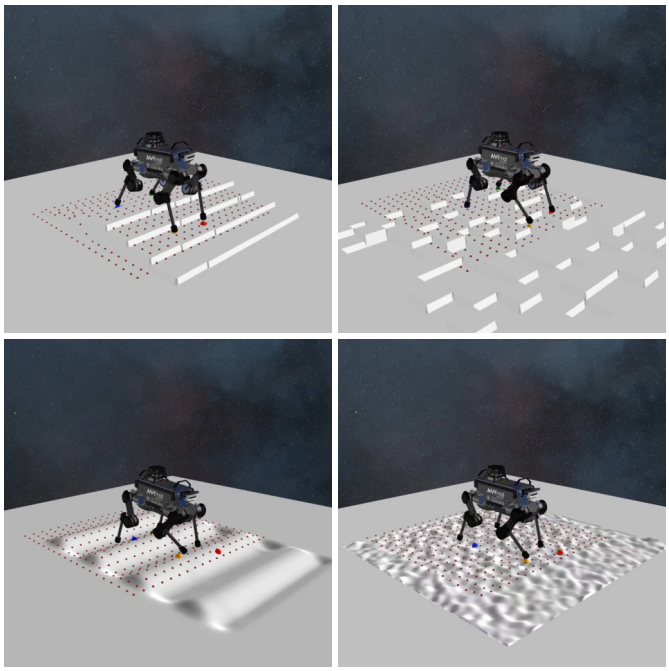
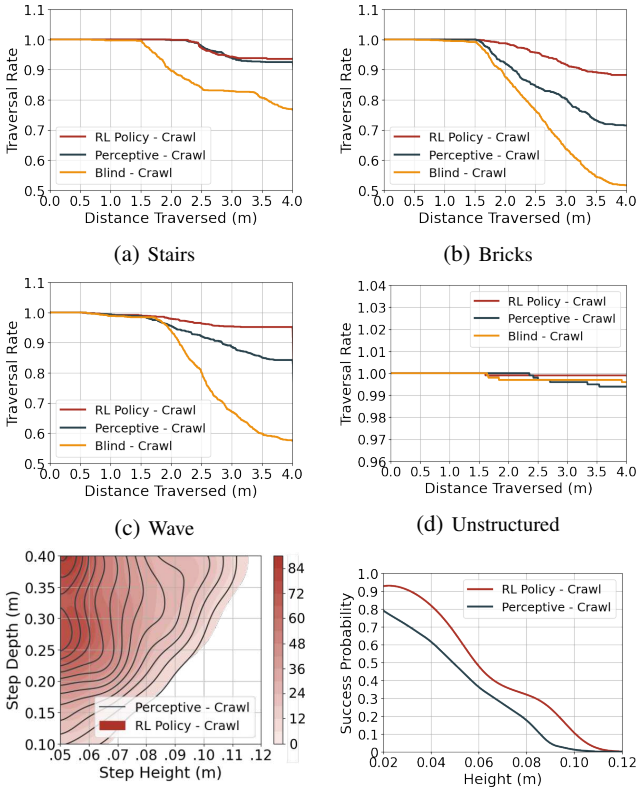


Fig. 8: **(Top)** the stairs (left) and brick (right) terrains. **(Bottom)** the wave (left) and unstructured (right) terrains.

TABLE VII: Success rates of policies measured for locomotion over different terrains generated for 1000 runs each.

| | Crawl | | | Trot | | |
|---|---|---|---|---|---|---|
| | **Blind** | **Perceptive** | **RL Policy** | **Blind** | **Perceptive** | **RL Policy** |
| **Unstructured** | 99.6 | 99.4 | **99.9** | **99.7** | 99.6 | 99.6 |
| **Stairs** | 77.0 | 92.5 | **93.6** | 72.2 | 92.3 | **92.7** |
| **Wave** | 58.1 | 83.6 | **94.4** | 55.7 | 81.5 | **93.0** |
| **Bricks** | 52.1 | 71.5 | **88.2** | 51.1 | 70.9 | **85.3** |




(a) Stairs (b) Bricks (c) Wave (d) Unstructured

(e) Estimate of success rate for traversal over stairs with varying depth and height

(f) Estimate of success rate for terrains with varying brick height

Fig. 9: **(a-d)** represent the traversal rate of each of the planning policies for the different types of terrains used for evaluation. This corresponds to the rate of success observed for the distance traversed. **(e)** is the plot representing the success rate for traversal over stairs with varying step height and depth. **(f)** represents the probability of success observed for brick terrain of varying brick heights.

implying that the maximum deviation in feet height is twice the height of the bricks. We observed that the RL planner fails to traverse brick terrains with maximum elevation of higher than 0.12 m (corresponding to a maximum height deviation of 0.24 m). This mainly occurs due to the limitations of the physical dimensions of the quadruped.

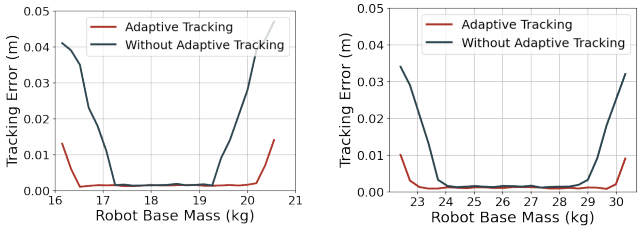
We also compared the generalization capability of the footstep planner obtained after training Session 1 and Session 2. We performed the experiments discussed above using the ANYmal C quadruped, adapting the motion controller for the parameter description of ANYmal C. The corresponding success rates are presented in Table VIII. As expected, we observe a better ability to generalize to a different platform with the policy trained using significant domain randomization.

### *B. Domain Adaptive Tracking*

In our experiments, the tracking error associated with the whole-body controller was negligible when the robot's base mass, of 18.3546 kg, was scaled in the range $[0.95, 1.05]$.

TABLE VIII: Success rates obtained for locomotion over different terrains using the ANYmal C quadruped comparing the RL footstep planner performance after training Session 1 and Session 2.

| | Crawl | | Trot | |
|---|---|---|---|---|
| | **Session 1** | **Session 2** | **Session 1** | **Session 2** |
| **Unstructured** | 99.9 | **99.9** | 99.4 | **99.5** |
| **Stairs** | 93.0 | **93.4** | 90.5 | **91.9** |
| **Wave** | 85.7 | **91.2** | 82.4 | **88.4** |
| **Bricks** | 76.3 | **85.8** | 71.6 | **82.2** |




(a) ANYmal B (b) ANYmal C

Fig. 10: The absolute mean tracking error observed before termination for varying physical parameters of the robot comparing the effects of the domain adaptive tracking policy.

However, when we decreased the mass by a factor below 0.94, or increased it beyond 1.06, the motion controller became unstable resulting in increased oscillations. This was caused due to the whole-body controller aggressively correcting for imprecise motion tracking. Introducing DAT helped stabilize whole-body motion tracking. We were able to scale the base mass in the range of $[0.84, 1.17]$ without increasing the amplitude of the oscillations. Figure 10a illustrates the $p_{CoM_x}$ tracking error observed before episode termination for different robot base masses. Here, the episode termination criteria include traversing a distance of 5 m or failure of whole-body control.

We also tested our domain adaptive tracking policy, without retraining, on the simulated model of the ANYmal C quadruped. Note that, while running experiments on the simulated ANYmal C, we increased the joint torque limit from $\pm 40\,\mathrm{N\,m}$ for the case of ANYmal B to a limit of $\pm 80\,\mathrm{N\,m}$. The tracking error measured for varying base mass of the ANYmal C quadruped is represented in Fig. 10b. With an original base mass of 26.3732 kg, the motion controller could track the forward velocity command of $0.3\,\mathrm{m/s}$ even when we scaled this mass in the range $[0.91, 1.10]$. When we introduced the domain adaptive tracking policy, we were further able to extend this range to $[0.81, 1.19]$.

We performed another test on the simulated ANYmal C robot. However, in this case, we used the parameter description of the ANYmal B quadruped. Therefore, the motion controller generated motion plans and control behavior for the ANYmal B quadruped even though the controller was used on ANYmal C. Note that, the domain adaptive policy was never trained using ANYmal C. Figure 11 represents the velocity tracking error observed in 1M simulation samples for randomly generated desired base velocity commands in the range of $[-0.3, 0.3]\mathrm{m\,s^{-1}}$ and $[-0.3, 0.3]\mathrm{rad\,s^{-1}}$ for the

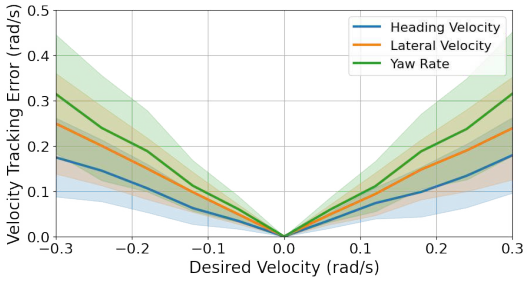



Fig. 11: Velocity tracking error observed for the domain adaptive tracking policy when running the motion controller on ANYmal C with the description parameters of ANYmal B.

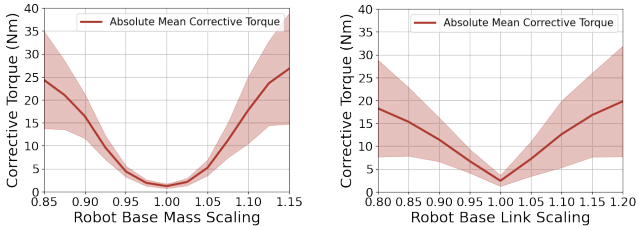



Fig. 12: The absolute mean corrective torques generated by the DAT to minimize motion plan tracking errors for ANYmal B base mass (left) and link (right) scaling. The color bands represent one standard deviation.

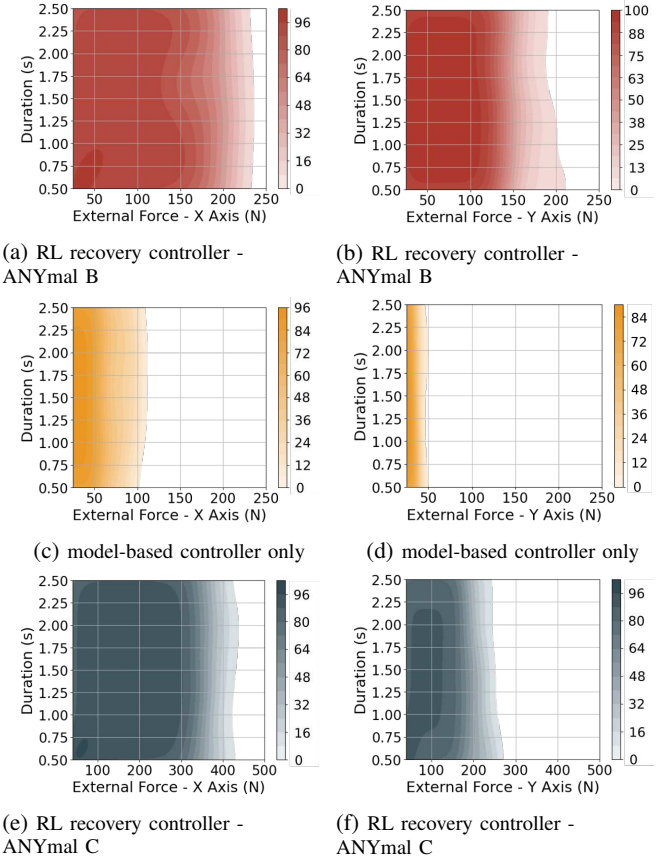
linear and angular velocities respectively. Note that, in this case of domain adaptive tracking, we did observe an oscillatory behavior in the limbs while tracking the base velocity commands. This was largely due to the whole-body controller outputting desired joint states which were then aggressively corrected by the domain adaptive tracking policy. Moreover, the domain adaptive tracker failed to stably track absolute velocity commands beyond $\pm 0.4\,\mathrm{m\,s^{-1}}$ for heading velocity, $\pm 0.3\,\mathrm{m\,s^{-1}}$ for lateral velocity, and $\pm 0.5\,\mathrm{rad\,s^{-1}}$ for yaw rate.

Figure 12 represents the mean and standard deviation of the absolute corrective joint torque generated by DAT to address ANYmal B base mass and link length scaling for locomotion on flat ground for a heading velocity command of $0.5\,\mathrm{m\,s^{-1}}$.

We have also provided a set of examples employing the domain adaptive tracker in Movie S2.

### C. Recovery Controller

We tested the behavior of the RL recovery control policy and the motion controller for a range of external forces applied to the robot's base along the heading and lateral axes for duration between $[0.5, 2.5]\,\mathrm{s}$. The observations representing the KDE of the recovery rate as a function of the magnitude and duration of the external force for each of the controllers are as shown in Fig. 13. The color-bar labels represent the success probability of the robot to stabilize following external perturbations. We measure this for both ANYmal B and ANYmal C and observe that compared to the motion controller which can handle perturbations of up to approximately $100\,\mathrm{N}$ along the x-axis for the ANYmal B quadruped, the RL policy can handle external force of up to approximately $225\,\mathrm{N}$ and




(a) RL recovery controller - ANYmal B

(b) RL recovery controller - ANYmal B

(c) model-based controller only

(d) model-based controller only

(e) RL recovery controller - ANYmal C

(f) RL recovery controller - ANYmal C

Fig. 13: Bivariate kernel density estimates representing the success probability of the control policies to stabilize upon external perturbations along the frontal (x) and the laeral (y) axes. Note that for **(e-f)** the x-axis scale of the plots is different.

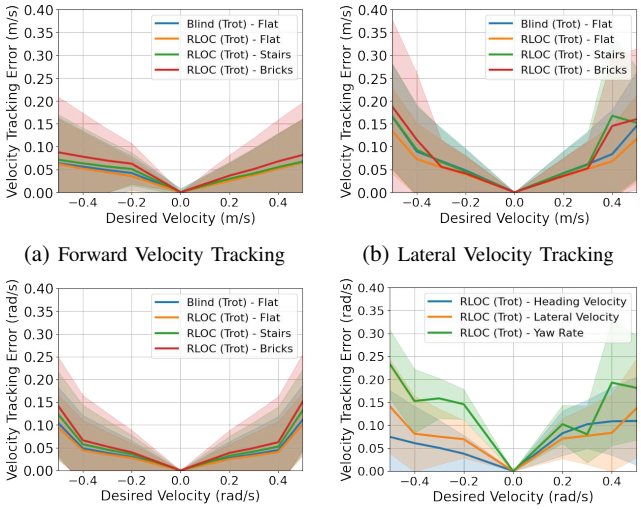
$400\,\mathrm{N}$ for ANYmal B and ANYmal C respectively. It is important to note that for the case of ANYmal C, we increased the position tracking gain to 60 compared to 45 for ANYmal B and also increased the joint torque limits to $\pm 80\,\mathrm{N\,m}$ compared to $\pm 40\,\mathrm{N\,m}$ for ANYmal B. The results obtained are also demonstrated in Movie S3.

### D. RLOC

Figure 14 represents the tracking error observed for the RLOC framework for traversal over different types of terrains. The improved tracking performance over the motion controller was a direct result of introduction of DAT. Removing DAT resulted in a very similar performance as that of the motion controller.

The experimental setups for each of these terrains included the generation of a velocity command in the range of $[-0.5, 0.5]\mathrm{m/s}$ for heading and lateral velocity tracking, and in the range of $[-0.5, 0.5]\mathrm{rad/s}$ for desired yaw rate tracking. Each episode comprised tracking of a set velocity command for a duration of $5\,\mathrm{s}$. We performed 100 runs for each command.

In order to quantify the effects of the recovery control policy, we performed experiments using the bricks and wave




(a) Forward Velocity Tracking

(b) Lateral Velocity Tracking

(c) Yaw Rate Tracking

(d) Velocity Tracking - Gazebo

Fig. 14: **(a-c)** represent the velocity tracking error measured for locomotion over even and uneven terrain for the ANYmal B quadruped tested in the RaiSim simulator. **(d)** represents the velocity tracking error on flat ground observed for the RLOC framework in the Gazebo simulator (with different actuation dynamics).

terrains for the trotting gait as introduced for evaluation of the FP. This is further presented in Movie S3. Introduction of the recovery controller resulted in improved success rates. The results are summarized in Table IX.

TABLE IX: Improvement in success rate (SR) observed after introducing the recovery controller (RC) for traversal over bricks and wave terrains. The episodes here refer to the number of episodes (out of 1000) in which the RC was called. Moreover, the recovery rate (RR) represents the successful recovery performed using RC. The RR and SR are expressed as %.

| | **RC Calls** | **Episodes** | **RR** | **SR w/o RC** | **SR with RC** |
|---|---|---|---|---|---|
| **Bricks** | 184 | 131 | 62.5 | 85.3 | 90.2 |
| **Wave** | 39 | 22 | 71.8 | 93.0 | 94.8 |

As previously mentioned, we used the RaiSim simulator to train and evaluate the performance of our control framework. In order to be able to port it to the physical robot, we further evaluated the velocity tracking performance of our framework in a slower but higher resolution *Gazebo* simulator [51], employing the ODE physics engine [52], as presented in Fig. 14d. Figure 1 presents examples of the experiments performed on the real ANYmal B quadruped. The corresponding video clips are included in Movie 1. Figure 15 represents the snapshots of the robot ascending and descending a staircase. In our experiments with the real robot, we observed motion behavior similar to the behavior in RaiSim especially for tests on simpler terrains such as a staircase. In the case of bricks, however, the state-estimation drift, sensor noise and the fast but less accurate nearest-neighbor interpolation technique used to obtain the elevation map along occluded regions sometimes resulted in the elevation map being distorted especially along the edges, and also slightly shifted with respect to the robot. The distortion of the elevation map is illustrated in Fig. 4 and Fig. 16 which represents a snapshot of the real ANYmal B quadruped tested on a terrain comprising bricks, and the corresponding visualization of the robot state and the elevation map. Due to this blurring and slight shifting of the elevation map, we observed that the robot stepped close to the edges especially in complex environments composed of a large number of smaller objects, such as bricks. The visualization of the elevation map for the test shown in Fig. 16 is also presented in Movie S4.

We also transferred the RLOC framework to the real ANYmal C quadruped for both indoor and outdoor locomotion as shown in Fig. 2. We were able to locomote outdoors over grassy and muddy terrain, and slopes. We observed a similar behavior as in the case of ANYmal B, however, the ANYmal C offered an advantage of higher joint torque limits enabling us to ascend steeper slopes compared to ANYmal B. For this transfer, the only changes included the robot parameter description for the motion controller and the impedance controller gains. The experiments performed with the real ANYmal B and ANYmal C are presented in Movie 1 and Movie S6.

In our experiments, we observed that ascending steep staircases where the ANYmal B base pitch exceeds $\pi/4$rad with respect to the horizontal frame results in failure to track heading velocity commands where the robot is unable to move forward. This is partially attributed to the joint torque limits while also resulting from the CoM motion planner that uses ZMP, implicitly assuming constant base height and attitude. For ANYmal C, we observe velocity command tracking failure beyond a pitch angle of approximately $\pi/3$rad. This was the case for both simulation and real robot. Additionally, uneven terrain locomotion failures also occurred when the deviation in terrain elevation was greater than the shank link length often resulting in collision of the knee with the terrain.

RLOC also enables the user to switch between locomotion gaits online. This is demonstrated in Movie S4.

## VI. Conclusion and Future Work

We presented an approach which bridges the gap between policies trained using RL and controllers developed using model-based optimization methods.

Our modular framework allowed us to greatly improve the success of traversal over complex terrains. For trotting over the bricks environment, as presented in Section V, we showed that our hybrid approach, RLOC, was able to increase the success rate from 71.5% to 90.2% in comparison to our baseline perceptive controller. We showed that we were able to traverse reliably (i.e. with a success rate of 90% or above over 4m) and with a dynamic gait (trot at an effective velocity of approximately $0.25\mathrm{m\,s^{-1}}$ on the real robot) over uneven terrain even though the model-based locomotion controller was primarily developed and tuned for quasi-flat terrains.

To address ascending steep stairs and slopes, in our future work we aim to learn a CoM and foot motion planner.

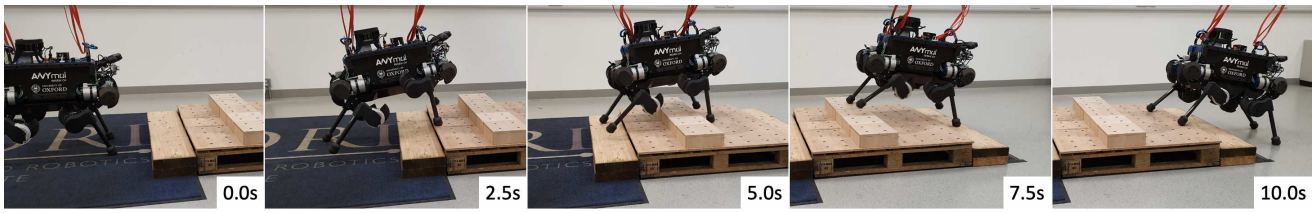



Fig. 15: Snapshots of the ANYmal B quadruped walking over a staircase using our RLOC framework.

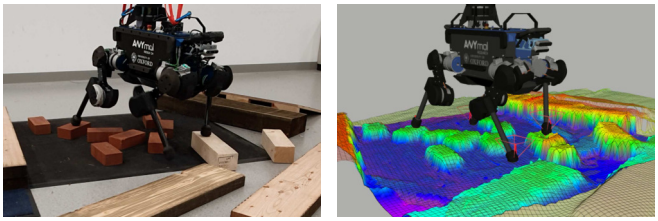


Fig. 16: ANYmal B quadruped tested on a terrain comprising bricks (left); visualization of the corresponding robot state and elevation map (right).

Another direction for future work includes obtaining a hybrid long and short horizon planning policy to avoid obstacles and perform CoM and feet motion adaptations for traversal over extremely complex and non-stationary terrains.

## NOMENCLATURE

| Notation | Definition |
|---|---|
| ${}_W r_{WB}$ or $r_B$ | translation of frame $B$ represented frame $W$ |
| $\mathbf{R}_{WB}$ or $\mathbf{R}_B$ | rotation matrix of $B$ represented in $W$ |
| $\mathbf{e}_z^W$ | $z$-axis of $W$ |
| $\mathrm{q}_j$ | joint positions |
| ${}_W \mathrm{v}_{WB}$ or $\mathrm{v}_B$ | linear velocity of $B$ measured and expressed in $W$ |
| ${}_W \omega_{WB}$ or $\omega_B$ | angular velocity of $B$ measured and expressed in $W$ |
| $\dot{\mathrm{q}}_j$ | joint velocities |
| $\mathbf{R}_{B_z}$ | $z$ decomposition of $\mathbf{R}_B$ |
| $\tau_j$ | joint torques |
| $\cdot^*$ | desired quantity |
| $\Phi$ | feet motion parameterization tuple |
| $r_F^l$, $r_F^d$, $r_F^h$ | feet lift-off, touch-down and clearance positions |
| $\mathrm{c}_F$ | feet contact state |
| $t_F^l$, $t_F^d$ | feet lift-off and touch-down times |
| $\mathbf{M}_H$ | robocentric terrain elevation represented in frame $H$ |
| $\mathrm{Z}_{\mathbf{M}_H}$ | latent embedding of $\mathbf{M}_H$ |
| $\mathcal{C}$ | 2d convolution operator |
| $\mathbf{s}_f$, $\mathbf{a}_f$, $\mathbf{r}_F$ | footstep planner state, action and reward |
| $\mathbf{s}_d$, $\mathbf{a}_d$, $\mathbf{r}_D$ | domain adaptive tracker state, action and reward |
| $\mathbf{s}_r$, $\mathbf{a}_r$, $\mathbf{r}_R$ | recovery controller state, action and reward |
| $\mathbf{c}^*$ | desired velocity command |
| $\pi_{\theta_f}, \pi_{\theta_d}, \pi_{\theta_r}$ | footstep planning, domain adaptive tracking and recovery control policies |
| $\mathcal{S}_m$ | stability margin function |
| $\mathbf{C}_\mathbf{M}$ | terrain cost map operator |
| $B_{\mathbf{M}_H}$ | elevation map update frame |

## APPENDIX

### A. Stability Margin

In our work, we define the stability margin as the distance between the Instantaneous Capture Point (ICP) [13] and the edges of the feasible region as introduced in [39]. The margin is positive when the ICP belongs to the feasible region and negative when it lies outside of it. A positive margin corresponds to a stable robot state whereas a negative margin implies that the robot is unstable and will eventually fall unless a recovery action is introduced. If the margin is bounded, however, it implies that the ICP does not diverge away from the ZMP and that the robot can still manage to recover to a stable configuration before the robot eventually falls. This is, for example, what happens for dynamic gait patterns such as trot and pace: in such cases the stability margin will be mostly negative but the robot will not fall if it manages to maintain a suitable lower bound value of stability margin at all the time.

The stability margin term, introduced in the RL reward functions, ensures the robot avoids unnecessary linear and angular accelerations and places its feet at the correct time instant and location. A maximization of the stability margin over a crawl gait (only one swing foot at the time), for example, will result in the policy trying to enlarge the support of the robot as much as possible corresponding to a larger feasible region.

The computation of the stability margin requires the estimation of the feasible region, which can be done by means of an iterative projection algorithm up to an arbitrary tolerance [53]. This procedure requires the solution of multiple linear programs, and even for low tolerance values, requires an estimated 0.5 ms for an optimized C++ implementation. For this reason we employ an MLP as an approximation of the analytical solution to perform extremely fast stability estimation. In our C++ implementation, we can perform a forward pass through this MLP in less than 4 µs making it appealing for use in sample-inefficient RL training.

We use an MLP with 2 hidden layers of size $\{48, 48\}$ which maps a 47-dimensional input into a scalar stability margin. The network input, $\mathbf{s}_m$, is given by the tuple

$$\mathbf{s}_m = \langle \mathbf{e}_z^B, {}_B v_{WB}, {}_B \omega_{WB}, {}_B \dot{\omega}_{WB}, \overrightarrow{F}_B, \overrightarrow{\tau}_B, \mathbf{T}_\mu, {}_B r_{BF}, \mathrm{c}_F, N_F \rangle$$

where $\overrightarrow{F}_B \in \mathcal{R}^3$ is the external force on base, $\overrightarrow{\tau}_B \in \mathcal{R}^3$ is the external torque applied on base, $\mathbf{T}_\mu \in \mathcal{R}$ is the terrain friction, and $N_F$ represents the contact normals.

Upon training the stability margin network, we observed an absolute prediction error mean of 0.0051 m and a standard

deviation of 0.0018 m when tested on a dataset comprising of 50k test samples.

In addition to efficiently obtaining an estimate of the stability margin, the use of a neural network, also enables computation of partial derivatives of the stability margin with respect to the robot's states thereby eliminating the need to compute the Jacobians by finite differences of the analytical solution.

This provides the possibility to formulate optimal control problems and gradient-descent based trajectory optimization problems that exploit the Jacobian with respect to the optimization variables of costs and constraints based on this MLP in order to maximize the overall stability margin or to make sure that this quantity lies within some specified boundaries. Fig. 17 represents the values of the stability margin obtained for a straight walk of Anymal B on a flat terrain for 2 s covering a distance of 0.5 m. Both the motions were obtained using the TOWR [54] library, which employs a minimal parameterization of the state of the robot based on the single rigid body dynamics model. The red values refer to a motion plan obtained with the baseline library, which attempts to obtain a solution that satisfies all the feasibility constraints such as kinematic reachability and dynamic consistency without any minimization cost. The blue curve, instead, refers to the same trajectory optimization problem where a cost has been added that maximizes the learned stability margin. As an effect of the added cost, the obtained solution presents smaller acceleration peaks and a larger stance configuration of the feet. These two elements together result in an increased value of the stability margin both during the triple and the quadruple stance phases and in an average improvement of the stability margin of about 0.025 m. Although the derivatives of the stability margin network represent an approximation of the Jacobians, we observed that this approximation still remained valid along the state regions sampled by TOWR, allowing us to obtain a feasible trajectory while maximizing the stability margin.

We observed a similar behavior while training our RL footstep planning policy. For a higher stability margin reward coefficient the footstep planning policy preferred a wider stance behavior. Upon increasing the reward coefficient for stepping close to the nominal feet position and lowering the stability margin coefficient, the footstep planning policy preferred a wider stance behavior only during unstable motion phases. This is demonstrated in Movie S5.

In our preliminary experiments, we observed that introducing the stability margin reward term in an environment setup with a simple reward function results in reduced base accelerations. As shown in Movie S5, a locomotion policy obtained using the reward function $-(0.7 - {}_B v_{WB_x})^2$, is significantly more aggressive compared to the reward function augmented using the stability margin. This reward function can further be tuned to obtain the desired locomotion behavior. In our RL environment setups, we use the stability margin in our reward function to encourage the RL agents to perform less aggressive maneuvers in order to maximize the stability metric.

We extended this work on rapid dynamic stability estimation for use with OC and RL based approaches. The detailed network training and deployment description can be found in [40].

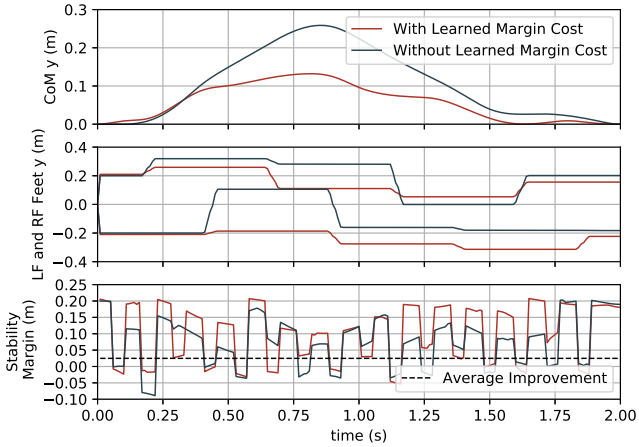
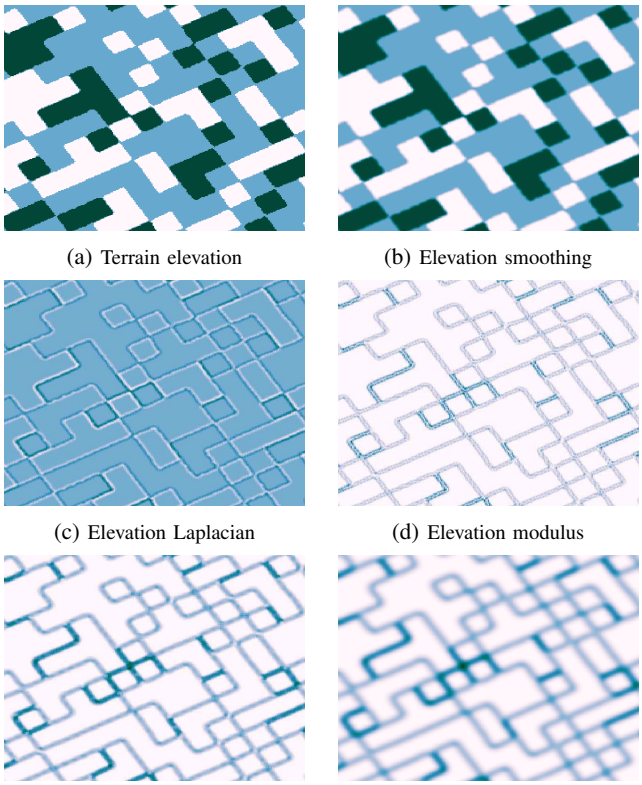



Fig. 17: Trajectories referring to two motion plans of Anymal B on flat terrain optimized using TOWR with (blue) and without (red) the cost on the learned stability margin. We can see that maximizing the learned stability margin leads to a decrease in unnecessary base motions (upper plot) and in a larger stance (mid plot). These two elements together lead to an average improvement of the stability margin of about 2.5 cm (lower plot).

### B. Cost Map

In order to drive the RL footstep planner to generate desired feet positions which avoid terrain edges, we compute a penalty based on the cost map obtained using the 1st and 2nd order derivatives of the elevation map local to the feet positions within a deviation of $\pm 0.065$ m along $\mathbf{e}_x^F$ and $\mathbf{e}_y^F$ represented by $\mathbf{M}_{F_i} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^{27\times 27}$ for foot $i$ sampled at a resolution of 0.005 m. We compute the cost map by performing operations in the following order:

1) Smoothing: $\mathbf{C_M}(r_{F_{xy},i}) \leftarrow \mathcal{C}(\mathbf{M}_{F_i}, k_s)$ where
$$k_s = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}$$
2) Approximation of second order derivative using Laplacian kernel: $\mathbf{C_M}(r_{F_{xy},i}) \leftarrow \mathcal{C}(\mathbf{C_M}(r_{F_{xy},i}), k_l)$ where
$$k_l = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
3) Modulus of the cost map to represent terrain edges without direction information: $\mathbf{C_M}(r_{F_{xy},i}) \leftarrow abs(\mathbf{C_M}(r_{F_{xy},i}))$
4) Estimation of deviation in heights along the edges using blurring kernel: $\mathbf{C_M}(r_{F_{xy},i}) \leftarrow \mathcal{C}(\mathbf{C_M}(r_{F_{xy},i}), k_b)$
$$k_b = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
5) Circular distance filtering such that terrain edge at $\mathbf{M}_{F_i}(r_{F_{xy},i})$ has the maximum weighting for computing



(a) Terrain elevation (b) Elevation smoothing

(c) Elevation Laplacian (d) Elevation modulus

(e) Elevation blurring (f) Terrain cost map

Fig. 18: Example of computation of the cost map for a bricks terrain. Green color represents upward extrusions corresponding to higher elevation while white color represents downward extrusions. For the cost map, dark blue represents positions with higher cost. Note that, the cost map shown here corresponds to the entire terrain which was obtained by performing convolution using the circular distance filter as opposed to element-wise multiplication which is done for extracting the cost at foot position in $\mathbf{M}_{F_i}$.

the cost penalty. This weight decreases with increase in the radial deviation from the center of the feet $r_{F_{xy},i}$. The circular distance filter can be represented as a matrix $\mathbf{M}_c \in \mathbb{R}^{21\times 21}$ where the center $\mathbf{M}_c(r_{F_{xy},i}) = 1$ and $\mathbf{M}_c(\hat{r}_{F_{xy},i}) = max(0.05 - ||\hat{r}_{F_{xy},i} - r_{F_{xy},i}||, 0)/0.05$. The cost map is then modified by performing an element-wise product: $\mathbf{C}_\mathbf{M}(r_{F_{xy},i}) \leftarrow M_c \odot \mathbf{C}_\mathbf{M}(r_{F_{xy},i})$

Note that, we start with $\mathbf{M}_{F_i} : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}^{27\times 27}$ which corresponds to an area of $(2\times 0.065)\times(2\times 0.065)\,\mathrm{m}^2$, and following the above procedure, we obtain the desired $21\times 21$-dimensional cost map which corresponds to an area of $(2\times 0.05)\times(2\times 0.05)\,\mathrm{m}^2$. This reduction is due to the convolutions performed without utilizing additional padding.

Figure 18 shows an example of the cost map on a terrains with bricks and holes. In practice, the whole cost map does not need to be explicitly computed since only the cost at each foot position is needed so the transformation are only applied on small patches around the selected footholds.

### C. Elevation Map Encoding

Our encoder network architecture design is based on the following reasons: (1) the encoder should capture as much information as possible, (2) the encoder forward pass time on the real robots should not exceed the control step time of 2.5 ms after the raw elevation map is computed in a parallel thread to avoid change in robot state resulting in feet position tracking drifts. For this, we introduced the unstructured terrain in the dataset for training the autoencoder. We observed that reducing the dimensionality of the latent representation below 96 resulted in poor reconstruction of the unstructured terrain, i.e. the encoder failed to detect the unstructured terrain, and the autoencoder output an almost smooth reconstruction. A forward pass through the encoder network requires approximately 1.7 ms on the real robots giving us sufficient time to perform state updates and forward passes through the RL planning and control policy networks.

While training the encoder using the autoencoding strategy, we minimize the loss

$$\mathcal{L}_{\mathbf{M}_H} = (\mathbf{M}_H^n - \hat{\mathbf{M}}_H^n)^2\,. \tag{26}$$

It is a common practice to include a regularization term in this reconstruction loss to discourage overfitting, however, reconstruction from noisy encoder input implies that the autoencoder is restricted from memorizing the training dataset. Instead, the encoder learns to map the input data to a lower-dimensional manifold which represents the most relevant features of the terrain while getting rid of the additional noise. Hence, it is not necessary for us to introduce additional regularization during training.

### D. Control Switching

RLOC executes the footstep planner, motion controller and domain adaptive tracker upon initialization. When any of the robot state parameters exceeds a certain threshold described by the activation limit in Table X, the recovery controller is activated while the footstep planner, motion controller and domain adaptive tracker are deactivated. The recovery controller stabilizes the robot and the active control state switches back to motion planning and whole-body control. This switch is performed when each of the robot state parameters lie within the deactivation limits presented in Table X.

### E. Terrain Generation

For terrain generation, we consider several objects such as a staircase, planks, bricks, unstructured sub-terrain and wave sub-terrain. Each terrain contains at least 1 and at most 5 objects. We position and rotate these objects within the terrain area to limit height deviations around close regions and increase the occupancy over the entire terrain, and represent their corresponding heights as elevation in the range of $[0, 65535]$ where 0 corresponds to no elevation and 65535 corresponds to an elevation of 2 m. We automate this generation process using a Python tool which stores a wide range of terrains, 10k in our case, in the form of single channel 16-bit PNG files. These PNGs are then imported in RaiSim as heightmaps. Figure 19

TABLE X: Robot state criteria for activating and deactivating the recovery controller. Here $|\cdot|$ refers to the $l_1$ norm of a vector, abs() is the element-wise modulus operator, max() refers to the maximum element in a vector, $\mathrm{q}_j^n$ is the vector of nominal joint positions and $(\mathrm{q}_j - \mathrm{q}_j^n)_{HAA}$ refers to the vector of elements relating to the four HAA (hip abduction-adduction) joints. HFE refers to hip flexion-extension and KFE refers to knee flexion-extension.

| State | Activation Limit | Deactivation Limit |
|---|---|---|
| $\lvert \mathrm{v}_B \rvert$ | 20.0 | 0.5 |
| $\lvert \boldsymbol{\omega}_B \rvert$ | 5.0 | 0.5 |
| $\max(\mathrm{abs}((\mathrm{q}_j - \mathrm{q}_j^n)_{HAA}))$ | $\pi/4$ | $\pi/12$ |
| $\max(\mathrm{abs}((\mathrm{q}_j - \mathrm{q}_j^n)_{HFE}))$ | $\pi/4$ | $\pi/12$ |
| $\max(\mathrm{abs}((\mathrm{q}_j - \mathrm{q}_j^n)_{KFE}))$ | $\pi/3$ | $\pi/12$ |
| $\lvert \cos^{-1}(\mathbf{e}_z^B \cdot \mathbf{e}_z^W) \rvert$ | $\pi/3$ | $\pi/4$ |
| $\lvert \cos^{-1}(\mathbf{e}_z^B \cdot \mathbf{e}_z^C) \rvert$ | $\pi/4$ | $\pi/6$ |

represents the generated terrains for the unstructured ground, stairs, wave and brick objects along with their corresponding visualization in RaiSim.

### *F. Actuation Dynamics*

The actuator network utilized for training footstep planner and domain adaptive tracker models an adaptive impedance controller which can be represented as

$$\tau_j = \mathbf{j}_R(\mathrm{q}_j^*, \mathrm{q}_j, \dot{\mathrm{q}}_j^*, \dot{\mathrm{q}}_j, \tau_{j_{FF}}, K_p, K_d) \ . \tag{27}$$

The input to the actuator network is thus represented as $j_s \in \mathbb{R}^{21}$ such that

$$j_s = \langle s_{j_{pe}}, s_{j_{ve}}, s_{j_{te}}, s_{j_{vd}}, s_{j_{td}}, s_{j_{kp}}, s_{j_{kd}} \rangle \ . \tag{28}$$

The definitions of the input terms are provided in Table XI. This actuator network contains 2 hidden layers with 48 nodes in each of the layers and a 1-dimensional output layer representing the estimate of the joint torque. We perform a forward pass through the network at each time step for each of the 12 joints of the quadruped.

TABLE XI: Actuator input term definitions. Here, $i$ refers to the $i$-th joint. Subscript $t$ refers to the measurement at the current control step and the duration between $t-1$ and $t$ corresponds to 2.5 ms.

| Actuator Network Input Terms |
|---|
| $s_{j_{pe}} = [(\mathrm{q}^*_{j,i_t} - \mathrm{q}_{j,i_t})^T \quad (\mathrm{q}^*_{j,i_{t-4}} - \mathrm{q}_{j,i_{t-4}})^T \quad (\mathrm{q}^*_{j,i_{t-8}} - \mathrm{q}_{j,i_{t-8}})^T]^T$ |
| $s_{j_{ve}} = [(\dot{\mathrm{q}}^*_{j,it} - \dot{\mathrm{q}}_{j,it})^T \quad (\dot{\mathrm{q}}^*_{j,i_{t-4}} - \dot{\mathrm{q}}_{j,i_{t-4}})^T \quad (\dot{\mathrm{q}}^*_{j,i_{t-8}} - \dot{\mathrm{q}}_{j,i_{t-8}})^T]^T$ |
| $s_{j_{te}} = [(\tau_{j,i_t} - \tau_{j,i_{FF}t})^T \quad (\tau_{j,i_{t-4}} - \tau_{j,i_{FF}t-4})^T \quad (\tau_{j,i_{t-8}} - \tau_{j,i_{FF}t-8})^T]^T$ |
| $s_{j_{vd}} = [(\dot{\mathrm{q}}^*_{j,i_t})^T \quad (\dot{\mathrm{q}}^*_{j,i_{t-4}})^T \quad (\dot{\mathrm{q}}^*_{j,i_{t-8}})^T]^T$ |
| $s_{j_{td}} = [\tau^T_{j,i_{FF}t} \quad \tau^T_{j,i_{FF}t-4} \quad \tau^T_{j,i_{FF}t-8}]^T$ |
| $s_{j_{kp}} = [K_{p_t}{}^T \quad K_{p_{t-4}}{}^T \quad K_{p_{t-8}}{}^T]^T$ |
| $s_{j_{kd}} = [K_{d_t}{}^T \quad K_{d_{t-4}}{}^T \quad K_{d_{t-8}}{}^T]^T$ |

For training the adaptive impedance actuator network, we collected data from the real system using the dynamic gaits motion controller to track randomly sampled base reference velocities. We recorded the desired relevant input quantities

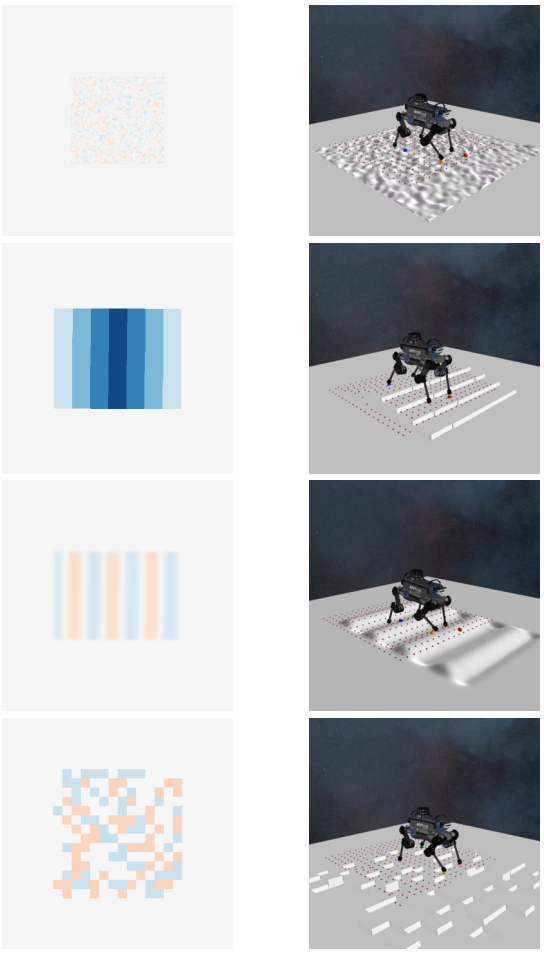


Fig. 19: **(top-bottom)** represent the terrains comprising of unstructured ground, stairs, wave and brick objects respectively. The left images represent the height maps imported in the RaiSim simulator as shown on the right.

at 400 Hz for 12 actuators thus generating a large dataset of 8.64M samples in approximately 30 minutes. However, since the impedance gains depend on the contact state of the robot, during data collection, we obtained sufficient samples for the swing and stance phases. However, during operation, we obtained significantly less samples for the contact-uncertainty state. As a result, using the actuator network trained with the naive supervised learning approach failed to properly track actuation commands for contact-uncertainty states in the simulator causing the motion controller to generate unstable motion tracking commands eventually resulting in failure. To circumvent this problem. we utilized SMOGN [55] to perform regression over imbalanced data by resampling the dataset generated from the real robot. SMOGN utilizes a random undersampling approach for the majority samples and a synthetic sample generation method for the minority samples. Table XII presents the mean absolute prediction error comparing the performance of networks trained using a naive supervised learning approach with resampled learning, tested on 50k samples for each of the different sets of PD gains.

TABLE XII: Effective joint torque prediction error comparing the actuator network performance for training with and without re-sampling of the training dataset using SMOGN. Here $\mu_e$ is the mean prediction error and $\sigma_e$ is the standard deviation of the prediction error observed over 50k test samples.

| | $K_p$=8, $K_d$=0.1 | $K_p$=35, $K_d$=0.4 | $K_p$=160, $K_d$=0.3 |
|---|---|---|---|
| **Baseline** | $\mu_e = 0.82$, $\sigma_e = 0.34$ | $\mu_e = 0.77$, $\sigma_e = 0.31$ | $\mu_e = 1.73$, $\sigma_e = 0.85$ |
| **SMOGN** | $\mu_e = 0.83$, $\sigma_e = 0.21$ | $\mu_e = 0.71$, $\sigma_e = 0.19$ | $\mu_e = 0.96$, $\sigma_e = 0.39$ |

### *G. Training*

Our motivation for selecting an RL training strategy is based on three main factors: (1) convergence to a desired behavior, (2) sample efficiency, and (3) training time.

We use SAC for training our footstep planning policy mainly for its improved sample efficiency over the widely used TRPO and PPO approaches. In our preliminary experiments for training the footstep planner, we observed that SAC required approximately a fourth of policy iterations for convergence compared to PPO for similar training parameters. For this reason, we decided to use SAC for training our footstep planner.

In the case of domain adaptive tracker, we observed that the stochasticity associated with SAC and PPO necessitated precise hyperparamter tuning, without which the policy failed to exhibit a stable locomotion behavior. Even then, we observed little improvement in domain adaptive tracking. This was largely due to the domain adaptive tracker generating large torques which caused the whole-body controller to aggressively correct for inaccurate tracking resulting in episode termination. Instead, we used TD3 which allowed us to perform exploration using an uncorrelated Gaussian noise. We thus employed a curriculum based exploration strategy where we initialized the policy to output values close to 0s, while also setting a small value for the Gaussian noise standard deviation. As training progressed, we increased the standard deviation. This allowed for stable learning and thus better convergence.

For the recovery controller, we use PPO for 2 main reasons: (1) Prior work on end-to-end state-to-joint RL control has utilized PPO and provided reproducible steps [21], and (2) we observed that, compared to off-policy approaches such as SAC and TD3, PPO required significantly less time to perform policy updates after each episode rollout. Since the footstep planner and domain adaptive tracker employed the motion controller during training, much of the training overhead occurred during simulation rollouts as opposed to policy updates. The recovery controller rollouts were significantly fast since no additional computation, such as motion optimization, was required. PPO thus enabled us to perform faster training even with reduced sample efficiency. In our preliminary experiments, we observed that for each policy iteration, PPO required approximately 3.2 s whereas SAC required approximately 13 s for similar training parameters. Although it is possible to reduce the policy update duration for SAC with additional parameter tuning, for our test case, PPO still offered policy convergence in less training time than SAC. We therefore decided to use PPO for training the recovery controller.

We trained each of our RL policies on a desktop computer with an Intel i7-8700 processor clocked at 4.0 GHz and an NVidia RTX 2080Ti graphics card. The footstep planner Session 1 training time and hyperparameters are shown in Table XIII. For training Session 2, we use the SAC hyperparameters shown in Table XIII for additional 100 policy update iterations requiring roughly 4 hours of training. For training the domain adaptive tracking policy, we use the parameters provided in Table XIV, and we use the parameters provided in Table XV for training the recovery controller.

TABLE XIII: Footstep planner Session 1 training time and hyperparameters.

| Parameter | Value |
|---|---|
| Discount factor, $\gamma$ | 0.994 |
| Learning rate | 1e-4 |
| Buffer size | 1e5 |
| Batch size | 8192 |
| Prioritized replay buffer | False |
| Learning starts | 24576 |
| Parallel environments, $n_{env}$ | 24 |
| Soft update coefficient, $\tau$ | 1e-3 |
| Training frequency | 1 |
| Twin Q functions | True |
| Target entropy | Auto |
| IGPO $\lambda_d$ | 0.996 |
| IGPO $n_{eps}$ | 100 |
| Steps per iteration, $l_{eps}$ | $1024 \times n_{env}$ |
| SAC time per iteration | $\sim$120 s |
| Guided learning time per iteration | $\sim$105 s |
| Total training time | $\sim$28 h |

TABLE XIV: Domain adaptive tracker training time and hyperparameters.

| Parameter | Value |
|---|---|
| Discount factor, $\gamma$ | 0.998 |
| Learning rate | 5e-4 |
| Buffer size | 1e6 |
| Batch size | 19200 |
| Learning starts | 76800 |
| Parallel environments, $n_{env}$ | 24 |
| Soft update coefficient, $\tau$ | 5e-3 |
| Prioritized replay | True |
| Prioritized replay $\alpha$ | 0.6 |
| Prioritized replay $\beta$ | 0.6 |
| Twin Q functions | True |
| Steps per iteration, $l_{eps}$ | $3200 \times n_{env}$ |
| TD3 time per iteration | $\sim$4 s |
| Total training time | $\sim$22 h |


## ACKNOWLEDGMENT

S.G. trained the actuator network, stability margin network and the denoising convolutional autoencoder. S.G. setup the

TABLE XV: Recovery controller training time and hyperparameters.

| Parameter | Value |
|---|---|
| Discount factor, $\gamma$ | 0.996 |
| Learning rate | 5e-4 |
| Batch size | 115200 |
| Mini-batch size | 14400 |
| Parallel environments, $n_{env}$ | 48 |
| Entropy coefficient | 5e-3 |
| Value coefficient | 0.25 |
| GAE | True |
| GAE $\lambda$ | 0.98 |
| Steps per iteration, $l_{eps}$ | $2400 \times n_{env}$ |
| PPO time per iteration | $\sim$3 s |
| Total training time | $\sim$6 h |

RL environments and performed training of footstep planning, domain adaptive tracking and recovery control policies. M.G. extended the model-based motion controller for locomotion over uneven terrain and helped set up simulation environments. R.O. extended the stability margin analysis to dynamic motions with 2 feet contacts. S.G. and M.G. wrote the control software for the real robot. S.G., M.G and R.O. performed hardware experiments. I.H. and M.F. provided funding and supervised the project. S.G. took the lead in writing the manuscript. M.G., R.O., M.F. and I.H. provided critical feedback and helped shape the manuscript.

We would like to thank David Rytz, Matias Mattamala and Oliver Bartlett of the Oxford Robotics Institute for helping us with our experiments.

## REFERENCES

[1] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 38–44.

[2] E. Ackerman, "Anybotics introduces sleek new anymal c quadruped," *IEEE Spectrum. https://spectrum.ieee.org/automaton/robotics/industrial-robots/anybotics-introduces-sleek-new-anymal-c-quadruped*, 2019.

[3] ——, "North sea deployment shows how quadruped robots can be commercially useful," *IEEE Spectrum. https://spectrum.ieee.org/automaton/robotics/industrial-robots/north-sea-deployment-shows-how-quadruped-robots-can-be-commercially-useful*, November 2018.

[4] H. Kolvenbach, D. Wisth, R. Buchanan, G. Valsecchi, R. Grandia, M. Fallon, and M. Hutter, "Towards autonomous inspection of concrete deterioration in sewers with legged robots," *Journal of Field Robotics*, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21964

[5] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. Graph.*, vol. 31, no. 4, Jul. 2012. [Online]. Available: https://doi.org/10.1145/2185520.2185539

[6] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2536–2542.

[7] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4730–4737.

[8] R. Full and D. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3325–3332, 1999. [Online]. Available: http://jeb.biologists.org/content/202/23/3325

[9] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, Oct 2001, pp. 239–246 vol.1.

[10] I. Poulakakis, E. Papadopoulos, and M. Buehler, "On the stability of the passive dynamics of quadrupedal running with a bounding gait," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 669–687, 2006. [Online]. Available: https://doi.org/10.1177/0278364906066768

[11] I. Poulakakis and J. W. Grizzle, "The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper," *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 1779–1793, 2009.

[12] M. Vukobratović and D. Juričić, "Contribution to the synthesis of biped gait," *Proc. IFAC Symp. Technical and Biological Problem on Control*, Erevan, USSR, 1968.

[13] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," pp. 200 – 207, 01 2007.

[14] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Auton. Robots*, vol. 35, no. 2-3, pp. 161–176, Oct. 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-013-9341-4

[15] H. Dai and R. Tedrake, "Planning robust walking motion on uneven terrain via convex optimization," in *Humanoids*, 2016.

[16] R. Orsolino, M. Focchi, C. Mastalli, H. Dai, D. G. Caldwell, and C. Semini, "Application of wrench based feasibility analysis to the online trajectory optimization of legged robots," *IEEE Robotics and Automation Letters (RA-L)*, 2018.

[17] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, 2018.

[18] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, "Trajectory generation for multi-contact momentum control," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 874–880.

[19] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain–online foothold optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5370–5376, 2020.

[20] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2464–2470.

[21] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau5872

[22] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

[23] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3642–3649, 2020.

[24] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

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

[26] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

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

[28] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Conference on Robot Learning*, 2020, pp. 1–10.

[29] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.

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

[31] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.
[32] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.
[33] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, 2018.
[34] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.
[35] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.
[36] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.
[37] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3359–3365.
[38] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, "Dynamic locomotion on slippery ground," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4170–4176, 2019.
[39] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, and C. Semini, "Feasible region: an actuation-aware extension of the support region," *IEEE Transactions on Robotics (TRO)*, 2020.
[40] R. Orsolino, S. Gangapurwala, O. Melon, M. Geisert, I. Havoutis, and M. Fallon, "Rapid stability margin estimation for contact-rich locomotion," 2021.
[41] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.
[42] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2018.
[43] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.
[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[45] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.
[46] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.
[47] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, 2018, pp. 1582–1591.
[48] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.
[49] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Real-time trajectory adaptation for quadrupedal locomotion using deep reinforcement learning," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers, 2021.
[50] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
[51] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.
[52] R. Smith *et al.*, "Open dynamics engine," 2005.
[53] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, 2008.
[54] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems through Phase-based End-Effector Parameterization," *IEEE Robotics and Automation Letters (RA-L)*, pp. 1–1, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8283570/
[55] P. Branco, L. Torgo, and R. P. Ribeiro, "Smogn: a pre-processing approach for imbalanced regression," in *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, 2017, pp. 36–50.



**Siddhant Gangapurwala** completed his Bachelor of Engineering (B.E.) in Electronics from the University of Mumbai in 2016. In 2017, he joined the AIMS program as a doctoral candidate at the University of Oxford. The following year, Siddhant joined the Dynamic Robot Systems (DRS) group of the Oxford Robotics Insitute (ORI) to pursue his DPhil degree with focus on machine learning and optimal control based approaches for robotic locomotion over uneven terrain where he continues to work as a doctoral researcher.



**Mathieu Geisert** received his M. Eng. in Aerospace from Institut Supérieur de l'Aéronautique et de l'Espace SUPAERO (Toulouse, France) in 2013. He then joined the humanoid robotics team, GEPETTO, at LAAS-CNRS (Toulouse, France) where he worked for 5 years in different positions: as an intern, engineer, Ph.D. student and finally as a post doctoral researcher. Since 2018, He is a post doctoral researcher in the DRS group of the ORI. His research focuses on Planning, Optimal Control and Machine Learning for legged robots.



**Romeo Orsolino** completed his B.Sc. in mechanical engineering from the University of Genova in 2013 and his M.Sc. in robotics engineering at the École Centrale de Nantes in 2015. In the same year, he joined DLS at IIT for a Ph.D. focusing on motion planning for legged locomotion in rough terrains. In October 2019, he joined the DRS group as a post-doctoral researcher where he pursued his research on multi-contact motion planning, optimal control, dynamics and perception. In December 2020, Romeo joined Arrival Ltd. where continues to work on perceptive motion planning and control of dynamical systems.



**Maurice Fallon** studied Electronic Engineering at University College Dublin. His Ph.D. research in the field of acoustic source tracking was carried out in the Engineering Department of the University of Cambridge. After his Ph.D. he moved to MIT as a post doctoral researcher and later research scientist in the Marine Robotics Group (2008-2012). From 2012-2015 he was the perception lead of MIT's team in the DARPA Robotics Challenge. From 2015, he was a Lecturer at University of Edinburgh where he led research in collaboration with NASA's humanoid robotics program. He moved to Oxford in April 2017 and took up the Royal Society University Research Fellowship in October 2017.



**Ioannis Havoutis** is a Departmental Lecturer at the Oxford Robotics Institute. His research combines dynamic whole-body motion planning and control with machine learning, focusing on robots with arms and legs. He received his Ph.D. (2011) and M.Sc. with distinction (2007) from the University of Edinburgh, where he worked on machine learning for motion planning and control of articulated robots. He then joined the Dynamic Legged Systems Lab at the Advanced Robotics Department of IIT, where he led the Locomotion Group within the HyQ team. He was a postdoctoral researcher at the Robot Learning & Interaction Group of the Idiap Research Institute, where he worked on learning complex skills from demonstration.

# 7

# Paper V
# Learning Dynamics Conditioned Platform Adaptive Quadrupedal Locomotion Policy

# Manuscript Preface

This work proposes a training approach to obtain a dynamics conditioned locomotion policy that can transfer to wide domains of quadrupedal platforms without requiring retraining. For this, an encoding network is employed to generate a latent representation of system dynamics. The latent dynamics representation is used to condition a locomotion policy which maps robot state information to desired joint positions. Figure 15 shows the latent dynamics representation obtained for different quadrupeds for 1 s of locomotion. While these unrolled representations look similar, the magnitude of the elements differ for each of the quadrupeds. More importantly, Section IV-E details upon the pattern observed in each of the unrolled latent dynamics representations. This corresponds to temporal locality associated with the latent representations and is based on the assumption that for smaller temporal regions where the contact forces remain constant, the quadrupedal base and joint accelerations are largely described by the actuation torques (control signal). In this regard, the latent dynamics representation changes throughout locomotion as different feet contact configurations are observed.

Since the dynamics encoder is parameterized as a recurrent network, it becomes counter-intuitive that the latent dynamics representation is temporally local. The recurrent structure, however, is motivated by the architecture utilized in DeepIM [Li et al., 2018] wherein a recurrent network is employed to iteratively improve the pose estimation of a reference object. Along this direction, the encoding network utilizes the hidden state to iteratively improve its dynamics estimation. In this regard, the encoding network is able to generate an accurate estimation of the system dynamics in less than 5 control steps which corresponds to a duration of 10 ms.

The proposed approach draws parallels between the work of *Yu et al.* on learning universal policy with online system identification [Yu et al., 2017]. The work of *Yu et al.* presents a control architecture which utilizes a history of state and actions to explicitly estimate the dynamics model parameters of the control system. The control policy is then conditioned on these parameters. Another work of *Yu et al.* leverages domain randomization to obtain a family of policies that exhibit different behaviors [Yu et al., 2018]. The best policy is then chosen in a target environment depending upon the task performance. This, therefore, eliminates the need to perform online dynamics identification. Such an approach, however, delivers limited performance depending upon the number of policies trained and evaluated online. In contrast, the work presented in the attached manuscript employs a dynamics encoding network which generates a latent dynamics representation. To train this encoding network, a predictor network is utilized to generate estimates of system accelerations given the latent dynamics representation when conditioned on a control signal.

While this work utilizes significant dynamics randomization to obtain an adaptive control policy, it is important to note that for a training environment that does not encompass the target domain, the performance of the adaptive policy deteriorates upon transfers to unobserved target domains. In this regard, Section IV-A describes inability of the control policy to adapt to a domain containing delayed actuation. Therefore, the training environment includes randomization of actuation delays whilst also utilizing actuator networks in some environments to model complex non-linear actuation dynamics [Hwangbo et al., 2019]. During evaluation, the policy is demonstrated to successfully transfer to physical robots with different actuation characteristics such as ANYmal B (which contains series-elastic actuators) and A1 (containing quasi-direct drive actuators) [Kim et al., 2021].

# Learning Dynamics Conditioned Platform Adaptive Quadrupedal Locomotion Policy

Siddhant Gangapurwala[1], Suyoung Choi[2], Mathieu Geisert[1], Jemin Hwangbo[2] and Ioannis Havoutis[1]

[1]Dynamic Robot Systems, Oxford Robotics Institute, University of Oxford
[2]RaiLab, Department of Mechanical Engineering, KAIST

***Abstract*— We present a locomotion control framework that generalizes to quadrupedal robots of varying kinematic and dynamic properties. For this, we train a reinforcement learning (RL) control policy, over diverse domains of procedurally generated quadrupeds, to map proprioceptive robot state information and reference base velocity commands into desired actuation targets. To achieve platform adaptive characteristics, we introduce an embedded representation of system dynamics in the RL state space. We generate this *latent dynamics* embedding, at each control step, through a recurrent encoding network utilizing a series of observed state-transitions. We do not explicitly model the system description parameters, instead infer it using the dynamics encoder. This allows us to transfer the control policy to different platforms without performing exhaustive system identification. We show that a control policy conditioned on system dynamics is able to generate locomotion behaviors for a wide range of quadrupedal platforms without requiring retraining. We demonstrate the effectiveness of our approach on a wide range of simulated quadrupeds, and further transfer the learned RL policy to different physical robots.**

## I. Introduction

Quadrupedal locomotion research has remarkably progressed in recent years observing contributions in numerical optimization [1], [2] and RL [3], [4] based control approaches that demonstrate complex dynamic locomotion behavior even on challenging [5] and uneven terrains [6], [7]. The development process of these control solutions, however, is often laborious, requiring precise tuning of control parameters and cost functions. Moreover, these controllers are typically tailored to exhibit optimal control behavior for a target system. When transferred to domains not considered during development, the performance of these control architectures deteriorates. This demands additional redevelopment and adaptation of existing control strategies for deployment on different robots. As a solution, this work investigates platform adaptive locomotion to obtain a control policy that is capable of zero-shot generalization to a diverse set of quadrupedal systems.

While majority of analytical model-based control approaches necessitate exhaustive hand-tuning of control parameters, including the dynamics model employed for generation of control action, works which utilize an analytical prior to learn a dynamics model have also been proposed [8], [9], [10]. *Grandia et al.* present an approach to learn the error terms for rigid body dynamics and actuation dynamics of a quadruped using Gaussian process regression [11]. In this direction, the research works [12], [13], [14], [15] utilize a *learned* dynamics model to obtain a control policy. Based on these works, *Deisenroth et al.* introduced an efficient model-based policy search approach, PILCO [16], for control of complex systems. It learns a Gaussian process (GP) model, and uses this model to sample trajectories which are then used to train a controller. Alternatively, recent works which utilize Bayesian neural networks to learn system dynamics [17], [18] have also been presented. The use of recurrent neural networks (RNNs) has also been proposed to learn temporal dynamic characteristics [19]. *Ha et al.* further introduced an approach of employing a generative RNN to learn *world models* that are used to obtain control policies through evolutionary strategies [20]. *Sekar et al.* further proposed Plan2Explore [21] to perform zero or few-shot adaptations of tasks during evaluation.

In this work, We propose an approach to condition a locomotion policy on system dynamics to allow for emergence of adaptive behavior which is robust to variations in kinematic and dynamic properties. An overview of our

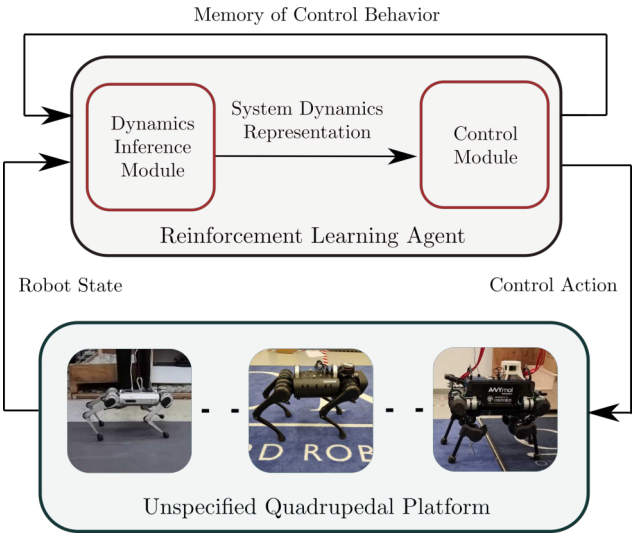



Fig. 1: Overview of our approach. The reinforcement learning agent comprises an inference module which generates an estimated dynamics representation using a series of state transitions. The control module then generates a control action for a robot state conditioned on the system dynamics to perform quadrupedal locomotion.

control architecture is shown in Fig. 1. We introduce an inference module to generate an embedded representation of systems dynamics, and refer to the embedding as *latent dynamics*. The control module (CM) utilizes the obtained latent dynamics to generate a control action for a non-identified quadrupedal robot. The state-transition observed is then used by the dynamics inference module (DIM) to iteratively improve the latent dynamics estimation, thereby improving control performance. We train, both, DIM and CM simultaneously, and then evaluate the transferability of our method on different quadrupeds.

Our contributions with this work are: (1) presenting a training method that utilizes an auxiliary loss to learn a dynamics inference module concurrently with learning a platform-adaptive RL locomotion policy, and (2) demonstrating successful transfer of our method to different quadrupeds in simulation, and on the real A1 and ANYmal B robots.

## II. Preliminaries

### A. System Model

We model a quadrupedal system as a floating base $B$ with four actuated limbs attached to it. The robot state is represented w.r.t. a global inertial reference frame $W$ such that the $z$-axis of $W$, $\mathbf{e}_z^W$, aligns with the gravity axis. The base position is then expressed as $r_B \in \mathbb{R}^3$, and the orientation, $\mathrm{q}_B \in SO(3)$, is represented by unit quaternions with a corresponding rotation matrix $\mathbf{R}_B \in SO(3)$. The angular positions of the rotational joints in each of the limbs are described by the vector $\mathrm{q}_j \in \mathbb{R}^{n_j}$. For the quadrupeds considered in this work, $n_j = 12$. The linear and angular velocities of the base w.r.t. the global frame are written as $\mathrm{v}_B \in \mathbb{R}^3$ and $\omega_B \in \mathbb{R}^3$ respectively. The generalized coordinates and velocities are thus stacked as vectors q and u where

$$\mathrm{q} = \begin{bmatrix} r_B \\ \mathrm{q}_B \\ \mathrm{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \quad \mathrm{u} = \begin{bmatrix} \mathrm{v}_B \\ \omega_B \\ \dot{\mathrm{q}}_j \end{bmatrix} \in \mathbb{R}^{6+n_j}. \tag{1}$$

The feet positions are represented as $r_F \in \mathbb{R}^{3\times n_f}$, where $n_f = 4$ is the number of feet.

We assume that quadrupedal systems are actuated using the joint control torques $\tau_j \in \mathbb{R}^{n_j}$ computed using an impedance control model expressed as

$$\tau_j = K_p(\mathrm{q}_j^* - \mathrm{q}_j) + K_d(\dot{\mathrm{q}}_j^* - \dot{\mathrm{q}}_j) + \tau_{j_{FF}}, \tag{2}$$

where $K_p$ and $K_d$ refer to the position and velocity tracking gains respectively, $\mathrm{q}_j^*$ is the vector representing desired joint positions, $\dot{\mathrm{q}}_j^*$, the desired joint velocities, and $\tau_{j_{FF}}$ refers to the feed-forward joint torques. In this work, we set $\dot{\mathrm{q}}_j^* = 0$ and $\tau_{j_{FF}} = 0$. Eq. 2 can thus be simplified to

$$\tau_j = K_p(\mathrm{q}_j^* - \mathrm{q}_j) - K_d\dot{\mathrm{q}}_j. \tag{3}$$

While eq. 3 corresponds to an ideal dynamics model, the actuation response observed on physical systems often demonstrates non-linear characteristics and command tracking latency [22], [3]. We therefore rewrite the impedance control model as

$$\tau_j = \mathbf{j}(\mathrm{q}_j^*, \mathrm{q}_j, \dot{\mathrm{q}}_j, K_p, K_d) \tag{4}$$

where $\mathbf{j}$ represents the complex actuation dynamics model. Additionally, since actuators cannot provide infinite torques, we clip $\tau_j$ between $[-\tau_j^{max}, \tau_j^{max}]$.

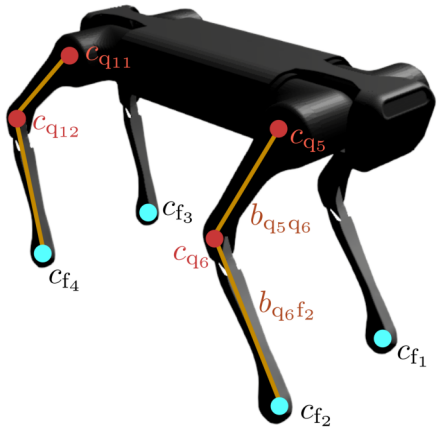



Fig. 2: Illustration of the kinematic robot description utilized in this work. The red circles represent rotational joints, whereas the blue circles represent fixed joints.

### B. Robot Description

We represent the kinematic model of a general quadruped as a combination of individual bodies, $b_{AB}$, linked either via fixed or rotational joints, $A$ and $B$, where $A$ is the parent joint and $B$ is the child joint. An exception to this is the *base* body, represented as $b_{base}$, which is considered to be a floating joint. Each body, $b_{AB}$ has an associated mass, $m_{AB} \in \mathbb{R}$, and a center of mass (CoM), $o_{AB} \in \mathbb{R}^3$. The position of each child joint, $B$, relative to the parent joint $A$, is represented as $c_B \in \mathbb{R}^3$. As an example, in Fig. 2, the body $b_{\mathrm{q}5\mathrm{q}6}$ is linked to joints $\mathrm{q}_5$ and $\mathrm{q}_6$, and has a mass $m_{\mathrm{q}5\mathrm{q}6}$ and CoM $o_{\mathrm{q}5\mathrm{q}6}$. Additionally, each of the feet has a friction coefficient associated with it. These coefficients are stacked in a vector $\mu_f \in \mathbb{R}^4$. We also attach a *frame* to each of the feet. The corresponding linear and angular feet frame velocities, expressed in the $W$ frame, are stacked into vectors $v_f \in \mathbb{R}^{12}$ and $\omega_f \in \mathbb{R}^{12}$.

### C. Reinforcement Learning

We formulate the problem of sequential decision making in the framework of a discrete time Markov decision process (MDP) [23]. An MDP is defined by the tuple - $\langle S, A, R, P, \mu \rangle$, where $S$ represents a set of states, $A$ a set of actions, $R : S \times A \times S \to \mathbb{R}$ the reward function, $P : S \times A \times S \to [0,1]$ the state transition probability, and $\mu$ the initial state distribution. An RL agent described in this MDP framework interacts with an environment based on a policy $\pi : S \to \mathcal{P}(A)$ which is defined as a function mapping states to probability distributions over actions such that $\pi(a|s)$ denotes the probability of selecting

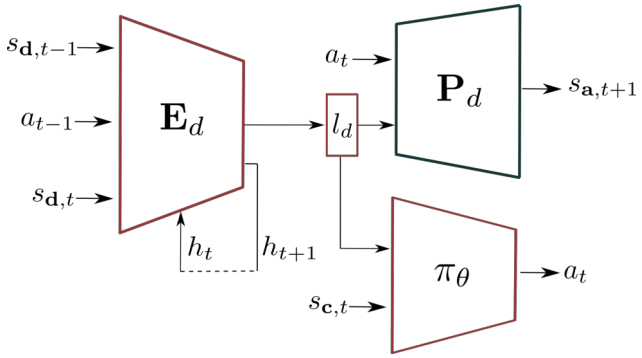



Fig. 3: Block diagram of our training setup. The red blocks are utilized for both training and evaluation, whereas, the blue blocks are only used for training. The state term definitions are provided in Table I.

action $a$ in state $s$. The objective of the RL agent is to then obtain a policy $\pi_\theta$ for parameters $\theta$ in order to maximize the expected cumulative discounted return,

$$J(\pi) \doteq \mathop{\mathrm{E}}_{\mathcal{T}\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1})\right], \tag{5}$$

where $\gamma \in [0,1)$ is the discount factor and $\mathcal{T}$ denotes a trajectory dependent on $\pi_\theta$. In this work, we parameterize the policy, $\pi_\theta$, as a neural network. Note that, while the RL policy generates a probability distribution over actions, for the purpose of evaluation, we consider that the RL policy is deterministic. We therefore refer to the policy $\pi_\theta$ as a function mapping state to desired control action.

## III. Methodology

We propose a recurrent encoding network, $\mathbf{E}_d$, to estimate and generate the latent dynamics vector. This constitutes DIM. We then introduce the generated vector into the state space of the locomotion policy, $\pi_\theta$, representing CM. $\pi_\theta$ outputs an action vector comprising the desired joint positions which are then tracked by a low-level actuation controller on the robots. Both DIM and CM are executed successively at 100 Hz. We train our RL agent, comprising DIM and CM, with 25k different quadrupeds of varying physical and actuation characteristics. We additionally introduce a transition prediction network, $\mathbf{P}_d$, to estimate the change in robot state, conditioned on a control action, for given latent dynamics. The gradient flow then occurs backwards from both $\mathbf{P}_d$ and $\pi_\theta$ into $\mathbf{E}_d$ during training. The training setup is illustrated in Fig. 3.

This section details upon the robot generation procedure, the dynamics encoding network architecture, the control policy architecture, and the training strategy.

### A. Robot Generation

We utilize 5 simplified reference models based on existing quadrupeds to generate a massive set of robots by randomizing their kinematic and dynamic properties. The *reference quadrupeds* include A1 [24], Aliengo [25], Laikago [26], ANYmal B [27] and ANYmal C [28], referred to, in this manuscript, as M1, M2, M3, M4 and M5 respectively. These are shown in Fig. 4.

TABLE I: State term definitions. Here, $t$ refers to the current measurement and the duration between $t-1$ and $t$ corresponds to 10 ms

| State definitions |
|---|
| $s_R = \mathbf{e}_z^B$ |
| $s_v = [(\mathbf{R}_B^T \mathrm{v}_B)^T \quad (\mathbf{R}_B^T \boldsymbol{\omega}_B)^T]^T$ |
| $s_j = [\mathrm{q}_j^T \quad \dot{\mathrm{q}}_j^T]^T$ |
| $s_c = \mathbf{c}^*$ |
| $s_f = [(\mathbf{R}_B^T \mathrm{v}_f)^T \quad (\mathbf{R}_B^T \boldsymbol{\omega}_f)^T]^T$ |
| $s_h = [\mathrm{q}_{j,t-1}^T \quad \mathrm{q}_{j,t-2}^T \quad \mathrm{q}_{j,t-1}^{*T} \quad \mathrm{q}_{j,t-2}^{*T} \quad \dot{\mathrm{q}}_{j,t-1}^T \quad \dot{\mathrm{q}}_{j,t-1}^T]^T$ |
| $s_n = \mathrm{q}_j^n$ |

We generate 5k quadrupeds for each of the reference models by randomizing the parameters provided in Table IV. We defer the reader to [29] for additional details on randomizing robot description in the RaiSim simulator [30]. Note that, we do not respect symmetry for actuation tracking gains ($K_p$ and $K_d$), torque limits ($\tau_j^{max}$) and feet friction ($\mu_f$). The rest of the parameters are symmetric with respect to corresponding bodies. Moreover, we utilize four different nominal joint configurations, $q_j^n$. These are: Configurations X, A, O and V, and are illustrated in Fig .5.

We also introduce two feasibility criteria:

1) None of the bodies should collide with another body (except for the parent).
2) The $z$ position of the feet, $c_{f,z}$ should be lower than any of the collision bodies' lowest point.

Robots which violate any of the above criteria are discarded. While this feasibility does not guarantee generation of robots capable of tracking a reference base velocity command $\mathbf{c}^*$, we still perform training over this set of generated quadrupeds, *Set 1*. Figure 17 provides examples of the generated quadrupeds. The velocity command is expressed in the base frame as

$$\mathbf{c}^* = [\mathrm{v}_x^* \mathbf{e}_x^B \quad \mathrm{v}_y^* \mathbf{e}_y^B \quad \omega_z^* \mathbf{e}_z^B]^T \tag{6}$$

where $\mathrm{v}_x^*$ and $\mathrm{v}_y^*$ are the desired heading and lateral velocities expressed along $\mathbf{e}_x^B$ and $\mathbf{e}_y^B$ respectively, and $\omega_z^*$ is the desired yaw rate around the vertical axis $\mathbf{e}_z^B$. We then repeat the robot generation process by introducing additional feasibility criteria: the root mean squared (RMS) base velocity tracking error for a randomly sampled velocity command should be lower than a threshold ($\varepsilon_v$ for linear velocity and $\varepsilon_\omega$ for angular velocity). For this, we utilize the trained DIM and CM. In our work, for an episode length of 4 s, we set $\varepsilon_v = 0.2\mathrm{m\,s^{-1}}$ and $\varepsilon_v = 0.4\mathrm{rad\,s^{-1}}$.

We then retrain the RL agent using the new set of generated robots, *Set 2*. While this approach does enable learning a platform adaptive locomotion policy, the sampling distributions in Table IV were obtained through exhaustive hand-tuning. As part of future work, we aim to investigate an auto-curricula based robot generation scheme.

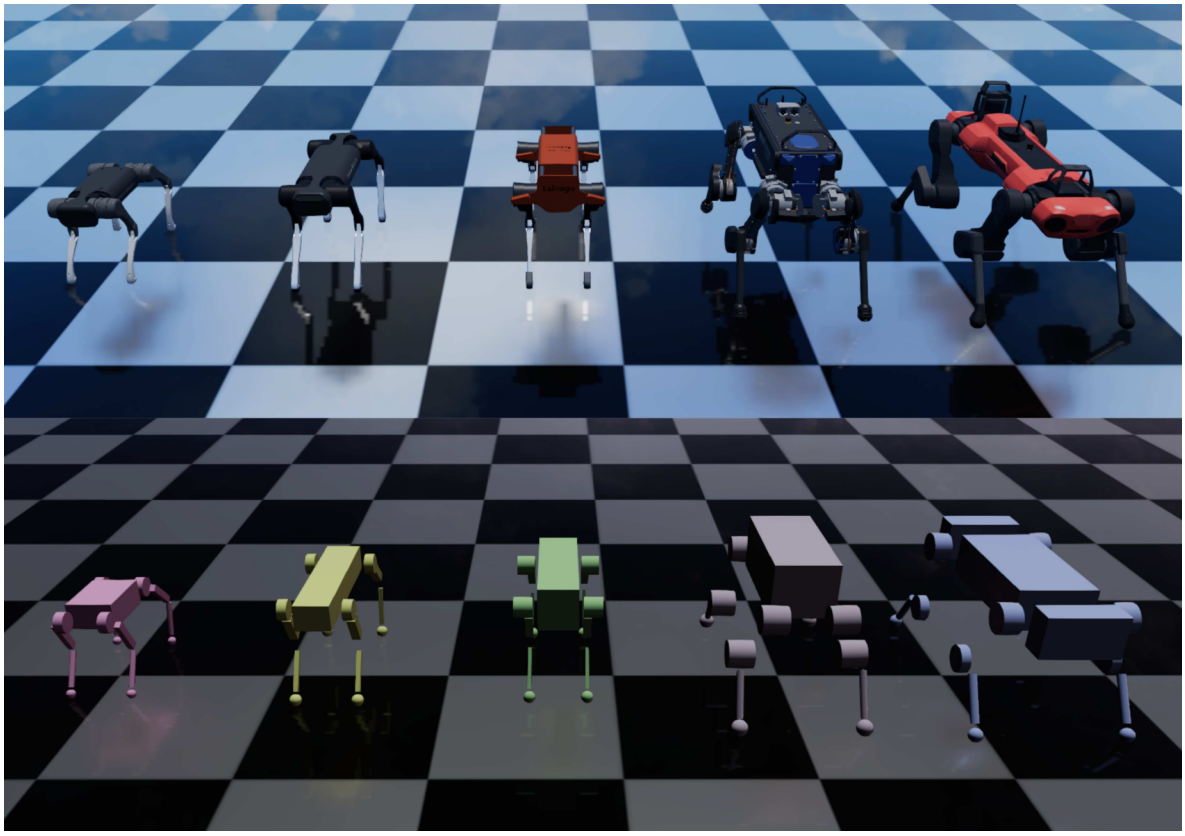


Fig. 4: Existing quadrupedal platforms (top) and their corresponding simplified models (bottom). The simplified models are used as a reference for generation of a large set of robots. The quadrupeds shown above (from left to right) are A1, Aliengo, Laikago, ANYmal B and ANYmal C.

### B. Dynamics Encoding

We consider that, for a state $x_{t-1}$, system dynamics describe the effect of a control action, $u_{t-1}$, resulting in a transition $x_t$. This can be written as

$$x_t = \mathbf{D}(x_{t-1}, u_{t-1}), \tag{7}$$

where $\mathbf{D}$ represents the dynamics model. With the dynamics encoding network, our objective is to obtain an inverse function, $\mathbf{E}$, such that, given $x_{t-1}$, $u_{t-1}$ and the transition $x_t$, we can generate a representation of the system dynamics model given by

$$l = \mathbf{E}(x_{t-1}, u_{t-1}, x_t). \tag{8}$$

The encoding network, $\mathbf{E}_d$, in Fig. 3 models the above formulation to generate a latent dynamics embedding, $l_d$, that represents the system dynamics model.

Moreover, for a quadruped, the rigid body dynamics model is given as

$$\mathrm{M}\dot{\mathrm{u}} + \mathrm{h} = \mathbf{S}^T \tau_j + \mathrm{J}^T \lambda, \tag{9}$$

where $\mathrm{M} \in \mathbb{R}^{(6+n_j)\times(6+n_j)}$ is the mass matrix relative to the joints, $\mathrm{h} \in \mathbb{R}^{6+n_j}$ comprises Coriolis, centrifugal and gravity terms, $\mathbf{S}^T = [\mathbf{0}_{n_j \times 6} \ \mathbf{I}_{n_j \times n_j}]^T$, and J is the Jacobian which maps the contact forces $\lambda \in \mathbb{R}^{n_f}$ at $n_f$ feet to generalized forces. Assuming that the effect of h remains constant throughout locomotion, we state that system accelerations mainly depend upon the actuation torques and feet contact forces. Furthermore, for state-transitions within a short duration, wherein the feet contact states do not change, another assumption can be introduced: the feet contact forces, $\lambda$, remain constant. Therefore, for *temporally local dynamics*, eq. 9 can be approximated as

$$\dot{\mathrm{u}} = \mathbf{L}(\tau_j), \tag{10}$$

where $\mathbf{L}$ represents the dynamics function mapping actuation torques to system accelerations. In this work, the control action, $a \in \mathbb{R}^{12}$, generated by the RL agent is defined as $a := \langle q_j^* \rangle$. For an RL policy that can forsee position tracking errors based on its history, the desired joint positions can be mapped into $\tau_j$ [3]. We therefore rewrite eq. 11 as

$$\dot{\mathrm{u}} = \mathbf{L}(q_j^*) = \mathbf{L}(a). \tag{11}$$

We further propose that, given an embedded representation of $\mathbf{L}$ and action, $a$, we can estimate the system accelerations. The transition prediction block, $\mathbf{P}_d$, in Fig. 3 therefore models

$$\dot{\mathrm{u}} = \mathbf{P}_d(a, l_d). \tag{12}$$

The input to the dynamics encoding network comprises $\langle s_{\mathbf{d},t-1}, a_{t-1}, s_{\mathbf{d},t} \rangle$, where $s_d \in \mathbb{R}^{33}$ is defined as $s_{\mathbf{d}} :=$

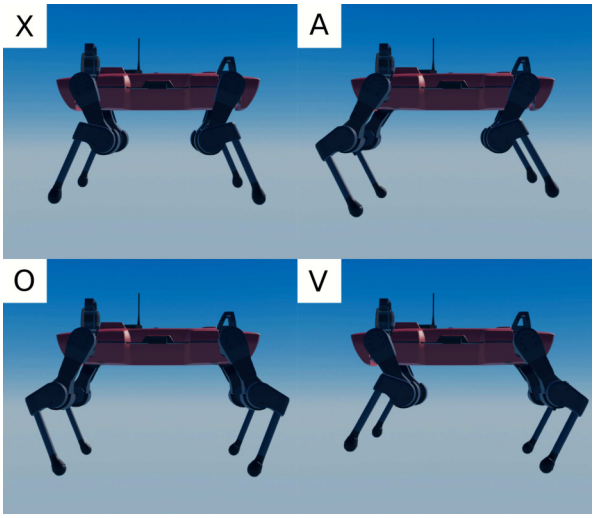



Fig. 5: Different operational joint configurations utilized during robot generation.

$\langle s_R, s_v, s_j \rangle$. The individual state term definitions are provided in Table I. Given the state-transition tuple, $\mathbf{E}_d$ generates a dynamics representation $l_d \in \mathbb{R}^{36}$.

$$l_d = \mathbf{E}_d(s_{\mathbf{d},t-1}, a_{t-1}, s_{\mathbf{d},t}). \tag{13}$$

For training the dynamics encoding network, we introduce the transition prediction network, $\mathbf{P}_d$, to estimate system accelerations, $s_{\mathbf{a},t+1} \in \mathbb{R}^{42}$, defined as $s_{\mathbf{a}} := \langle \dot{\mathbf{u}}, \dot{s_f} \rangle$, where $s_f \in \mathbb{R}^{24}$ corresponds to the vector of linear and angular velocities observed at each of the feet frames. We augment the L.H.S. of eq. 12 to include the feet accelerations. We hypothesize that, if $\mathbf{P}_d$ is able to predict the accelerations at each of the end-effectors (feet), the latent dynamics vector, $l_d$, also contains a representation of the system kinematics. Therefore, $\mathbf{P}_d$ generates the system accelerations expressed as

$$s_{\mathbf{a},t+1} = \mathbf{P}_d(l_d, a_t). \tag{14}$$

Motivated by the work of [19], [31], we further propose iteratively improving the $l_d$ estimate using a recurrent network with hidden state $h \in \mathbb{R}^{36}$.

Note that, the presented dynamics encoding and prediction architecture is based on the assumption that the dynamics majorly dependent on $a$ are temporally local. This implies that $l_d$ is only descriptive of the local system dynamics as opposed to global dynamics. This is consistent with our observations. In our preliminary experiments, we observed that, given an arbitrary input $a_{t-r}$ from the history of performed actions, where $r < t$, $\mathbf{P}_d$ failed to predict $s_{\mathbf{a},t-r+1}$. In this regard, our future work includes exploring *latent disagreement* [21] strategy to allow for learning the global dynamics representation, and then performing zero or few-shot transfers to domains not introduced during training.

### C. Locomotion MDP

The locomotion policy $\pi_\theta$, parameterized as a multi-layer perceptron (MLP) with two 512 dimensional hidden layers and softsign activation function, maps the inputs $l_d$ and $s_{\mathbf{c}} \in \mathbb{R}^{132}$ into control actions $a$. Here, $s_{\mathbf{c}} := \langle s_R, s_v, s_j, s_h, s_n, s_c \rangle$. This state-action space formulation is directly adapted from our prior work [32]. We defer the reader to the original work [3] for a detailed analysis on the design motivation of the RL MDP. In this work, we additionally condition the policy output on the latent dynamics. The reward function utilized in the locomotion MDP is given by

$$\begin{aligned}\mathbf{r}_{\mathbf{c}} = {} & 2\mathbf{r}_l + 1.75\mathbf{r}_a - 0.5d\mathbf{r}_R - (6e-5)d\mathbf{r}_\tau \\ & - (5e-4)d\mathbf{r}_{\dot{\mathrm{q}}_j} - (4e-2)d\mathbf{r}_\mu - (1e-3)d\mathbf{r}_{\ddot{\mathrm{q}}_j} \\ & - 0.5d\mathbf{r}_{\mathrm{q}_j} - 0.04d\mathrm{r}_{sm}.\end{aligned} \tag{15}$$

The individual reward term definitions are provided in Table II.

TABLE II: Reward term definitions for the locomotion MDP. Here, $K(x) := exp(-x)$, and $\mathrm{c}_{f,i}$ refers to the contact state of foot $i$. $\mathrm{c}_{f,i} = 1$ implies that the foot $i$ is in contact with the ground.

| Reward Definitions |
|---|
| $\mathbf{r}_l = K\left(2\lVert \mathbf{c}^*_{xy} - \mathbf{c}_{xy} \rVert^2\right)$ |
| $\mathbf{r}_a = K\left(2.5\lVert \mathbf{c}^*_{z} - \mathbf{c}_{z} \rVert^2\right)$ |
| $\mathbf{r}_R = (acos(R_{3,3}))^2$ |
| $\mathbf{r}_\tau = \lVert \tau_j \rVert^2$ |
| $\mathbf{r}_{\dot{\mathrm{q}}_j} = \lVert \dot{\mathrm{q}}_j \rVert^2$ |
| $\mathbf{r}_{\ddot{\mathrm{q}}_j} = \lVert \ddot{\mathrm{q}}_j \rVert^2$ |
| $\mathbf{r}_\mu = \sum \lVert \dot{r}_{F,i} \rVert, \quad \forall i, \mathrm{c}_{F,i} = 1, i \in \{0,1,2,3\}$ |
| $\mathbf{r}_{\mathrm{q}_j} = \lVert \mathrm{q}^n_j \mathrm{q}_j \rVert^2$ |
| $\mathbf{r}_{sm} = \lVert a_t - a_{t-1} \rVert^2$ |

### D. Training Setup

We train the locomotion policy using proximal policy optimization (PPO) [33]. In addition, we introduce an auxiliary loss to ensure that the encoding network learns to generate a dynamics representation. Therefore, the overall platform adaptive locomotion (PAL) loss function can be expressed as

$$\mathcal{L}_{PAL} = \mathcal{L}_{PPO} + \mathcal{L}_D, \tag{16}$$

where $\mathcal{L}_D = \lVert s_{\mathbf{a}} - \hat{s}_{\mathbf{a}} \rVert^2$ is the dynamics prediction loss, and the target accelerations, $\hat{s}_{\mathbf{a}}$, are obtained directly from the RaiSim simulator. We leverage the ability of autograd library, PyTorch [34], to automatically generate a compute graph allowing for gradient flows from the associated individual loss functions. In this regard, the gradient flow through $\mathbf{P}_d$ is associated with $\mathcal{L}_D$, gradient flow through $\mathbf{E}_d$ is associated with $\mathcal{L}_D$ and $\mathcal{L}_{PPO}$, and gradient flow through $\pi_\theta$ is associated with $\mathcal{L}_{PPO}$. The training time and hyperparameters are provided in Table III. Note that, for the recurrent encoding network, we use a batched back-propagation through time (BPTT) approach to reduce the computational overhead during policy iteration.

TABLE III: RL Training time and hyperparameters.

| Parameter | Value |
|---|---|
| Discount factor, $\gamma$ | 0.996 |
| Learning rate | 1e-3 |
| Exponential learning rate decay factor | 0.99908 |
| Batch size | 24000 |
| Mini-batch size | 6000 |
| Parallel environments, $n_{env}$ | 60 |
| Batched-BPTT Batch Size | 50 |
| Entropy coefficient | 0 |
| Value coefficient | 0.5 |
| GAE | True |
| GAE $\lambda$ | 0.95 |
| Steps per iteration, $l_{eps}$ | $400 \times n_{env}$ |
| PPO time per iteration | $\sim$2.4 s |
| Total training time | $\sim$18 h |

## IV. Results and Discussion

### A. System Identification

In our preliminary experiments, we transferred the trained dynamics encoding and locomotion policy networks to the Mini Cheetah quadruped [35]. We observed that, for the actuation tracking gains tested in simulation for smooth motions ($K_p \in [15, 30]$ and $K_d = 0.5$), the resulting behavior on the real robot was extremely aggressive. Lowering $K_p$ to a value of 3 enabled us to execute the locomotion framework on the physical system, however, the behavior was still undesirably non-smooth and unstable. The motion is shown in Fig. 6. We observed this aggressive behavior for the same network parameters even on the real A1 and ANYmal B robots. Upon further investigation, we recognized the discrepancy between the simulated and real robots originated from the actuation command tracking latency. Our training environment was based on the assumption that the command transfer from the control policy to low-level actuators is tracked immediately. This, however, is not the case on the real robots. For the Mini Cheetah and A1, we observed an actuation command tracking latency within the range of 6 ms and 8 ms, while on the ANYmal B, the latency was approximately 15 ms.

Despite the significant domain randomization associated with robot generation, the obtained policy was unable to transfer to domains not observed during training. This necessitates system identification, and implies that dynamics randomization may not be sufficient for successful sim-to-real transfer. In this regard, we introduce actuation command tracking delays in the simulator. Moreover, for systems, such as ANYmal B, that exhibit complex non-linear actuation dynamics, we also introduce actuator networks [3] trained using data collected from real ANYmal B and ANYmal C robots in the simulator for randomly assigned quadrupeds. We additionally scale the actuator network outputs in the range $[0.4, 2]$.

### B. Velocity Command Tracking

After training the locomotion policy with actuation latency, we evaluated the velocity command tracking characteristics of each of the simulated reference quadrupeds, and compared it with *test quadrupeds* that were not introduced in the training setup. The test quadrupeds include Mini Cheetah, Spot [36], HyQ [37] and HyQ2Max [38]. Note that, for each of the robots, we set the actuation tracking gains as $K_p = 1.2 m_{base}$ and $k_d = 0.1 m_{base}$. The behavior observed for tracking a reference heading velocity command, $\mathrm{v}_x^* \mathbf{e}_x^B$, of $0.5\,\mathrm{m\,s^{-1}}$ for a duration of 4 s is shown in Fig. 7 and Fig. 8.

We additionally generated the kernel density estimate (KDE) for tracking randomly sampled heading velocity commands for 100 episodes, each of 10 s duration. The corresponding KDEs are shown in Fig. 9.

For the test quadrupeds with kinematic and dynamics parameters close to those introduced during training, we observed the velocity tracking performance was better than robots with description parameters significantly farther than the sampling domains provided in Table IV. This relates to A1 (reference) and Mini Cheetah (test), and ANYmal C (reference) and Spot (test) being structurally and dynamically similar therefore demonstrating comparable control behavior. In contrast, both the HyQ and HyQ2Max are significantly heavier than the reference robots and have larger velocity tracking errors. Nonetheless, we were still able to deploy the locomotion policy on different quadrupeds without retraining. One of the drawbacks of our method is the loss in control performance for quadrupeds significantly different than used for training. We highlight this in Section V.

### C. Sim to Real Transfer

We were able to successfully transfer the trained networks on to the real ANYmal B and A1 robots as shown in Fig. 10. Figure 11 provides the tracking error estimates for 5 min of operation on each of the robots. Based on our observations, the tracking behavior on the real robots was similar to the performance of the robots in simulation. While these tests are limited, our future work includes deploying the learned control policy on even more real robots to perform a comprehensive analysis on the sim-to-real transfer aspects of the platform adaptive locomotion policy.

### D. Rough Terrain Locomotion

Although our dynamics representation is descriptive for temporally local behavior, implying that the the control policy cannot perform long-horizon planning, we tested the ability of the control policy to plan ahead, and thus recover from unstable states. In this regard, we introduced rough terrains while tracking reference base velocity commands as shown in Fig. 12. We evaluated the success rate of the quadrupeds to traverse through unexpected perturbations. For this, we considered a run successful if the robot tracked $\mathrm{v}_x^* \mathbf{e}_x^B$ of $0.5\,\mathrm{m\,s^{-1}}$ for a period of 4 s. If any of the robot bodies, except for the feet, collided with another body or terrain, we considered the run a failure. We also considered the

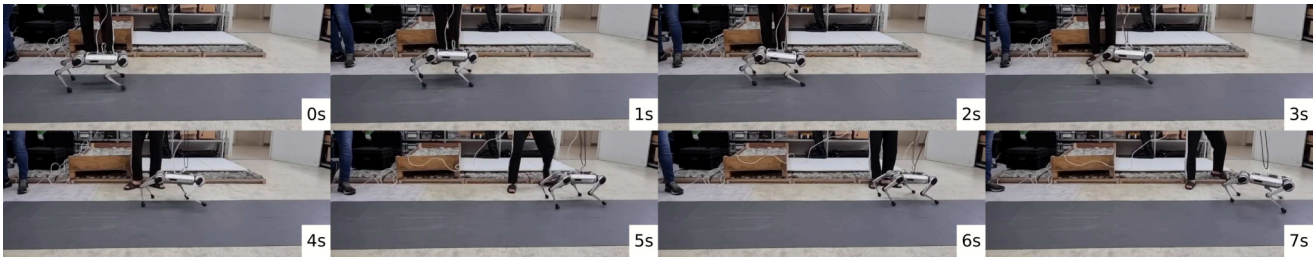



Fig. 6: Preliminary tests on the Mini Cheetah quadruped.

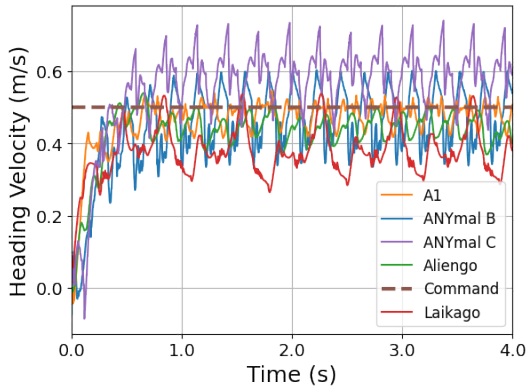



Fig. 7: The measured velocity of each of the reference quadrupeds for $\mathrm{v}_x^* \mathbf{e}_x^B = 0.5\,\mathrm{m\,s^{-1}}$.

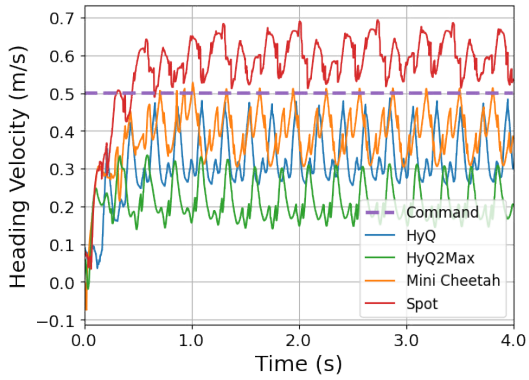



Fig. 8: The measured velocity of the test quadrupeds, not included in the RL training, for $\mathrm{v}_x^* \mathbf{e}_x^B = 0.5\,\mathrm{m\,s^{-1}}$.

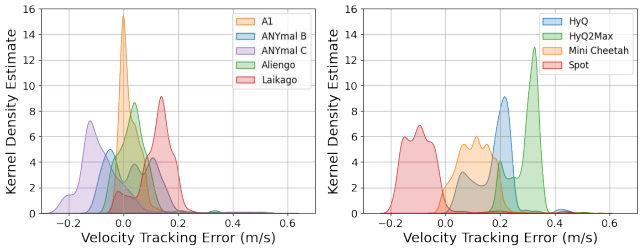



Fig. 9: The kernel density estimate of velocity tracking errors observed for randomly sampled heading velocity commands over a duration of 10 s across 100 trials.

run a failure if the mean absolute tracking error was greater than $0.3\,\mathrm{m\,s^{-1}}$. We repeated this experiment for each of the reference quadrupeds on rough terrain of varying heights. For each sampled height, we ran 100 episodes with randomized initial robot position. To generate the rough terrains, we utilized the unstructured ground presented in our previous work [39] with maximum height deviation along the highest and lowest point varying between 0 m to 2 m. The obtained success rates are presented in Fig. 13.

### *E. Latent Dynamics*

We investigated the transformation of the latent dynamics vector, $l_d$, during locomotion for tracking a heading base velocity command of $0.5\,\mathrm{m\,s^{-1}}$. Figure 15 provides visual representation of this transformation. The discontinuities in the latent dynamics were mainly associated with change in feet contact states. This is consistent with our approximation of learning a temporally local dynamics model. Additionally, for robots with similar dynamics and kinematic properties, the transformation of latent dynamics is similar. This can be seen in Fig. 16, especially, for the case of Mini Cheetah and A1.

While we do observe a change in latent dynamics during locomotion, we further investigated its relevance for obtaining a platform adaptive locomotion policy. In this case, we performed an ablation study by removing $\mathbf{E}_d$ and $\mathbf{P}_d$, and directly learning $\pi_\theta$ with the PPO loss as described in Section III-D. The policy obtained in this case, however, failed to track the control objective on majority of the quadrupeds. The tracking error density estimates for following a $0.5\,\mathrm{m\,s^{-1}}$ heading velocity command are provided in Fig. 14. As seen in this figure, the error density is close to $0.5\,\mathrm{m\,s^{-1}}$, implying, the quadrupeds fail to track the velocity command.

This result enables us to conclude that domain randomization alone is not sufficient to learn a robust control behavior. Instead, without introducing domain adaptive properties, domain randomization can result in sub-optimal control behavior [40]. In this regard, introducing additional information in the RL state space relating to the system characteristics, the RL agent can condition its behavior on the dynamics prior to ensure an optimal policy is achieved.

## V. Conclusion and Future Work

In this work, we presented an approach for obtaining a platform adaptive locomotion policy conditioned on temporally local system dynamics. We concluded that utilizing

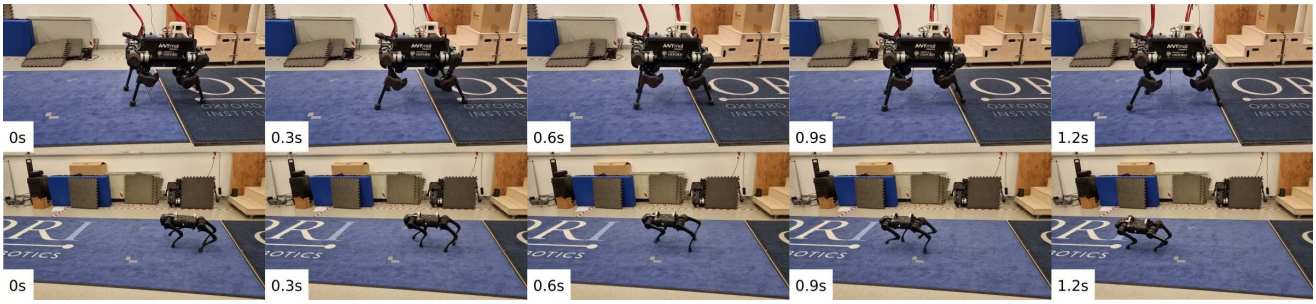



Fig. 10: Locomotion policy deployed on real ANYmal B (top) and A1 (bottom) robots.

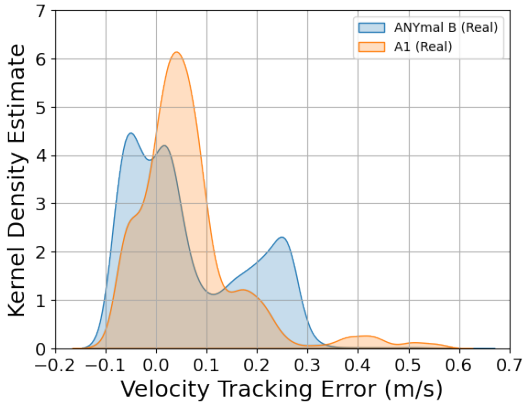



Fig. 11: The tracking error estimates obtained for the real ANYmal B and A1 quadrupeds.

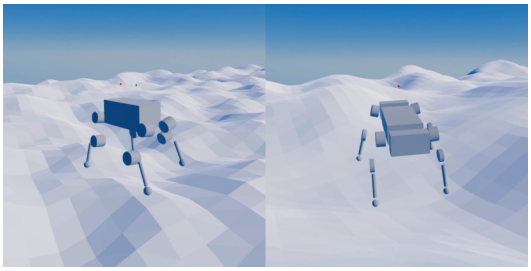


Fig. 12: ANYmal B (left) and ANYmal C (right) tracking reference base velocity commands on rough terrain.

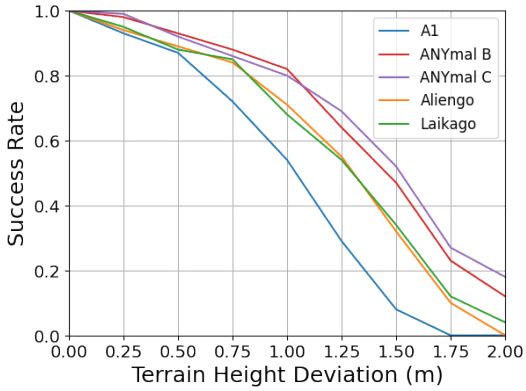



Fig. 13: Success rates obtained by each of the robots for traversing rough terrains of varying height deviations.

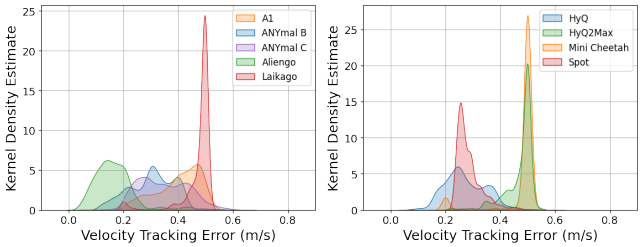



Fig. 14: Density estimates of tracking errors obtained for policy trained without the latent dynamics representation on the reference quadrupeds.

dynamics randomization techniques may lead to inferior control performance without an appropriate state space.

As part of future work, our goal is to transfer the obtained control policy to more quadrupeds, such as, ANYmal C and Mini Cheetah.

While the proposed approach allowed for successful transfer of one control policy across different quadrupeds, the current method is lacking with regards to planning ahead which is important to adapt to different tasks, such as operations in multi-contact environments. The latent disagreement [21] approach provides a relevant baseline to learn the global dynamics, and will be investigated as part of future research.

To obtain the sampling distributions for quadruped generation, this work employs a laborious hand tuning process. As an alternative, our future work aims to investigate auto-curricula strategies utilizing self-evolving quadrupedal platforms. An approach for this is through augmenting or reducing sampling distributions based on the returns observed by the RL agent.

The ability to generate an estimate of the global system dynamics online during locomotion would also allow us to approach problems such as locomotion over uneven terrain. Additionally, adaptation of the locomotion policy through online training is also an interesting future direction.

In conclusion, we propose that encouraging the RL agent to infer system dynamics allows for emergence of adaptive behavior. Additionally, learning a global dynamics prior fosters the agent to plan ahead and further adapt to tasks through zero or few-shot training.

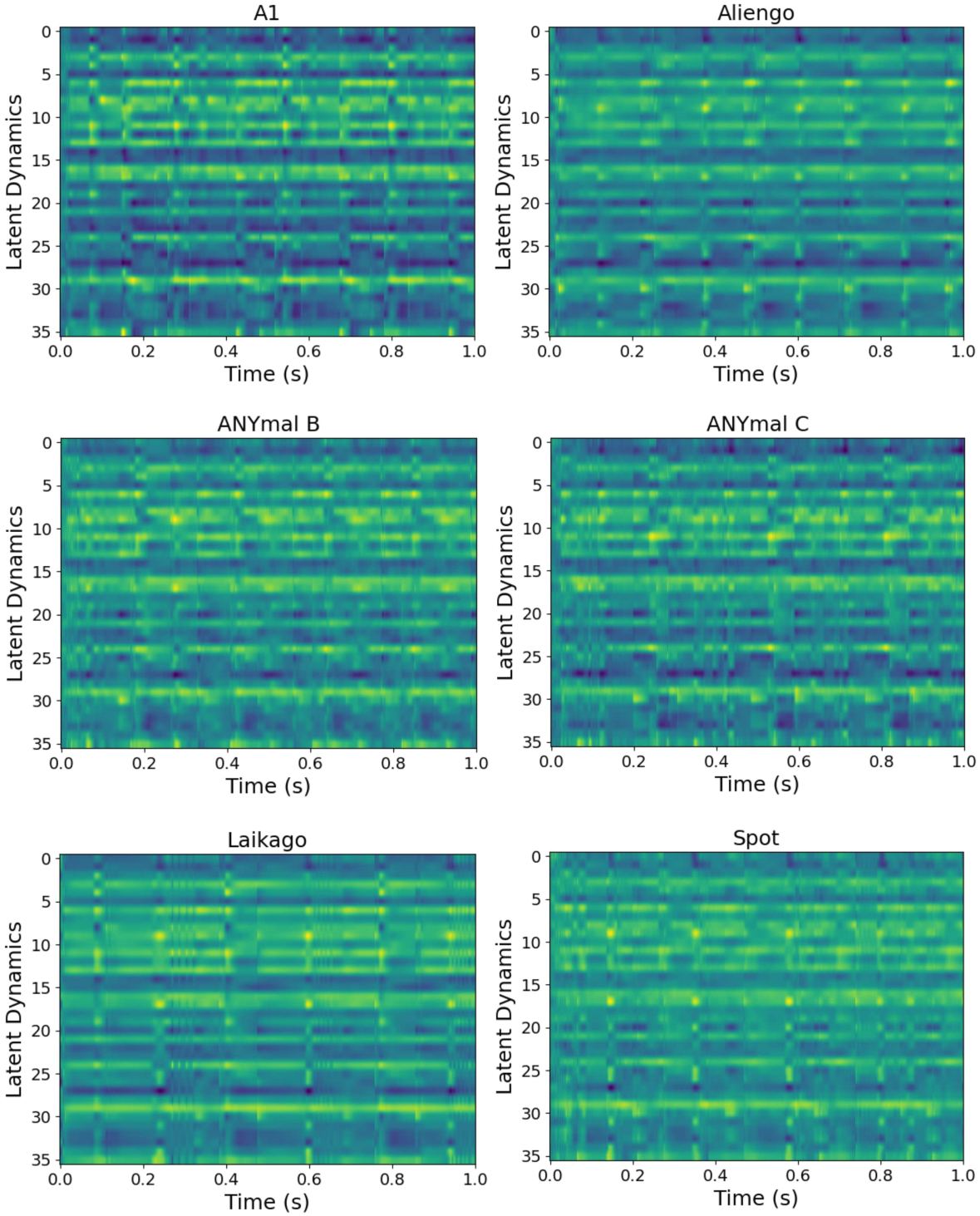



Fig. 15: The normalized latent dynamics vector, obtained for different quadrupeds, unrolled for 1 s of tracking a reference heading velocity command of $0.5\,\mathrm{m\,s^{-1}}$. The brightest region corresponds to a value of 1, and the darkest region corresponds to a value of -1.

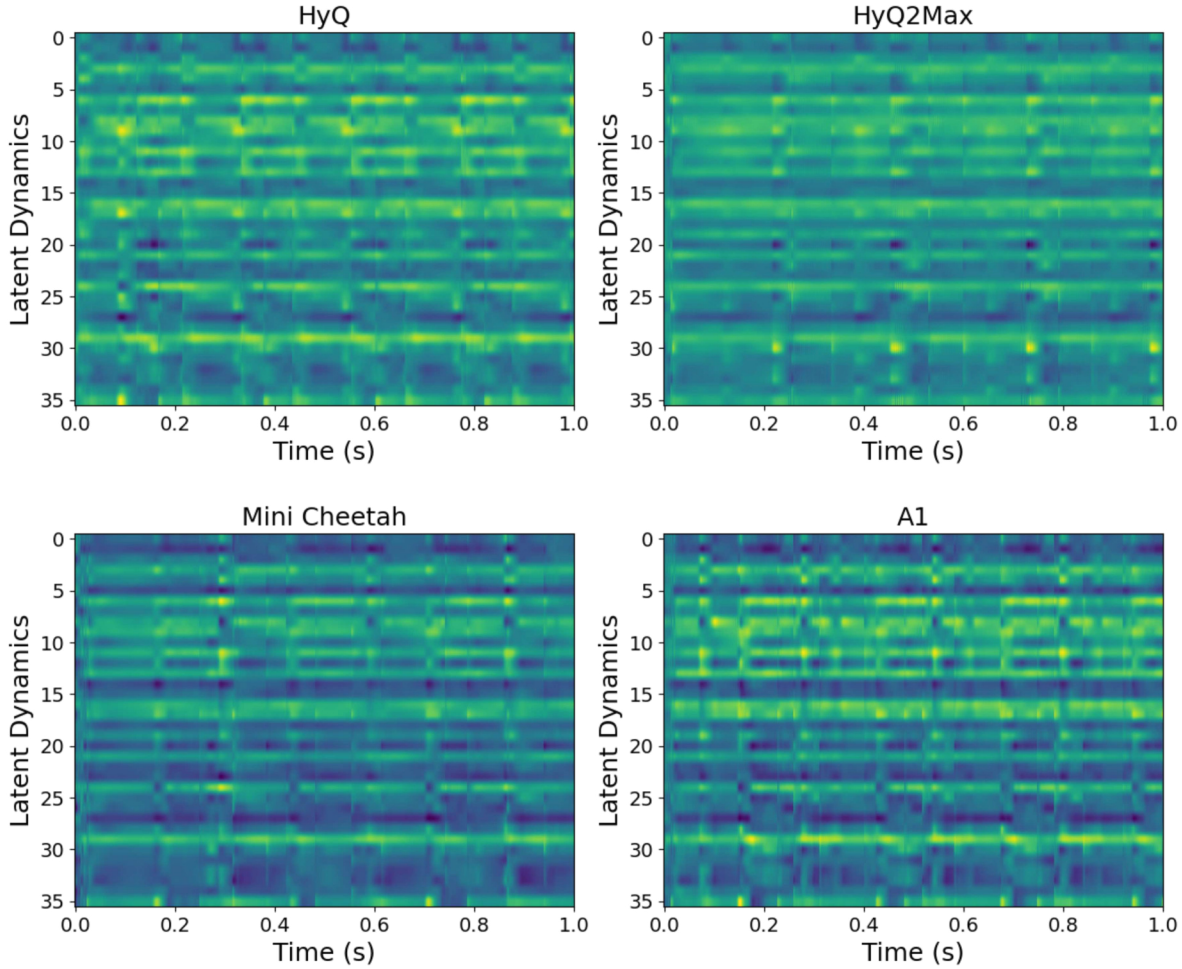



Fig. 16: The latent dynamics vector unrolled for large and heavy quadrupeds (HyQ and HyQ2Max) compared to much smaller and lighter robots (Mini Cheetah and A1).

## REFERENCES


[1] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, 2018.

[2] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[3] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau5872

[4] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[5] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.

[6] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive locomotion in rough terrain–online foothold optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5370–5376, 2020.

[7] Z. Xie, X. Da, B. Babich, A. Garg, and M. van de Panne, "Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model," *arXiv preprint arXiv:2104.09771*, 2021.

[8] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 2677–2682.

[9] J. S. De La Cruz, D. Kulić, and W. Owen, "Online incremental learning of inverse dynamics incorporating prior knowledge," in *International Conference on Autonomous and Intelligent Systems*. Springer, 2011, pp. 167–176.

[10] R. Camoriano, S. Traversaro, L. Rosasco, G. Metta, and F. Nori, "Incremental semiparametric inverse dynamics learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 544–550.

[11] R. Grandia, D. Pardo, and J. Buchli, "Contact invariant model learning for legged robot locomotion," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2291–2298, 2018.

[12] P. J. Werbos, "Learning how the world works: Specifications for predictive networks in robots and brains," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics, N.Y.*, 1987.

[13] ——, "Neural networks for control and system identification," in *Proceedings of the 28th IEEE Conference on Decision and Control,*. IEEE, 1989, pp. 260–265.

[14] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural networks," in *Advanced neural computers*. Elsevier, 1990, pp. 11–19.

[15] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, 1991, pp. 222–227.

[16] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*. Citeseer, 2011, pp. 465–472.

[17] Y. Gal, R. McAllister, and C. E. Rasmussen, "Improving pilco with bayesian neural network dynamics models," in *Data-Efficient Machine Learning workshop, ICML*, vol. 4, no. 34, 2016, p. 25.

[18] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Learning and policy search in stochastic dynamical systems with bayesian neural networks," *arXiv preprint arXiv:1605.07127*, 2016.

[19] J. Schmidhuber, "On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models," *arXiv preprint arXiv:1511.09249*, 2015.

[20] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[21] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8583–8592.

[22] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[23] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[24] Unitree, "A1," https://www.unitree.com/products/a1, accessed: 27 Oct 2021.

[25] ——, "Aliengo," https://www.unitree.com/products/aliengo, accessed: 27 Oct 2021.

[26] ——, "Laikago," https://www.unitree.com/products/laikago, accessed: 27 Oct 2021.

[27] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 38–44.

[28] E. Ackerman, "Anybotics introduces sleek new anymal c quadruped," *IEEE Spectrum. https://spectrum.ieee.org/automaton/robotics/industrial-robots/anybotics-introduces-sleek-new-anymal-c-quadruped*, 2019.

[29] R. Tech, "RaiSim: Articulated System," https://raisim.com/sections/ArticulatedSystem.html, accessed: 27 Oct 2021.

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

[31] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[32] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3642–3649, 2020.

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

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[35] MIT, "Mini Cheetah," https://robots.ieee.org/robots/minicheetah/, accessed: 27 Oct 2021.

[36] E. Ackerman, "Boston dynamics' spotmini is all electric, agile, and has a capable face-arm," *IEEE SPECTRUM*, 2016.

[37] IIT, "HyQ," https://www.iit.it/web/dynamic-legged-systems/hyq, accessed: 27 Oct 2021.

[38] ——, "HyQ2Max," https://www.iit.it/web/dynamic-legged-systems/hyq2max, accessed: 27 Oct 2021.

[39] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *arXiv preprint arXiv:2012.03094*, 2020.

[40] Z. Xie, X. Da, M. van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4955–4961.

# Appendix

TABLE IV: Robot description parameter randomization distributions utilized for generation of quadrupeds.

| **Parameter(s)** | **Sampling Distributions** | | | | |
|---|---|---|---|---|---|
| | M1 | M2 | M3 | M4 | M5 |
| $c_{\mathrm{q1},x}$, $c_{\mathrm{q4},x}$, $c_{\mathrm{q7},x}$, $c_{\mathrm{q10},x}$ | $\mathcal{U}(0.17, 0.4)$ | $\mathcal{U}(0.15, 0.55)$ | $\mathcal{U}(0.24, 0.38)$ | $\mathcal{U}(0.225, 0.45)$ | $\mathcal{U}(0.18, 0.5)$ |
| $c_{\mathrm{q1},y}$, $c_{\mathrm{q4},y}$, $c_{\mathrm{q7},y}$, $c_{\mathrm{q10},y}$ | $\mathcal{U}(0, 0.25)$ | $\mathcal{U}(0.08, 0.2)$ | $\mathcal{U}(0.12, 0.2)$ | $\mathcal{U}(0.05, 0.23)$ | $\mathcal{U}(0.08, 0.27)$ |
| $c_{\mathrm{q1},z}$, $c_{\mathrm{q4},z}$, $c_{\mathrm{q7},z}$, $c_{\mathrm{q10},z}$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.12)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.18, 0.18)$ | $\mathcal{U}(-0.22, 0.05)$ |
| $c_{\mathrm{q2},x}$, $c_{\mathrm{q5},x}$, $c_{\mathrm{q8},x}$, $c_{\mathrm{q11},x}$ | $\mathcal{U}(-0.05, 0.1)$ | $\mathcal{U}(-0.1, 0.08)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.15)$ | $\mathcal{U}(-0.1, 0.2)$ |
| $c_{\mathrm{q2},y}$, $c_{\mathrm{q5},y}$, $c_{\mathrm{q8},y}$, $c_{\mathrm{q11},y}$ | $\mathcal{U}(-0.05, 0.12)$ | $\mathcal{U}(0.08, 0.12)$ | $\mathcal{U}(0.01, 0.13)$ | $\mathcal{U}(0.015, 0.12)$ | $\mathcal{U}(-0.15, -0.05)$ |
| $c_{\mathrm{q2},z}$, $c_{\mathrm{q5},z}$, $c_{\mathrm{q8},z}$, $c_{\mathrm{q11},z}$ | $\mathcal{U}(-0.08, 0.08)$ | $\mathcal{U}(-0.06, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.07, 0.07)$ | $\mathcal{U}(-0.1, 0.06)$ |
| $c_{\mathrm{q3},x}$, $c_{\mathrm{q6},x}$, $c_{\mathrm{q9},x}$, $c_{\mathrm{q12},x}$ | $\mathcal{U}(-0.05, 0.1)$ | $\mathcal{U}(-0.05, 0.15)$ | $\mathcal{U}(-0.06, 0.18)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.04, 0.2)$ |
| $c_{\mathrm{q3},y}$, $c_{\mathrm{q6},y}$, $c_{\mathrm{q9},y}$, $c_{\mathrm{q12},y}$ | $\mathcal{U}(0, 0.08)$ | $\mathcal{U}(0, 0.05)$ | $\mathcal{U}(0, 0.06)$ | $\mathcal{U}(0.05, 0.16)$ | $\mathcal{U}(-0.2, 0.15)$ |
| $c_{\mathrm{q3},z}$, $c_{\mathrm{q6},z}$, $c_{\mathrm{q9},z}$, $c_{\mathrm{q12},z}$ | $\mathcal{U}(-0.24, -0.1)$ | $\mathcal{U}(-0.25, -0.1)$ | $\mathcal{U}(-0.27, -0.12)$ | $\mathcal{U}(-0.35, -0.18)$ | $\mathcal{U}(-0.35, -0.15)$ |
| $m_{base}$ | $\mathcal{U}(2, 12)$ | $\mathcal{U}(6, 15)$ | $\mathcal{U}(9, 22)$ | $\mathcal{U}(12, 25)$ | $\mathcal{U}(18, 50)$ |
| $m_{\mathrm{q1q2}}$, $m_{\mathrm{q4q5}}$, $m_{\mathrm{q7q8}}$, $m_{\mathrm{q10q11}}$ | $\mathcal{U}(0.25, 1.1)$ | $\mathcal{U}(0.25, 2.6)$ | $\mathcal{U}(0.25, 2)$ | $\mathcal{U}(0.5, 2.5)$ | $\mathcal{U}(1.4, 4.6)$ |
| $m_{\mathrm{q2q3}}$, $m_{\mathrm{q5q6}}$, $m_{\mathrm{q8q9}}$, $m_{\mathrm{q11q12}}$ | $\mathcal{U}(0.5, 1.6)$ | $\mathcal{U}(0.4, 2)$ | $\mathcal{U}(0.4, 2.5)$ | $\mathcal{U}(0.6, 2.7)$ | $\mathcal{U}(1.8, 5.8)$ |
| $m_{\mathrm{q3f1}}$, $m_{\mathrm{q6f2}}$, $m_{\mathrm{q9f3}}$, $m_{\mathrm{q12f4}}$ | $\mathcal{U}(0.08, 0.5)$ | $\mathcal{U}(0.1, 0.5)$ | $\mathcal{U}(0.1, 0.5)$ | $\mathcal{U}(0.15, 0.6)$ | $\mathcal{U}(0.25, 1.2)$ |
| $o_{base,x}$ | $\mathcal{U}(-0.2, 0.0175)$ | $\mathcal{U}(-0.2, 0.2)$ | $\mathcal{U}(-0.4, 0.4)$ | $\mathcal{U}(-0.28, 0.28)$ | $\mathcal{U}(-0.27, 0.27)$ |
| $o_{base,y}$ | $\mathcal{U}(-0.02, 0.02)$ | $\mathcal{U}(-0.01, 0.01)$ | $\mathcal{U}(-0.072, 0.072)$ | $\mathcal{U}(-0.075, 0.075)$ | $\mathcal{U}(-0.18, 0.18)$ |
| $o_{base,z}$ | $\mathcal{U}(-0.2, 0.2)$ | $\mathcal{U}(-0.3, 0.3)$ | $\mathcal{U}(-0.3, 0.3)$ | $\mathcal{U}(-0.2, 0.4)$ | $\mathcal{U}(-0.2, 0.4)$ |
| $o_{\mathrm{q1q2},x}$, $o_{\mathrm{q4q5},x}$, $o_{\mathrm{q7q8},x}$, $o_{\mathrm{q10q11},x}$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.2, 0.2)$ | $\mathcal{U}(-0.2, 0.2)$ |
| $o_{\mathrm{q1q2},y}$, $o_{\mathrm{q4q5},y}$, $o_{\mathrm{q7q8},y}$, $o_{\mathrm{q10q11},y}$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.01, 0.12)$ | $\mathcal{U}(-0.01, 0.12)$ | $\mathcal{U}(-0.2, 0.2)$ | $\mathcal{U}(-0.2, 0.2)$ |
| $o_{\mathrm{q1q2},z}$, $o_{\mathrm{q4q5},z}$, $o_{\mathrm{q7q8},z}$, $o_{\mathrm{q10q11},z}$ | $\mathcal{U}(-0.1, 0.2)$ | $\mathcal{U}(-0.3, 0.3)$ | $\mathcal{U}(-0.3, 0.3)$ | $\mathcal{U}(-0.3, 0.3)$ | $\mathcal{U}(-0.3, 0.3)$ |
| $o_{\mathrm{q2q3},x}$, $o_{\mathrm{q5q6},x}$, $o_{\mathrm{q8q9},x}$, $o_{\mathrm{q11q12},x}$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.2, 0.2)$ | $\mathcal{U}(-0.2, 0.2)$ |
| $o_{\mathrm{q2q3},y}$, $o_{\mathrm{q5q6},y}$, $o_{\mathrm{q8q9},y}$, $o_{\mathrm{q11q12},y}$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.2, 0.2)$ | $\mathcal{U}(-0.2, 0.2)$ |
| $o_{\mathrm{q2q3},z}$, $o_{\mathrm{q5q6},z}$, $o_{\mathrm{q8q9},z}$, $o_{\mathrm{q11q12},z}$ | $\mathcal{U}(-0.2, 0.2)$ | $\mathcal{U}(-0.3, 0.3)$ | $\mathcal{U}(-0.2, 0.2)$ | $\mathcal{U}(-0.3, 0.1)$ | $\mathcal{U}(-0.3, 0.1)$ |
| $o_{\mathrm{q3f1},x}$, $o_{\mathrm{q6f2},x}$, $o_{\mathrm{q9f3},x}$, $o_{\mathrm{q12f4},x}$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.12, 0.12)$ | $\mathcal{U}(-0.12, 0.12)$ |
| $o_{\mathrm{q3f1},y}$, $o_{\mathrm{q6f2},y}$, $o_{\mathrm{q9f3},y}$, $o_{\mathrm{q12f4},y}$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ |
| $o_{\mathrm{q3f1},z}$, $o_{\mathrm{q6f2},z}$, $o_{\mathrm{q9f3},z}$, $o_{\mathrm{q12f4},z}$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.1, 0.1)$ | $\mathcal{U}(-0.18, 0.1)$ | $\mathcal{U}(-0.18, 0.1)$ |
| $c_{\mathrm{f1},x}$, $c_{\mathrm{f2},x}$, $c_{\mathrm{f3},x}$, $c_{\mathrm{f4},x}$ | $\mathcal{U}(-0.025, 0.05)$ | $\mathcal{U}(-0.075, 0.1)$ | $\mathcal{U}(-0.05, 0.08)$ | $\mathcal{U}(-0.08, 0.08)$ | $\mathcal{U}(-0.08, 0.08)$ |
| $c_{\mathrm{f1},y}$, $c_{\mathrm{f2},y}$, $c_{\mathrm{f3},y}$, $c_{\mathrm{f4},y}$ | $\mathcal{U}(-0.08, 0.08)$ | $\mathcal{U}(-0.05, 0.05)$ | $\mathcal{U}(-0.08, 0.08)$ | $\mathcal{U}(-0.08, 0.08)$ | $\mathcal{U}(-0.08, 0.08)$ |
| $c_{\mathrm{f1},z}$, $c_{\mathrm{f2},z}$, $c_{\mathrm{f3},z}$, $c_{\mathrm{f4},z}$ | $\mathcal{U}(0, 0.025)$ | $\mathcal{U}(0, 0.025)$ | $\mathcal{U}(0, 0.025)$ | $\mathcal{U}(0, 0.025)$ | $\mathcal{U}(-0.025, 0.025)$ |
| $q_1^n$, $q_4^n$, $q_7^n$, $q_{10}^n$ | $\mathcal{U}(-0.05, 0.2)$ | | | | |
| (Configuration X) $q_2^n$, $q_5^n$ | $\mathcal{U}(0.4, 0.8)$ | | | | |
| (Configuration X) $q_8^n$, $q_{11}^n$ | $\mathcal{U}(0.4, 0.8)$ | | | | |
| (Configuration X) $q_3^n$, $q_6^n$, $q_9^n$, $q_{12}^n$ | $\mathcal{U}(-0.6, -1.2)$ | | | | |
| (Configuration A) $q_2^n$, $q_5^n$ | $\mathcal{U}(0.4, 0.8)$ | | | | |
| (Configuration A) $q_8^n$, $q_{11}^n$ | $\mathcal{U}(0.6, 1.2)$ | | | | |
| (Configuration A) $q_3^n$, $q_6^n$, $q_9^n$, $q_{12}^n$ | $\mathcal{U}(-0.9, -1.3)$ | | | | |
| (Configuration O) $q_2^n$, $q_5^n$ | $\mathcal{U}(0.4, 0.8)$ | | | | |
| (Configuration O) $q_8^n$, $q_{11}^n$ | $\mathcal{U}(0.4, 0.8)$ | | | | |
| (Configuration O) $q_3^n$, $q_6^n$, $q_9^n$, $q_{12}^n$ | $\mathcal{U}(-0.6, -1.2)$ | | | | |
| (Configuration V) $q_2^n$, $q_5^n$ | $\mathcal{U}(0.4, 0.8)$ | | | | |
| (Configuration V) $q_8^n$, $q_{11}^n$ | $\mathcal{U}(0.6, 1.2)$ | | | | |
| (Configuration V) $q_3^n$, $q_6^n$, $q_9^n$, $q_{12}^n$ | $\mathcal{U}(-0.9, -1.3)$ | | | | |
| $K_p$ | $\mathcal{U}(10, 35)$ | $\mathcal{U}(15, 45)$ | $\mathcal{U}(25, 60)$ | $\mathcal{U}(20, 60)$ | $\mathcal{U}(35, 100)$ |
| $K_d$ | $\mathcal{U}(0.1, 0.7)$ | $\mathcal{U}(0.1, 0.85)$ | $\mathcal{U}(0.1, 0.7)$ | $\mathcal{U}(0.2, 1.25)$ | $\mathcal{U}(0.5, 2)$ |
| $\tau_j^{max}$ | $\mathcal{U}(10, 25)$ | $\mathcal{U}(20, 40)$ | $\mathcal{U}(30, 50)$ | $\mathcal{U}(30, 60)$ | $\mathcal{U}(80, 140)$ |
| $\mu_f$ | $\mathcal{U}(0.3, 0.8)$ | | | | |

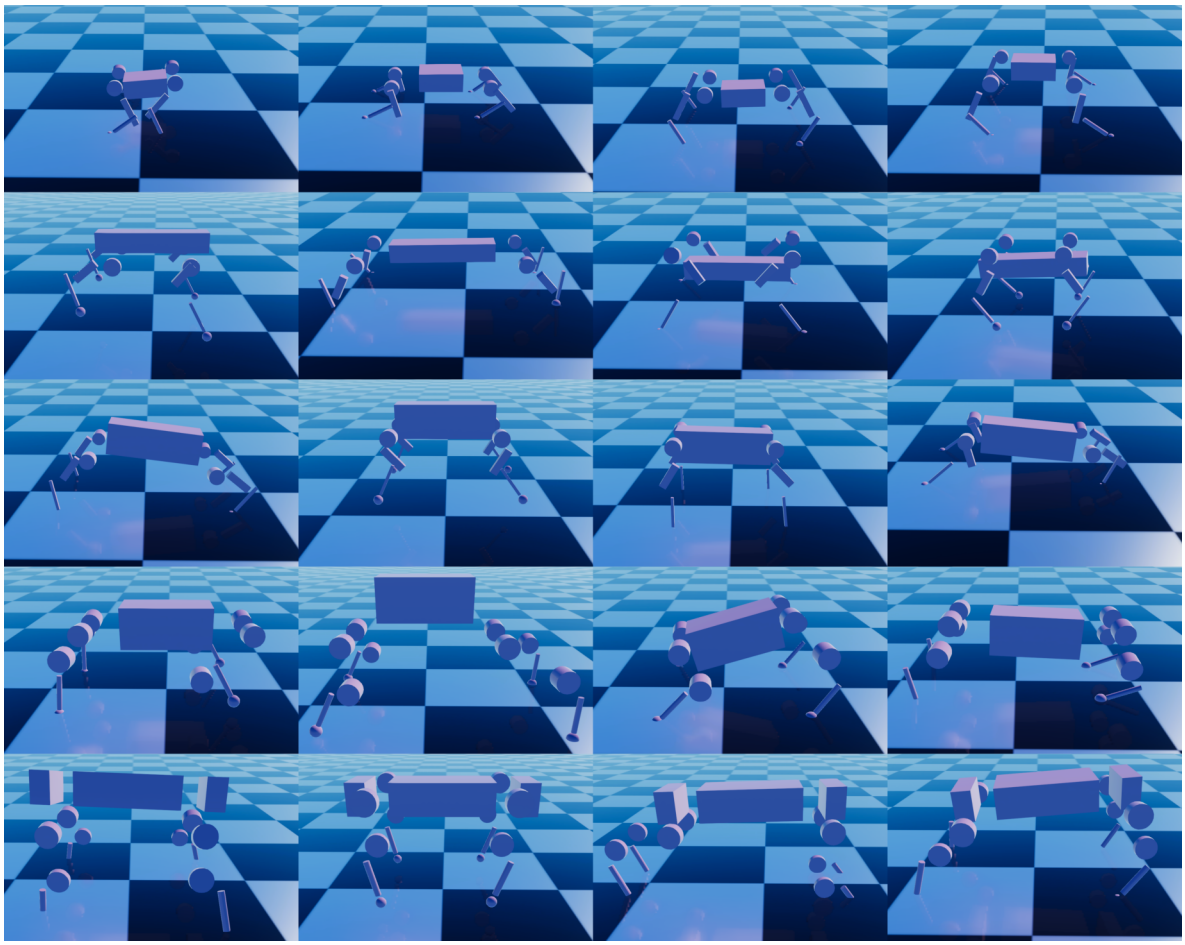


Fig. 17: Samples of generated quadruped models from Set 1 used for initial training.

# 8

# Conclusion and Outlook

This chapter summarizes the main results of this thesis and outlines prospective research directions.

## 8.1 Achievements

In this thesis, I presented control architectures for quadrupedal locomotion over flat ground and uneven terrain. I demonstrated that an RL control policy could be trained with a constrained policy optimization strategy for use in safety-critical operations while still allowing for a natural emergence of locomotion behavior that is more efficient, robust, and versatile than its model-based counterpart that employs an approximated dynamics model with predefined motion templates. I also presented an approach to guide the behavior of the control policy towards a desired optimum, eliminating the need for precise reward and hyperparameter tuning. I showed that encouraging the control policy to even implicitly model the system dynamics allows for emergence of adaptive behavior for variations in dynamic properties.

Following the narrative of adaptive behavior conditioned on the system dynamics, I further presented an approach to enhance tracking of long-horizon motion plans for

quadrupedal locomotion on uneven terrain. The proposed control architecture employed an RL policy to map history of trajectory tracking errors to additive deviations in the reference motion plan. This enabled real-time trajectory adaptation which significantly improved long-horizon trajectory tracking performance.

I presented a modular control architecture for dynamic locomotion over uneven terrain. I showed that utilizing a system adaptive RL policy to aid whole-body motion tracking allows us to realize a control framework which can perform zero-shot transfer to domains not explicitly modeled during training and development. This relates to transfer of the control framework trained using the ANYmal B robot for deployment on ANYmal C; two quadrupeds with different dynamic properties.

The initial research presented in this thesis focused on obtaining system-adaptive control characteristics through implicit modeling of the system dynamics. This implies that the control architecture does not contain modules that model or infer the system dynamics, instead, introducing a history of robot state information and tracking errors in the RL state space enables the control policy to condition its behavior for variations in system dynamics. These variations may be local, such as making or breaking contacts, or global, such as permanent changes to robot mass or link lengths. While these works demonstrated system adaptive behavior for relatively small variations in dynamics, I further presented a solution for an ambitious problem statement - *Learning a Universal Quadrupedal Locomotion Policy.* I showed that, a control architecture that infers the system dynamics through an explicit objective can be used to obtain a control policy that can be transferred to quadrupeds with significantly different kinematic and dynamic properties without requiring retraining.

## 8.2 Future Directions

While the research presented in this dissertation focused on system adaptive control through a domain randomization approach wherein the control policy, parameterized as a deep neural network, is able to interpolate within the domains observed during training,

the adaptive property of the obtained controller is still lacking - when transferred to unobserved domains, such as significantly heavier robots than used for training, the control performance significantly deteriorates. An alternative to this randomization approach is to employ a meta-learning strategy which allows to perform a zero-shot or few-shot transfer to different robots using model-based RL where the model is inferred using a dynamics encoding technique similar to the one presented in Paper V. The work of *Sekar et al.* on generating world models through task-agnostic exploration [Sekar et al., 2020] proposes a promising baseline for this future direction.

While Paper IV presented an approach for terrain-aware locomotion, the discussed method is only suitable for short-horizon motion planning and control. Deployment of legged robots in the real world requires control intelligence that can perform both long-horizon motion planning and adapt to these motion plans using a short-horizon control strategy (as presented in Paper III) to address unexpected perturbations and inaccurate motion plan tracking. This hierarchical planning and control framework can enable the robot to plan ahead for tasks such as jumping while allowing to stabilize mid-air using short-horizon trajectory adaptation. Extending the approaches presented in Paper III and Paper IV to achieve a combined hierarchical planning and control framework will certainly be an exciting direction. In this regard, both the long and short horizon motion planners can be expressed as asynchronous RL policies which perform direct mapping between robot state and terrain information to desired motion plans.

In combination, a *hierarchical terrain-aware system-adaptive motion planning and control framework* holds promise to achieve a generalizable control intelligence for legged robots that can match the level of agility exhibited by their biological counterparts.

# Bibliography


Bernardo Aceituno-Cabezas, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G Caldwell, José Cappelletto, Juan C Grieco, Gerardo Fernández-López, and Claudio Semini. Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters*, 3(3):2531–2538, 2017.

Evan Ackerman. Boston dynamics' spotmini is all electric, agile, and has a capable face-arm. *IEEE SPECTRUM*, 2016.

Evan Ackerman. Agility robotics introduces cassie, a dynamic and talented robot delivery ostrich. *IEEE Spectrum. https://spectrum.ieee.org/automaton/robotics/industrial-robots/agility-robotics-introduces-cassie-a-dynamic-and-talented-robotdelivery-ostrich*, 2017.

Evan Ackerman. Anybotics introduces sleek new anymal c quadruped. *IEEE Spectrum. https://spectrum.ieee.org/automaton/robotics/industrial-robots/anybotics-introduces-sleek-new-anymal-c-quadruped*, 2019.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *arXiv preprint arXiv:1705.08439*, 2017.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

C Dario Bellicoso, Christian Gehring, Jemin Hwangbo, Péter Fankhauser, and Marco Hutter. Perception-less terrain adaptation through whole body control and hierarchical optimization. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 558–564. IEEE, 2016.

C Dario Bellicoso, Marko Bjelonic, Lorenz Wellhausen, Kai Holtmann, Fabian Günther, Marco Tranzatto, Peter Fankhauser, and Marco Hutter. Advances in real-world applications for legged robots. *Journal of Field Robotics*, 35(8):1311–1326, 2018a.

C Dario Bellicoso, Fabian Jenelten, Christian Gehring, and Marco Hutter. Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots. *IEEE Robotics and Automation Letters*, 3(3):2261–2268, 2018b.

Martim Brandão, Omer Burak Aladag, and Ioannis Havoutis. Gaitmesh: Controller-aware navigation meshes for long-range legged locomotion planning in multi-layered environments. *IEEE Robotics and Automation Letters*, 5(2):3596–3603, 2020. doi: 10.1109/LRA.2020.2979628.

Jan Carius, René Ranftl, Vladlen Koltun, and Marco Hutter. Trajectory optimization with implicit hard contacts. *IEEE Robotics and Automation Letters*, 3(4):3316–3323, 2018.

Justin Carpentier and Nicolas Mansard. Multicontact locomotion of legged robots. *IEEE Transactions on Robotics*, 34(6):1441–1460, 2018.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

CK Chow and DH Jacobson. Studies of human locomotion via optimal programming. *Mathematical Biosciences*, 10(3-4):239–306, 1971.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302. IEEE, 2014.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022. ISSN 1476-4687. doi: 10.1038/s41586-021-04301-9. URL `https://doi.org/10.1038/s41586-021-04301-9`.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in Neural Information Processing Systems*, 33:13049–13061, 2020.

Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016.

Johannes Englsberger, Christian Ott, and Alin Albu-Schäffer. Three-dimensional bipedal walking control based on divergent component of motion. *Ieee transactions on robotics*, 31 (2):355–368, 2015.

Farbod Farshidian, Edo Jelavic, Asutosh Satapathy, Markus Giftthaler, and Jonas Buchli. Real-time motion planning of legged robots: A model predictive control approach. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 577–584. IEEE, 2017a.

Farbod Farshidian, Michael Neunert, Alexander W Winkler, Gonzalo Rey, and Jonas Buchli. An efficient optimal planning and control framework for quadrupedal locomotion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 93–100. IEEE, 2017b.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

R.J. Full and D.E. Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of Experimental Biology*, 202(23):3325–3332, 1999. ISSN 0022-0949. URL `http://jeb.biologists.org/content/202/23/3325`.

Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, page 25, 2016.

Christian Gehring, Stelian Coros, Marco Hutter, Carmine Dario Bellicoso, Huub Heijnen, Remo Diethelm, Michael Bloesch, Péter Fankhauser, Jemin Hwangbo, Mark Hoepflinger, et al. Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot. *IEEE Robotics & Automation Magazine*, 23(1):34–43, 2016.

Thomas Geijtenbeek, Michiel Van De Panne, and A Frank Van Der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)*, 32 (6):1–11, 2013.

Ghost Robotics. Vision 60. `https://www.ghostrobotics.io/robots`. Accessed: 03 Oct 2021.

Ruben Grandia, Diego Pardo, and Jonas Buchli. Contact invariant model learning for legged robot locomotion. *IEEE Robotics and Automation Letters*, 3(3):2291–2298, 2018.

Ruben Grandia, Farbod Farshidian, René Ranftl, and Marco Hutter. Feedback mpc for torque-controlled legged robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4730–4737. IEEE, 2019.

V S Gurfinkel, E V Gurfinkel, A.Yu. Shneider, E A Devjanin, A V Lensky, and L G Shtilman. Walking robot with supervisory control. *Mechanism and Machine Theory*, 16(1): 31–36, 1981. ISSN 0094-114X. doi: https://doi.org/10.1016/0094-114X(81)90047-1. URL `http://www.sciencedirect.com/science/article/pii/0094114X81900471`.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Alexander Herzog, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. Trajectory generation for multi-contact momentum control. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 874–880. IEEE, 2015.

Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491, 2016.

Shigeo Hirose and Yoji Umetani. The synthesis of basic motion regulator system for quadrupedal walking vehicle and its experiments. *Transactions of the Society of Instrument and Control Engineers*, 16(5):747–753, 1980.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44. IEEE, 2016.

Je Min Hwangbo. Simulation to real world: Learn to control legged robots. *ETH Zurich*, 2018.

Jemin Hwangbo, Joonho Lee, and Marco Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi: 10.1126/scirobotics.aau5872. URL `https://robotics.sciencemag.org/content/4/26/eaau5872`.

ICSF/EPSRC RAIN Hub. Robotics and ai in nuclear - rain hub. `https://rainhub.org.uk/our-technologies/#inspection`. Accessed: 06 Oct 2021.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Rae Jeong, Jost Tobias Springenberg, Jackie Kay, Daniel Zheng, Yuxiang Zhou, Alexandre Galashov, Nicolas Heess, and Francesco Nori. Learning dexterous manipulation from suboptimal experts. *arXiv preprint arXiv:2010.08587*, 2020.

S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 239–246 vol.1, Oct 2001. doi: 10.1109/IROS.2001.973365.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal. Fast, robust quadruped locomotion over challenging terrain. In *2010 IEEE International Conference on Robotics and Automation*, pages 2665–2670. IEEE, 2010.

Oussama Kanoun, Florent Lamiraux, and Pierre-Brice Wieber. Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. *IEEE Transactions on Robotics*, 27(4):785–792, 2011.

Joonyoung Kim, Taewoong Kang, Dongwoon Song, and Seung-Joon Yi. Design and control of a open-source, low cost, 3d printed dynamic quadruped robot. *Applied Sciences*, 11(9): 3762, 2021.

Kai Henning Koch, Katja Mombaur, and Philippe Soueres. Optimization-based walking generation for humanoid robot. *IFAC Proceedings Volumes*, 45(22):498–504, 2012.

Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 3, pages 2619–2624. IEEE, 2004.

Hendrik Kolvenbach, David Wisth, Russell Buchanan, Giorgio Valsecchi, Ruben Grandia, Maurice Fallon, and Marco Hutter. Towards autonomous inspection of concrete deterioration in sewers with legged robots. *Journal of Field Robotics*, 2020. doi: 10.1002/rob.21964. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21964`.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Scott Kuindersma, Frank Permenter, and Russ Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2589–2594. IEEE, 2014.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pages 143–156. PMLR, 2017.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Hun-Ok Lim and Atsuo Takanishi. Biped walking robots created at waseda university: Wl and wabian family. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1850):49–64, 2007.

Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2536–2542. IEEE, 2020.

Robert B McGhee and Andrew A Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3:331–351, 1968.

Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992.

Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4), July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185539. URL `https://doi.org/10.1145/2185520.2185539`.

Michael C Mozer. A focused backpropagation algorithm for temporal. *Backpropagation: Theory, architectures, and applications*, 137, 1995.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3): 1458–1465, 2018.

Derrick Nguyen and Bernard Widrow. The truck backer-upper: An example of self-learning in neural networks. In *Advanced neural computers*, pages 11–19. Elsevier, 1990.

Jorge Nocedal and Stephen J Wright. Sequential quadratic programming. *Numerical optimization*, pages 529–562, 2006.

David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous robots*, 35(2):161–176, 2013.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1): 69–81, 2014.

Ioannis Poulakakis, Evangelos Papadopoulos, and Martin Buehler. On the stability of the passive dynamics of quadrupedal running with a bounding gait. *The International Journal of Robotics Research*, 25(7):669–687, 2006. doi: 10.1177/0278364906066768. URL `https://doi.org/10.1177/0278364906066768`.

Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *2006 6th IEEE-RAS international conference on humanoid robots*, pages 200–207. IEEE, 2006.

Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. BigDog, the Rough-Terrain Quadruped Robot. *IFAC Proceedings Volumes*, 41(2):10822–10825, 2008. ISSN 1474-6670. doi: https://doi.org/10.3182/20080706-5-KR-1001.01833. URL `http://www.sciencedirect.com/science/article/pii/S1474667016407020`.

Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

Mark E Rosheim. *Robot evolution: the development of anthrobotics*. John Wiley & Sons, 1994.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *arXiv preprint arXiv:2109.11978*, 2021.

Lewis A Rygg. Mechanical horse, Feb 1893.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

Jürgen Schmidhuber. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. *arXiv preprint arXiv:1511.09249*, 2015.

Erik Schuitema, Daan GE Hobbelen, Pieter P Jonker, Martijn Wisse, and JG Daniël Karssen. Using a controller based on reinforcement learning for a passive dynamic walking robot. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pages 232–237. IEEE, 2005.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

Claudio Semini, Nikos G Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. Design of hyq–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011.

S. Seok, A. Wang, Meng Yee Chuah, D. Otten, J. Lang, and S. Kim. Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot. In *2013 IEEE International Conference on Robotics and Automation*, pages 3307–3312, May 2013. doi: 10.1109/ICRA.2013.6631038.

Daniel Shevitz and Brad Paden. Lyapunov stability theory of nonsmooth systems. *IEEE Transactions on automatic control*, 39(9):1910–1914, 1994.

Bruno Siciliano and J-JE Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Fifth International Conference on Advanced Robotics' Robots in Unstructured Environments*, pages 1211–1216. IEEE, 1991.

Jonah Siekmann, Srikar Valluri, Jeremy Dao, Lorenzo Bermillo, Helei Duan, Alan Fern, and Jonathan Hurst. Learning memory-based control for human-scale bipedal locomotion. *arXiv preprint arXiv:2006.02402*, 2020.

Manuel F Silva and JA Tenreiro Machado. A historical perspective of legged robots. *Journal of Vibration and Control*, 13(9-10):1447–1486, 2007.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, 1994.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Russ Tedrake, H Sebastian Seung, et al. Improved dynamic stability using reinforcement learning. In *International Conference on Climbing and Walking Robots (CLAWAR)*, pages 341–348, 2002.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Vassilios Tsounis, Mitja Alge, Joonho Lee, Farbod Farshidian, and Marco Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020.

Unitree. A1. `https://www.unitree.com/products/a1/`. Accessed: 27 Oct 2021.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Miomir Vukobratović and Juri Stepanenko. On the stability of anthropomorphic systems. *Mathematical biosciences*, 15(1-2):1–37, 1972.

K. Waldron and R. McGhee. The adaptive suspension vehicle. *IEEE Control Systems Magazine*, 6(6):7–12, December 1986. ISSN 2374-9385. doi: 10.1109/MCS.1986.1105145.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

P. J. Werbos. Learning how the world works: Specifications for predictive networks in robots and brains. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics, N.Y.*, 1987.

Paul J Werbos. Neural networks for control and system identification. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, pages 260–265. IEEE, 1989.

Pierre-Brice Wieber, Russ Tedrake, and Scott Kuindersma. *Modeling and Control of Legged Robots*, pages 1203–1234. Springer International Publishing, Cham, 2016. ISBN 978-3-319-32552-1. doi: 10.1007/978-3-319-32552-1_48. URL `https://doi.org/10.1007/978-3-319-32552-1_48`.

Alexander W Winkler. Optimization-based motion planning for legged robots. *ETH Zurich*, 2018.

Alexander W Winkler, C Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018.

Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Conference on Robot Learning*, pages 317–329. PMLR, 2020.

Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

Wenhao Yu, C Karen Liu, and Greg Turk. Policy transfer with strategy optimization. *arXiv preprint arXiv:1810.05751*, 2018.

# Appendices

## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

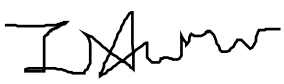
| | |
|---|---|
| Title of Paper | Guided Constrained Policy Optimization for Dynamic Quadrupedal Robot Locomotion |
| Publication Status | ✓ Published □ Accepted for Publication<br>□ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Siddhant Gangapurwala, Alexander Mitchell and Ioannis Havoutis. Published in IEEE Robotics and Automation Letters (RA-L) 2020. |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Siddhant Gangapurwala | | |
| Contribution to the Paper | *S.G. and I.H. came up with the initial idea. S.G. developed the methods into working algorithms and ran experiments in the simulation. S.G. and A.M. performed experiments on the physical robots. S.G. took the lead in writing the manuscript. A.M. and I.H. provided critical comments and helped shape the manuscript.* | | |
| Signature |  | Date | 7 September 2021 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | | |
|---|---|---|---|
| Supervisor name and title: Dr. Ioannis Havoutis | | | |
| Supervisor comments:<br>The description of the contributions is accurate. | | | |
| Signature |  | Date | 07/09/2021 |

## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

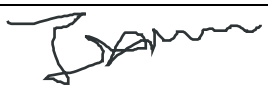
| | |
|---|---|
| Title of Paper | Rapid Stability Margin Estimation for Contact-Rich Locomotion |
| Publication Status | ✓ Published □ Accepted for Publication<br>□ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Romeo Orsolino, Siddhant Gangapurwala, Oliwier Melon, Mathieu Geisert, Ioannis Havoutis and Maurice Fallon. Published in IEEE International Conference on Intelligent Robots and Systems (IROS) 2021. |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Siddhant Gangapurwala | | |
| Contribution to the Paper | *R.O., S.G. and M.G. came up with the initial idea. R.O. and S.G. developed the methods into working algorithms. O.M. and M.G. provided software support and feedback for developing these methods. R.O., S.G., O.M. and M.G. performed experiments. R.O. took the lead in writing the manuscript. S.G., O.M., M.G., I.H. and M.F. provided critical comments and helped shape the manuscript.* | | |
| Signature |  | Date | 7 September 2021 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | |
|---|---|---|
| Supervisor name and title: Dr. Ioannis Havoutis | | |
| Supervisor comments:<br>The description of the contributions is accurate. | | |
| Signature |  | Date 07/09/2021 |

## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

| | |
|---|---|
| Title of Paper | Real-Time Trajectory Adaptation for Quadrupedal Locomotion using Deep Reinforcement Learning |
| Publication Status | ✓ Published □ Accepted for Publication<br>□ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon and Ioannis Havoutis. Published in IEEE International Conference on Robotics and Automation (ICRA) 2021. |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Siddhant Gangapurwala | | |
| Contribution to the Paper | *S.G., M.G. and I.H came up with the initial idea. S.G. developed the methods into working algorithms. M.G. and R.O. provided software support and feedback for developing the methods. S.G. ran experiments in simulation. S.G., M.G. and R.O. performed experiments on physical platform. S.G. took the lead in writing the manuscript. M.G., R.O., M.F. and I.H. provided critical comments and helped shape the manuscript.* | | |
| Signature |  | Date | 7 September 2021 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | | |
|---|---|---|---|
| Supervisor name and title: Dr. Ioannis Havoutis | | | |
| Supervisor comments:<br>The description of the contributions is accurate. | | | |
| Signature |  | Date | 07/09/2021 |

## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

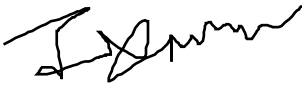
| | |
|---|---|
| Title of Paper | RLOC: Terrain-Aware Legged Locomotion using Reinforcement Learning and Optimal Control |
| Publication Status | □ Published □ Accepted for Publication<br>✓ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon and Ioannis Havoutis. Under Review for publication in IEEE Transactions on Robotics (T-RO) 2021. |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Siddhant Gangapurwala | | |
| Contribution to the Paper | *S.G., M.G., R.O. and I.H came up with the initial idea. S.G. developed the methods into working algorithms. M.G. and R.O. provided software support and feedback for developing the methods. S.G. ran experiments in simulation. S.G., M.G. and R.O. performed experiments on physical robots. S.G. took the lead in writing the manuscript. M.G., R.O., M.F. and I.H. provided critical comments and helped shape the manuscript.* | | |
| Signature |  | Date | 7 September 2021 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | | |
|---|---|---|---|
| Supervisor name and title: Dr. Ioannis Havoutis | | | |
| Supervisor comments:<br>The description of the contributions is accurate. | | | |
| Signature |  | Date | 07/09/2021 |

## Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

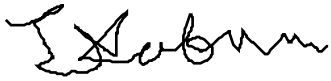
| | |
|---|---|
| Title of Paper | Learning Dynamics Conditioned Platform Adaptive Quadrupedal Locomotion Policy |
| Publication Status | □ Published □ Accepted for Publication<br>□ Submitted for Publication ✓ Unpublished and unsubmitted work written in a manuscript style |
| Publication Details | Siddhant Gangapurwala, Suyoung Choi, Mathieu Geisert, Jemin Hwangbo and Ioannis Havoutis. Published in IEEE International Conference on Robotics and Automation (ICRA) 2021. |

### Student Confirmation

| | | | |
|---|---|---|---|
| Student Name: | Siddhant Gangapurwala | | |
| Contribution to the Paper | *S.G., J.H. and I.H. came up with the initial idea. S.G. developed the methods into working algorithms. S.C. helped perform experiments and shape the methodology. M.G. provided software support and feedback for developing the methods. S.G. and S.C. ran experiments in simulation. S.G., S.C. and M.G. performed experiments on physical platforms. S.G. took the lead in writing the manuscript. S.C., M.G., J.H. and I.H. provided critical comments and helped shape the manuscript.* | | |
| Signature |  | Date | 31 October 2021 |

### Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

| | | |
|---|---|---|
| Supervisor name and title: Dr. Ioannis Havoutis | | |
| Supervisor comments:<br>The description of the contributions is accurate. | | |
| Signature  | Date | 31/10/2021 |