Bridging Vision and Dynamic Legged Locomotion

Octavio Villarreal

Istituto Italiano di Tecnologia, Italy
Università degli Studi di Genova, Italy

Thesis submitted for the degree of:
Doctor of Philosophy (32^{nd} *cycle*)

February 13, 2020

Octavio Villarreal
Bridging Vision and Dynamic Legged Locomotion
Candidate student for the *PhD Program in Bioengineering and Robotics*
Curriculum: *Advanced and Humanoid Robotics*
Genoa, Italy - December 2019

Tutor:
Dr. Victor Barasuol
Dynamic Legged Systems Lab.,
Istituto Italiano di Tecnologia

Co-Tutor:
Dr. Claudio Semini
Dynamic Legged Systems Lab.,
Istituto Italiano di Tecnologia

Head of the PhD program:
Dr. Giorgio Cannata
Dipartamento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi,
Università degli Studi di Genova

Reviewers:
Dr. Ludovic Righetti, Associate Professor
Electrical and Computer Engineering Department,
New York University

Dr. Giuseppe Oriolo, Professor
Dipartimento di Ingegneria Informatica, Automatica e Gestionale (DIAG),
Sapienza Università di Roma

This dissertation was typeset with LaTeX in Linux.
Template adapted from TU Delft template:
https://www.tudelft.nl/en/tu-delft-corporate-design/downloads/

A mis padres Paty y Alfredo,
a mi hermano Alfredo,
y sobre todo a María.

Declaration

I hereby declare that, except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

February 13, 2020 Octavio Villarreal

Acknowledgements

I would like to start by saying that doing a PhD at IIT has been my most enjoyable academic experience. Of course, there were difficult and exhausting times, but for the most part I felt happy and excited about everything that I learned and did during the past three years. There was not a single day that I woke up feeling that I made the wrong choice. This motivation has been sustained mainly by my loving family, my dear friends and my brilliant colleagues.

A special mention goes to Dr. Victor Barasuol, my tutor for the entire duration of my PhD. His expertise in control, robotics, hydraulics and legged locomotion has contributed greatly to my own personal development. As my direct tutor, Dr. Barasuol not only acted as a mentor with advice and direction; we also shared those sleepless nights of paper submissions and deadlines. I will always value the support he gave me during these stressful periods. We also became good friends and still share barbecues and football matches.

Dr. Claudio Semini, the head of the Dynamic Legged Systems (DLS) lab and my co-supervisor, has also played a very important part in these three years. As the head of the lab, Dr. Semini has always been an example of hard work and empathy. From the moment I started my PhD, I always felt that I could count on Dr. Semini's support and advice. As a co-supervisor, his insight regarding robotics and research in general has been of profound value to me. His feedback on the work here presented was crucial both on the technical side and on the way that the information is conveyed.

My experience would not have been the same without my colleagues from the DLS lab, current and former members alike. Not only were they part of my academic growth, but we also came to be very good friends. I am thankful to Lidia , Shamel, Geoff, Romeo, Andreea, Carlos G., Angelo, Niraj and Michele for offering me their friendship. I also recognize the work of those lab members and external collaborators that have been keeping the lab standing strong: Salvatore, Mohamed, Amit, Matteo, Hendrik, Marco C., Marco F., Kyle, Michel, Dhinesh, Alex, Marco R., Gennaro, Fabrizio and Jonathan. A special mention has to be made to Chundri, who has given new life to our group since his arrival. I also had the chance to learn

from and work together with Norman and Evelyn during their internships at IIT, with whom I hope to meet in the future of their promising careers. I would also take the chance to wish the best to Domingo, Abdo and Hossein who recently joined the lab.

I am specially grateful to all the administrative and technical staff of ADVR, who are crucial to keep research and development at IIT running: Silvia Ivaldi, Laura Galinta, Giulia Persano, Valentina Rosso, Monica Vasco, Floriana Sardi, Riccardo Sepe, Paolo Guria, Marco Migliorini, Stefano Cordasco, Alessio Margan, Lorenzo Baccelliere, Diego Verdelago, Mattia Arena and Phil Hudson.

I would also like to thank Professor Patrick Wensing for giving me the chance to spend the last summer of my PhD at the Robotics, Optimization, and Assistive Mobility (ROAM) lab from the University of Notre Dame. I value enormously the knowledge that Professor Wensing and the other lab members shared with me. This experience made me grow as a researcher and on the way I made very good friends. I thank all the members of the ROAM lab who made me feel welcomed in the lab from the very beginning: Itu, Bravo, John, Roopak, Li, Aravind, Martin and Tan. I would like to specially thank Taylor (and Chris) for hosting me and being the best lab mom.

I always felt loved and appreciated in Genova thanks to all the friends I made during my PhD. I thank them for always being my family away from home: Esaú, Edith, Olmo, Edwin, Paquito, Maggy, Jorge, Abril, Eamon, Fabrizia, Sep, Wiebke and Emily. My thoughts are also with my friends from Mexico, with whom I never lost contact and were with me at all times despite the distance and time differences.

Certainly, I would not be what I am today without my parents: Alfredo and Paty, and my brother: Alfredo, to whom I will dedicate some words in Spanish: Mamá y papá, gracias por ser el apoyo que me permite avanzar sin miedo al futuro. No hay certeza más grande en mi vida que el amor que me tienen. Alfredo, gracias por haber sido siempre mi inspiración y la voz más coherente cuando necesito consejo.

Finally, I would like to thank María, my talented and loving partner. Thank you for always pushing me forward and for making me feel hopeful about the future. Thank you for your unconditional support to everything that I have done and for making us a great team. You have always been the corner stone in all of my achievements.

Genoa, Istituto Italiano di Tecnologia — Octavio Villarreal
February 13, 2020

Abstract

Legged robots have demonstrated remarkable advances regarding robustness and versatility in the past decades. The questions that need to be addressed in this field are increasingly focusing on reasoning about the environment and autonomy rather than locomotion only. To answer some of these questions visual information is essential.

If a robot has information about the terrain it can plan and take preventive actions against potential risks. However, building a model of the terrain is often computationally costly, mainly because of the dense nature of visual data. On top of the mapping problem, robots need feasible body trajectories and contact sequences to traverse the terrain safely, which may also require heavy computations. This computational cost has limited the use of visual feedback to contexts that guarantee (quasi-) static stability, or resort to planning schemes where contact sequences and body trajectories are computed before starting to execute motions.

In this thesis we propose a set of algorithms that reduces the gap between visual processing and dynamic locomotion. We use machine learning to speed up visual data processing and model predictive control to achieve locomotion robustness. In particular, we devise a novel foothold adaptation strategy that uses a map of the terrain built from on-board vision sensors. This map is sent to a foothold classifier based on a convolutional neural network that allows the robot to adjust the landing position of the feet in a fast and continuous fashion. We then use the convolutional neural network-based classifier to provide safe future contact sequences to a model predictive controller that optimizes target ground reaction forces in order to track a desired center of mass trajectory.

We perform simulations and experiments on the hydraulic quadruped robots *HyQ* and *HyQReal*. For all experiments the contact sequences, the foothold adaptations, the control inputs and the map are computed and processed entirely on-board. The various tests show that the robot is able to leverage the visual terrain information to handle complex scenarios in a safe, robust and reliable manner.

Table of Contents

List of Figures

List of Tables

Glossary

Notation

x, α	scalars
$\mathbf{x}, \boldsymbol{\alpha}$	vectors
$\mathbb{R}, \mathbb{N}$	sets
$\mathbf{A}, \mathbf{B}$	matrices
$\mathbf{A}^\intercal$	matrix transpose
$\mathbf{A}^{-1}$	matrix inverse
$\mathbf{1}_n$	$n \times n$ identity matrix
$\mathbf{0}_n$	$n \times n$ zero matrix
$[\mathbf{x}]_\times$	skew symmetric matrix $= \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$

Symbols

$\dot{\mathbf{c}}$	horizontal velocity to compute neutral point
$\dot{\mathbf{c}}_d$	desired horizontal velocity in computation of neutral point
$\mathbf{c}_{xy}$	projection of the center of mass in the xy plane
d	number of CNN parameters
Δt	remaining time of a specific phase (swing or stance)

Δz	constant reference distance for reference height
D_f	duty factor
$\mathbf{F}$	vector of ground reaction forces
f_s	step frequency
$\mathcal{G}$	gait sequence
$\mathbf{g}$	gravitional acceleration vector
$\mathbf{H}$	heightmap
$\mathbf{h}_a$	actuated vector of Coriolis, centrifugal and gravitational terms
$\mathbf{h}$	vector of Coriolis, centrifugal and gravitational terms
h_s	step height
$\mathbf{h}_u$	unactuated vector of Coriolis, centrifugal and gravitational terms
$\mathbf{I}$	inertia tensor
$\mathbf{J}_c$	contact Jacobian
$\mathbf{J}_{c,a}$	actuated contact Jacobian
$\mathbf{J}_{c,u}$	unactuated contact Jacobian
k_h	horizontal velocity gain in computation of neutral point
$\mathcal{L}_i$	lift-off time of leg i
$\boldsymbol{\ell}_s$	step length
$\mathbf{M}_a$	actuated part of the joint-space inertia matrix
m	total mass of the robot
$\mathbf{M}_{au}$	bottom-left cross-term of the joint-space inertia matrix
$\mathbf{M}$	joint-space inertia matrix
$\mathbf{M}_u$	unactuated part of the joint-space inertia matrix
$\mathbf{M}_{ua}$	top-right cross-term of the joint-space inertia matrix
N_e	number of training labeled examples
N	prediction horizon
ω	natural frequency
$\boldsymbol{\omega}$	angular velocity of body with respect to its base

$\bar{\mathbf{p}}_i$	center of elliptical trajectory of leg i
$\mathbf{p}_c$	center of the plane drawn by a set of predicted footholds
$\hat{\mathbf{p}}_i$	predicted foothold of leg i
ϕ	body roll with respect to the world
$\mathbf{p}_i$	location of the ith stance foot
$\mathbf{p}_n$	neutral point
$\mathbf{p}_r$	desired foot position to compute neutral point
ψ	body yaw with respect to the world
$\mathbf{q}$	actuated joint position
$\mathbf{R}$	rotation matrix that indicates the orientation of trunk with respect to the world
$\mathbf{r}$	body position of the robot with respect to the world
$\mathbf{s}$	floating-base generalized velocity
$\boldsymbol{\tau}$	vector of joint torques
$\boldsymbol{\Theta}$	body orientation with respect to the world
θ	body pitch with respect to the world
$\mathcal{T}_i$	touchdown time of leg i
t_s	step period
t_{st}	stance phase time
t_{sw}	swing phase time
$\mathbf{v}$	output "one-hot" vector of CNN
$\mathbf{V}_f$	forward velocity
$\boldsymbol{\nu}$	floating-base robot velocity
$\mathbf{w}$	weight vector of CNN
$\mathbf{z}$	zero moment point

Acronyms

AM	additively manufactured
ANA*	anytime non-parametric A*
ANN	artificial neural network
CNN	convolutional neural network
CoM	center of mass
CoP	Center of Pressure

CPG	central pattern generator
CWC	contact wrench cone
DDP	differential dynamic programming
DLS	Dynamic Legged Systems lab
DOF	degree of freedom
FWP	Feasible Wrench Polytope
GRF	ground reaction force
HAA	hip abduction/adduction
HFE	hip flexion/extension
HPU	hydraulic pressure unit
ICP	iterative closest point
IIT	Istituto Italiano di Tecnologia
iLQR	iterative linear quadratic regulator
IMU	inertial measurement unit
ISA	Integrated Smart Actuator
KFE	knee flexion/extension
LF	left-front
LH	left-hind
LIP	linear inverted pendulum
MICP	mixed-integer convex program
MPC	model predictive control
NLP	nonlinear program
NMPC	nonlinear model predictive control
PD	proportional-derivative
QP	quadratic program
RCF	Reactive Controller Framework
RF	right-front
RH	right-hind
SLQ	sequential linear quadratic
SQP	sequential quadratic programming
TO	trajectory optimization
VFA	Vision-based Foothold Adaptation
VO	visual odometry
WBC	whole-body control
ZMP	Zero-Moment Point
ZOH	zero-order hold

Chapter 1

Introduction

The idea of a legged robot that can replace humans in monotonous or dangerous tasks has fascinated us for a long time. In fact, science fiction has fed this concept for years, and in a certain way, has given direction to research in academia and industry. Already in the 1940's, Isaac Asimov was writing short stories about running robots on the moon in the year 2015[1]*; it is safe to say that we did not meet the deadline, but we are getting close.*

Legged robots in science fiction can perceive their surroundings and interact with them similarly to humans. They receive all sorts of stimuli like sound, texture and images, reason about them and react almost instantaneously. If we want to take legged robots from science fiction to science fact, there are still several steps to be taken. They need to be able to perform reliably along a wide spectrum of mobility: from very slow, stable and carefully planned motions to fast, dynamic and reactive behaviors. Furthermore, this versatility should be displayed regardless of the scenario that the robot is facing and the type of inputs that it can process. To reach this state, several complex problems still need to be solved.

This dissertation is an attempt to tackle one of these complex problems: dynamic legged locomotion using visual feedback. In this chapter, we motivate the importance of pursuing research in this topic and outline some of the challenges that it entails. We also describe how this work contributes to solve these challenges and provide the general outline of the dissertation.

[1] Asimov, Isaac. "Runaround". *I,Robot*, published by Bantam Dell, New York, NY, 1950, pp. 25-45.

1.1 Motivation

Robots are being pushed outside of the research laboratories. The advancements in areas such as mechanical design, control and artificial intelligence are bringing these machines into the real world. This transition from the lab into our lives is "digging out" new questions for research and industry. To answer these questions, robots of all shapes and sizes have been born and compose a very wide, dense and diverse spectrum. Legged robots have been earning a visible spot inside this spectrum in recent years.

It is no surprise that legged robots are gaining momentum; they are versatile and well suited for rough and non-flat terrain. Agile and dexterous robots are now capable of dealing with complex scenarios, most of the time only based on haptics and proprioception. However, in some contexts proprioceptive-based locomotion is not enough and understanding of the environment becomes necessary. In such cases, having visual feedback can be a game-changing factor.

There are examples where vision is used to overcome rough terrain [3, 4], but only a few that do not rely on external motion capture systems [5, 6] and even fewer that employ this type of feedback when the robot is performing dynamic locomotion [2]. Research and industry are still looking for ways to use exteroceptive inputs while performing fast and reliable locomotion. A number of factors make the problem difficult: the high density nature of visual data and the time it takes to be processed; errors in the robot's pose estimate with respect to the world; and inaccuracies in the map built by the robot using its sensors. However, there have been advances in locomotion control, state estimation, machine learning and mapping in recent years that can help to overcome these limitations. In this dissertation, we use some of these novel techniques from various fields to devise locomotion control schemes with vision to enhance the robot's capabilities, and develop strategies with two specific features in mind:

1. *Sensing and computing autonomy:* the robot should be able to perceive its environment and take actions only using on-board sensors and computers.

2. *Capability to reliably perform dynamic locomotion:* the strategies should allow the robot to use visual information to act considering the environment while performing dynamic locomotion[2].

1.2 Contributions

The main novel contributions provided throughout this thesis can be summarized as:

[2]A more detailed explanation of what is meant by "dynamic locomotion" throughout the text is given in Sections 2.1 and 2.2.

1. A vision-based locomotion strategy to perform dynamic locomotion while continuously adapting foothold locations based on information of the terrain using only on-board sensing and computing. This contribution is validated both in simulation and experiments.

2. A model predictive control (MPC)-based whole-body control strategy for dynamic locomotion that considers information about the morphology of the terrain with respect to future contacts, thanks to the computational gain obtained by the convolutional neural network (CNN)-based foothold classifier using only on-board sensing and computing. This contribution is validated in simulation.

Contribution 1 has been published in IEEE Robotics and Automation Letters [7] and contribution 2 has been accepted for presentation at the IEEE International Conference on Robotics and Automation (ICRA) 2020 [8]. Additionally, two secondary contributions can be mentioned:

- A self-supervised machine learning pipeline to train a computationally light CNN-based foothold classifier to select safe footholds for legged robots. The classifier is able to output safe foothold locations within less than 0.2 ms.

- An empirical demonstration of the influence of the wrench exerted on the body due to the inertia of the legs during swing when using the centroidal dynamics model as a basis for MPC during dynamic legged locomotion.

1.3 Outline

The remainder of this dissertation is organized as follows: Chapter 2 summarizes the previous works relevant to this thesis; Chapter 3 provides an overview of HyQ and HyQReal, the robots used to evaluate the methods and strategies presented in this study; Chapter 4 explains both the self-supervised machine learning pipeline and the vision-based locomotion strategy; Chapter 5 is dedicated to the MPC-based whole-body controller with terrain awareness and the empirical demonstration of the influence of the leg inertia during dynamic locomotion; Chapter 6 concludes this dissertation with final remarks and future research directions.

Chapter 2

Related Work

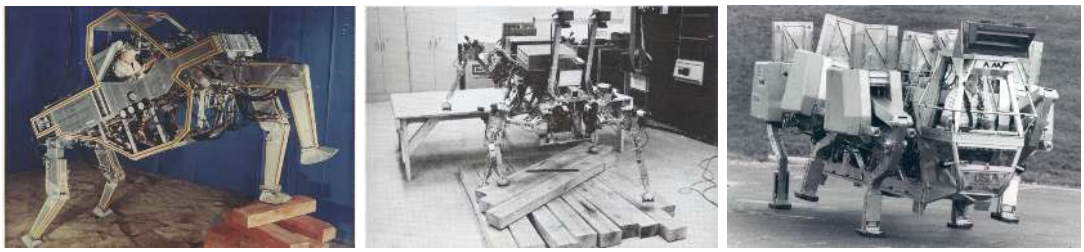
Legged robots have come a long way to achieve dynamic and versatile locomotion. They now display a wide variety of gaits and behaviors that has brought them closer to industrial, household and field applications. Nowadays, research in legged robots is paying increasingly more attention to the interaction with the environment, rather than looking at the machines as isolated systems. To understand this interaction, advancements in computer vision have made it possible for robots to use visual feedback and adapt to their surroundings.

Dynamic legged locomotion and computer vision have evolved drastically on their own in recent years, but it is still challenging to use vision in a dynamic locomotion setting. The purpose of the research outlined throughout this dissertation is to propose strategies that bring together these two features in legged locomotion systems.

In this chapter we provide a compendium of the relevant work regarding legged locomotion and visual terrain awareness to set the stage to address open questions and support the paths we took to solve them.

2.1 Legged Locomotion: Static vs. Dynamic

The idea of building legged machines can be traced all the way back to the 1960's [9]. The interest arose from the advantages that legged vehicles could have in rough terrain during space exploration and field applications, compared to their wheeled counterparts. Some examples of these early prototypes are shown in Figure 2.1. These vehicles pioneered the development of the legged robots we know today, but the motions they could achieve were very limited and their behavior was rather slow and "stiff", compared to humans and animals, or even



(a) GE Walking Truck **(b)** OSU Hexapod **(c)** ASV Hexapod

Figure 2.1: Examples of early prototypes of legged machines: *a*) General Electric Walking Truck, 1965 [20]: a $1400\,\mathrm{kg}$ hydraulically powered machine; *b*) Ohio State University Hexapod, 1976 [9]: a manually operated six-legged robot; *c*) Active Suspension Vehicle, 1984 [9]: weighing $2700\,\mathrm{kg}$, is one of the largest terrain-adapted walking machine ever built.

to wheeled vehicles. In those times, it was difficult to imagine a quadruped robot dancing to Uptown Funk or a humanoid performing parkour [10]. To get to this point, academia and industry have taken very large steps regarding mechanical design [11, 12, 13], manufacturing [14], computing, actuation [15, 16] and control [17, 18, 19] in the past 60 years.

Legged robots have walked a long path to become the athletic machines that they are today. Very much like humans, in the early stages of legged systems the initial concern was to make them walk without falling. The work of McGhee and Frank [21] on the stability properties of creeping gaits provided the fundamentals to build a stability criterion for walking robots known as the support polygon. This criterion states that the projection of the center of mass (CoM) on a plane perpendicular to gravity has to be within the convex hull of the robot's feet for the system to balance and be statically stable. The concept of the support polygon was extended by Vukobratović and Juricic [22] to propose the Zero-Moment Point (ZMP), defined as the point on the ground where the total moment produced by the inertial and gravitational forces is equal to zero, and the Center of Pressure (CoP), defined as the equivalent point of application of all the ground reaction forces. These concepts are still widely used in the legged locomotion community. More recent works [23, 24, 25, 6], use the support polygon or extensions of it as a criterion for stability during locomotion. A particularly interesting extension considers actuation limits using a six-dimensional polytope [26]. The polytope represents the set of allowable forces that the robot can generate with respect to its actuation limits, namely, the Feasible Wrench Polytope (FWP) [27]. Herein, the iterative projection algorithm [28] is used to project this six-dimensional polytope onto the two-dimensional space of the support region.

The relevance and usefulness of the support polygon, the ZMP and the CoP in the field of legged locomotion are indisputable. Their validity is such that the ZMP was revisited in

2004 by Vukobratović and Borovac [29], 35 years after it was initially proposed. They are simple, yet powerful tools to determine balance conditions on flat ground. Some researchers have even gone further and extended the ZMP to non-flat terrain [30]. Control strategies were devised to generate patterns of movement and achieve leg synchrony during locomotion (i.e., gaits) respecting these criteria. However, in the case of the support region, it only provides static balance guarantees. Because of this innate condition of static balance, the type of gaits that are generated using the support region are regarded as *static or quasi-static gaits*. Furthermore, in most cases ZMP-based strategies are not able to produce gaits that include flight phases, since the robot cannot exert any force due to lack of contact with the environment. These kind of strategies are suitable in safety-critical situations where well planned motions are required. However, using such strategies limits the performance of a robot if more dexterous and agile behavior is needed.

A wide variety of locomotion strategies emerged to grant robots the ability to perform reliably outside of the static balance criteria. Thanks to these strategies, some robots can perform highly *dynamic* motions. Some examples of these robots are shown in Figure 2.2. In this dissertation we focus on this kind of motions and in particular, strategies that can generate *dynamic gaits*. We use the term *dynamic gaits* to refer to the type of gaits in which the body is not strictly in equilibrium at all times while it is moving, forcing the robot to continuously take steps in order to prevent falling.

Gait parameters. We have briefly mentioned the term *gaits*, what it stands for (the synchronized patterns of motion of the legs during locomotion) and that they can be classified as static or dynamic. There are several types of human and animal gaits that have been widely studied in the field of biology [32]. A series of parameters related to leg synchronization and timing are used to differentiate one gait from another. Below, we list some of these parameters that will be used throughout this dissertation in order to specify a gait and a desired range of speeds [33].

(a) Mini Cheetah **(b)** Vision 60 **(c)** Spot Mini **(d)** Atlas

Figure 2.2: Examples of dynamic legged robots: *a*) Massachusetts Institute of Technology, Mini Cheetah [11]; *b*) Ghost Robotics, Vision 60 [31]; *c*) Boston Dynamics, Spot Mini [10]; *d*) Boston Dynamics, Atlas [10].

- *Swing phase time* ($t_{sw} \in \mathbb{R}$) is the period of the gait when a specific leg is in the air and it is measured in seconds (s).
- *Stance phase time* ($t_{st} \in \mathbb{R}$) is the period of the gait when a specific leg is on the ground and it is measured in seconds (s).
- *Step frequency* ($f_s \in \mathbb{R}$) is the number of steps taken per second by a specific leg and it is measured in hertz (Hz).
- *Step period* (t_s) is the inverse of the step frequency ($t_s = 1/f_s$) and it is equal to the sum of the stance and swing phase times. It is measured in seconds (s).
- *Duty factor* ($D_f \in \mathbb{R}$) is the ratio between the time that a leg is in stance phase with respect to the step period ($D_f = t_{st}/t_s$).
- *Step height* ($h_s \in \mathbb{R}$) is the maximum vertical distance (clearance) from the ground that a specific leg reaches during swing phase and it is measured in meters (m).
- *Step length* ($\boldsymbol{\ell}_s \in \mathbb{R}^3$) is the vector that indicates the direction and distance traveled by the foot with respect to the robot's base during stance phase and it is measured in meters (m).

An example of a leg trajectory is depicted in Figure 2.3 to illustrate some of the mentioned parameters. In most of the case studies, simulations and experiments in this dissertation we use a trot, which is a type of dynamic gait where the diagonal pairs of legs (left-front/right-hind and right-front/left-hind) move at the same time. However, the methods here proposed are not limited to this type of dynamic gait, and moreover can also be applied in the case of static gaits, as demonstrated in [7]. As it will be discussed in Section 4.3.1, static and dynamic gaits pose different challenges regarding foothold selection, which is one of the key components of this research.

The rest of this chapter is organized as follows: Section 2.2 gives an overview of locomotion strategies that are able to generate dynamic gaits; Section 2.4 focuses on describing key works regarding terrain awareness for legged robots; finally, Section 2.5 summarizes the material presented in this chapter and discusses how it relates to the methods and contributions of this thesis.

2.2 Dynamic Locomotion Strategies for Legged Robots

In this section we provide a summary of strategies that achieve dynamic locomotion[1] for legged robots. This type of locomotion is by itself a complex problem because it involves

[1]From here on we use the term "dynamic locomotion" to refer to the type of locomotion displayed by dynamic gaits.

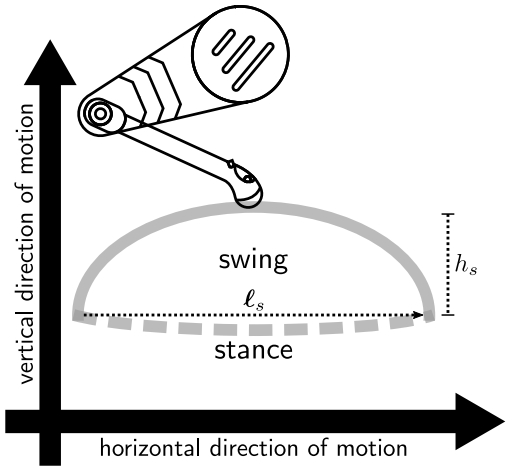


Figure 2.3: Example of a foot trajectory in the robot sagittal plane.

several components, such as balancing, attitude control, leg synchronization, contact sequence synthesis, foot trajectory generation and disturbance rejection. On top of this, to prevent failure, all of these components need to be computed within a short time window [34, 35]. To make the problem more tractable, each of these components are usually dealt with separately and tackled through a combination of approaches that take care of different parts of the process. These approaches can be heuristic, model-based or even data-driven.

One of the very early examples of this "divide and conquer" type of approach were the contributions from Raibert [36] and the Leg Laboratory, first at Carnegie Mellon University (CMU) and then at the Massachusetts Institute of Technology (MIT). Their contributions are considered pioneering works in the field. Their early prototypes included a 3D one-legged hopping machine (see Figure 2.4a). The robot was able to hop in place, and travel from one point to another on flat ground under position and velocity control while keeping its balance. The control was broken into three separate problems: hopping height, body attitude and forward velocity. The hopping height was controlled by adjusting the thrust given at each hop. The body attitude was controlled at stance phase by moving the hip. The forward velocity control was based on the so-called *neutral point*. This concept was one of the key components that drove robots to perform such athletic feats. The *neutral point* is a point localized on the ground at approximately the middle of the stance (usually defined as the projection of the hip position on the ground). If the robot places its foot at the neutral point, it causes a symmetric motion of the body about that point, causing the average momentum rate and acceleration to be equal to zero. The neutral point can be calculated as

$$\mathbf{p}_n = \mathbf{p}_r + \dot{\mathbf{c}}\frac{t_{st}}{2} + k_h(\dot{\mathbf{c}} - \dot{\mathbf{c}}_d) \tag{2.1}$$

where $\mathbf{p}_n \in \mathbb{R}^2$ is the neutral point, $\mathbf{p}_r \in \mathbb{R}^2$ is a known reference position on the ground (in this case the projection of the position of the hip on the ground), $\dot{\mathbf{c}} \in \mathbb{R}^2$ is the horizontal

(a) 3D One-Leg Hopper [36] **(b)** 3D Biped [36] **(c)** Quadruped [40]

Figure 2.4: MIT Leg Laboratory robots.

velocity of the robot, $\dot{\mathbf{c}}_d \in \mathbb{R}^2$ is the desired horizontal velocity of the robot, t_{st} is the duration of the stance phase and $k_h \in \mathbb{R}$ is a gain. These methods and ideas were transfered to more complex machines such as bipeds and quadrupeds (see Figures 2.4b and 2.4c), and are still widely referred and used nowadays due to their simplicity and effectiveness (e.g., [37, 38, 39]).

The groundbreaking works in dynamic legged locomotion from Raibert [36] in the 80's were based both in classic control theory and *heuristic* approaches. The term heuristic refers to a family of problem-solving approaches that rely mostly on observation, experience and practice rather than theory or mathematical constructs. They generally provide sufficient short-term solutions to complex problems based on empirical knowledge. Nevertheless, they are difficult to generalize due to their low reliance on theory.

The work from the MIT Leg Laboratory was further extended by Pratt et al. [41] who devised the *virtual model control*, a motion control framework that uses virtual components to create forces generated when they interact with a robot system. The virtual components consisted of spring-damper systems described in operational space [17]. Figure 2.5 shows a schematic drawing of the robot Spring Turkey with two virtual components. This strategy was applied to the biped robots Spring Turkey [41] and Spring Flamingo [42], which can be seen in Figure 2.6.

Raibert and Pratt developed both of their approaches to provide simple and intuitive solutions to complex problems based on their insight, and were able to devise complete locomotion frameworks. Other researchers have drawn inspiration from nature to solve specific parts of the locomotion problem, for example, to determine the stance phase duration in a bounding gait [43]. One representative example is the use of central pattern generators (CPGs). In animals, CPGs are neural circuits capable of producing coordinated patterns of high-dimensional rhythmic output signals while receiving only simple, low-dimensional, in-

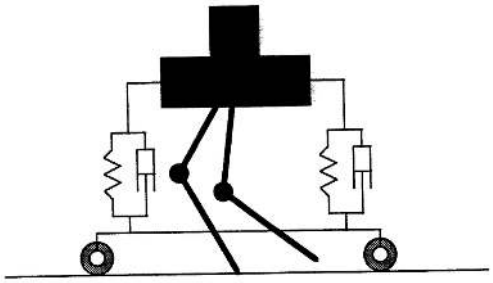
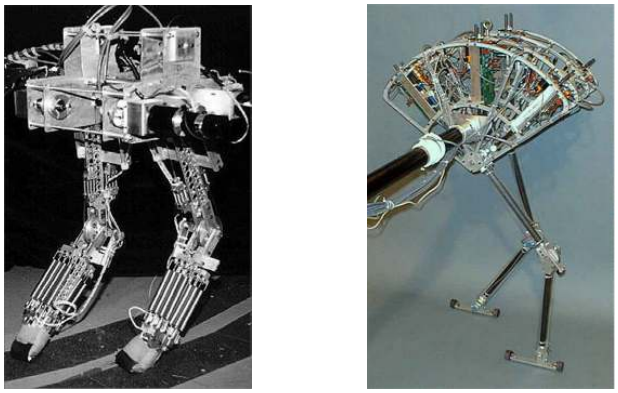


Figure 2.5: Schematic drawing of the Spring Turkey robot supported by the virtual "granny walker" composed of two virtual spring-damper systems [41].



(a) Spring Turkey [41] **(b)** Spring Flamingo [42]

Figure 2.6: Robots used to test the virtual model control framework.

put signals [44]. In robotics, this neural structures have inspired researchers to use artificial neural networks (ANNs) or systems of coupled nonlinear oscillators to assign trajectories of the legs in a synchronized manner. This approach was effectively used as an alternative to static stability criteria such as the ZMP or control laws based on virtual model control in locomotion frameworks for robots such as hexapods [45, 46, 47], quadrupeds [48, 49] and bipeds [50, 51]. In [52], the authors developed a workspace formulation for the synchronization and leg trajectory generation for the quadruped robot HyQ [53] using coupled nonlinear oscillators. The trajectory generation was part of a modular control framework (namely, the Reactive Controller Framework (RCF)) designed for quadrupedal dynamic locomotion. In this dissertation, we make use of the RCF and thanks to its modularity we are able to extend its capabilities to consider the morphology of the terrain. In Section 3.4.1 we provide a summary of this control framework.

In general, leg synchronization and gait sequence generation rely heavily on user experience and knowledge. Automatic gait synthesis and discovery is still one of the research opportunity areas in legged locomotion [54]. Furthermore, although there is plenty of research in contact/foothold planning [55, 56, 57, 58], there are not many examples of legged robots

performing dynamic locomotion in combination with terrain knowledge for contact/foothold selection in hardware. However, the advancements in body balance and attitude control have carried the load to improve dynamic locomotion in recent years. The evolution of hardware to decrease computational time and the development and implementation of algorithmic tools such as automatic differentiation [59, 60], have made it possible to implement trunk balance and attitude control strategies that rely on optimization to compute the robot joint efforts to achieve desired ground reaction forces (GRFs). In the following section we review some of the most relevant optimization-based controllers that help to produce dynamic locomotion.

2.2.1 Optimization-Based Whole-Body Control

Optimization has become one of the spears that has lead the charge in legged robotics research in recent years. Many stages of the development of legged robots such as mechanical design [61], CoM trajectory generation [55, 62, 63] and control [64] can now rely on optimization. One of the areas where it has had a large impact is in what it is referred as whole-body control (WBC). The objective of WBC for legged robots is to develop a framework that is general enough to solve multiple kinds of tasks, while keeping a solid set of rules (e.g., maintaining equilibrium) that guarantee the correct execution of such tasks [65]. In general, WBC frameworks are structured in such a way that first the equilibrium of the body is guaranteed by solving an optimization problem based on a model. We refer as model-based optimization strategies to the family of body controllers (referred from here on as trunk controllers) that rely on the computation of the control inputs based on the solution of an optimization problem, constrained to obey the behavior imposed by either a full or a simplified mathematical model of the robot dynamics.

A legged robot can be modeled as a *floating-base* kinematic tree with a specific number of joints [66]. The term *floating base* refers to a body (i.e., a link) from the tree connected to the fixed inertial frame via a 6-degree of freedom (DOF) fictitious joint. This situation is depicted in Figure 2.7. Since floating-base systems are also described as kinematic trees, they can be mathematically modeled similarly to fixed-base robots, with the consideration that the total number of DOFs is $N = n_j + 6$, where n_j is the total number of actuated joints in the robot (generally revolute or prismatic) and the additional 6 DOFs correspond to the fictitious joint that connects the inertial frame to the floating base. With this consideration in mind, the dynamic model of a floating-base system can be derived with the Euler-Lagrange equation commonly used to model mechanical systems [67]. Namely, the equation that describes the dynamics of an articulated floating-base[2] system with n_j actuated joints is given by

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{s}} + \mathbf{h}(\mathbf{q}, \mathbf{s}) = \mathbf{B}\boldsymbol{\tau} + \mathbf{J}_c^{\intercal}(\mathbf{q})\mathbf{F} \tag{2.2}$$

[2]For a complete derivation of this equation we refer the reader to Chapter 2 of Part A of [67].

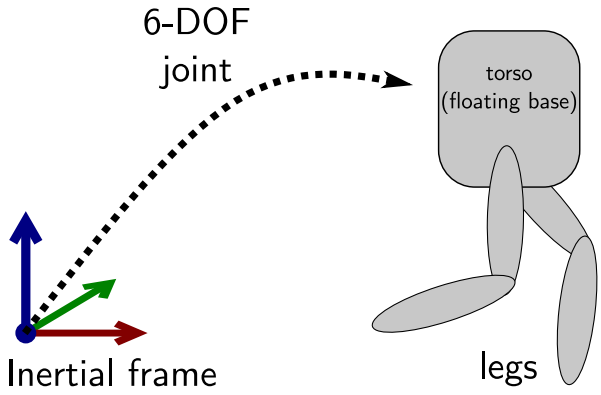


Figure 2.7: A floating-base mobile robot. The robot is fixed to the inertial frame via a 6-DOF joint and the body connected to this joint is called *floating base*. Image inspired on the lecture notes on computational robot dynamics by Roy Featherstone [1].

where $\mathbf{M} \in \mathbb{R}^{(6+n_j)\times(6+n_j)}$ is the joint-space inertia matrix, $\mathbf{s} \in \mathbb{R}^{6+n_j}$ is the generalized floating based velocity defined as $\mathbf{s} = [\boldsymbol{\nu}^\intercal \ \dot{\mathbf{q}}^\intercal]^\intercal$, where $\boldsymbol{\nu} \in \mathbb{R}^6$ is the floating-base robot velocity defined as $\boldsymbol{\nu} = [\dot{\mathbf{r}}^\intercal \ \boldsymbol{\omega}^\intercal]^\intercal$ with $\mathbf{r} \in \mathbb{R}^3$ being the 3D position of the CoM with respect to the world and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity of body with respect to the base, $\mathbf{q} \in \mathbb{R}^{n_j}$ is the vector describing the actuated joint positions, $\mathbf{h} \in \mathbb{R}^{6+n_j}$ is the vector of Coriolis, centrifugal and gravitational terms, $\mathbf{B}$ is a matrix that selects the actuated from the unactuated joints, $\boldsymbol{\tau} \in \mathbb{R}^{n_j}$ is the vector of joint torques, $\mathbf{J}_c$ is the contact Jacobian and $\mathbf{F}$ is the vector of external forces. Equation (2.2) can be separated into its actuated and unactuated parts as

$$\begin{bmatrix} \mathbf{M}_u & \mathbf{M}_{ua} \\ \mathbf{M}_{au} & \mathbf{M}_a \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\nu}} \\ \ddot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \mathbf{h}_u \\ \mathbf{h}_a \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \mathbf{J}_{c,u}^\intercal \\ \mathbf{J}_{c,a}^\intercal \end{bmatrix} \mathbf{F} \tag{2.3}$$

where $\mathbf{M}_u \in \mathbb{R}^{6\times6}$ and $\mathbf{M}_a \in \mathbb{R}^{n_j\times n_j}$ are the direct unactuated and actuated parts of the joint-space inertia matrix, whereas $\mathbf{M}_{ua} \in \mathbb{R}^{6\times n_j}$ and $\mathbf{M}_{au} \in \mathbb{R}^{n_j\times6}$ correspond to the cross terms between actuated and unactuated parts of the joint-space inertia matrix, $\mathbf{h}_u \in \mathbb{R}^6$ and $\mathbf{h}_a \in \mathbb{R}^{n_j}$ are the unactuated and actuated vectors of Coriolis, centrifugal and gravitational terms, and $\mathbf{J}_{c,u} \in \mathbb{R}^{n_c\times6}$ and $\mathbf{J}_{c,a} \in \mathbb{R}^{n_c\times n_j}$ are the unactuated and actuated contact Jacobians. The model description in (2.2) is what it is general referred to as *whole-body dynamics* or *full-body dynamics*. The elements inside the joint-space inertia matrix $\mathbf{M}$, vector $\mathbf{h}$ and the Jacobian $\mathbf{J}_c$ are dependent on the joint configuration $\mathbf{q}$ and the generalized velocity in a nonlinear fashion. Moreover, from (2.3) it can be seen that there are cross-term dependencies between the actuated and the unactuated joints of the robot, and this dependency is nonlinear as well. This high-level overview already shows that the full-body dynamics is significantly complex due to these nonlinearities and couplings.

The goal of a trunk controller is to be able to control the body pose of the robot at a specific given time. The problem can be summarized by having the control authority to determine

the following state vector

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\Theta} \\ \mathbf{r} \\ \boldsymbol{\omega} \\ \dot{\mathbf{r}} \end{bmatrix} \tag{2.4}$$

where $\boldsymbol{\Theta} \in \mathbb{R}^3$ is the orientation of the body with respect to the world expressed in Euler angles roll ($\phi \in \mathbb{R}$), pitch ($\theta \in \mathbb{R}$) and yaw ($\psi \in \mathbb{R}$), and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity of the robot with respect to the world. In other words, we would like to control the position and orientation of the robot at any given time. There are several ways to tackle this problem. One of them is to focus on the dynamics of the CoM only, the so-called *centroidal dynamics* [68, 69], which means to consider only the top part of (2.3), i.e.:

$$\mathbf{M}_u \dot{\boldsymbol{\nu}} + \mathbf{M}_{ua} \ddot{\mathbf{q}} + \mathbf{h}_u = \mathbf{J}_{c,u}^{\intercal} \mathbf{F} \tag{2.5}$$

Equation (2.5) can be further divided into the linear and rotational dynamics of the robot as

$$m\ddot{\mathbf{r}} + \mathbf{g} = \sum_{i=1}^{n} \mathbf{F}_i \tag{2.6}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} + \mathbf{M}_{ua} \ddot{\mathbf{q}} = \sum_{i=1}^{n} \mathbf{p}_i \times \mathbf{F}_i \tag{2.7}$$

where $m \in \mathbb{R}$ is the total mass of the robot, $\mathbf{g} \in \mathbb{R}^3$ is the vector of gravitational acceleration, $\mathbf{I} \in \mathbb{R}^{3\times3}$ is the inertia tensor, $\mathbf{p}_i \in \mathbb{R}^3$ is the location of the ith stance foot for $i = LF, RF, LH, RH$[3] and $n \in \mathbb{N}$ is the total number of feet on the ground. As we mentioned, we would like to be able to define a pose for the robot at any given time, i.e., be able to control the states defined in (2.4). Equations (2.6) and (2.7) do not contain the variables related to the orientation of the body (namely, $\boldsymbol{\Theta}$). To take them into account, we can consider the rotational kinematics of the robot given by

$$[\boldsymbol{\omega}]_\times = \dot{\mathbf{R}}\mathbf{R}^{\intercal} \tag{2.8}$$

where $\mathbf{R} \in SO(3)$ is the rotation matrix which transforms from body to world coordinates and the operator $[\mathbf{x}]_\times$ is the skew-symmetric matrix such that $[\mathbf{x}]_\times \mathbf{y} = \mathbf{x} \times \mathbf{y}$. Equation (2.8) can be rewritten as

$$\boldsymbol{\omega} = \begin{bmatrix} \cos\theta\cos\psi & -\sin\psi & 0 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.9}$$

then, the rate of change of the Euler angles can be obtained from (2.9) as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi/\cos\theta & \sin\psi/\cos\theta & 0 \\ -\sin\psi & \cos\psi & 0 \\ \tan\theta\cos\psi & \tan\theta\sin\psi & 1 \end{bmatrix} \boldsymbol{\omega} \tag{2.10}$$

[3]Herein i is the leg index with LF: Left-Front, RF: Right-Front, LH: Left-Hind and RH: Right-Hind.

If we define the matrix that maps the angular velocity to the angular rates on the right-hand side of (2.10) as $\mathbf{T}(\boldsymbol{\Theta})$, Equation (2.10) can be more compactly expressed as

$$\dot{\boldsymbol{\Theta}} = \mathbf{T}(\boldsymbol{\Theta})\boldsymbol{\omega} \tag{2.11}$$

Note that the inertia tensor with respect to the world $\mathbf{I}$ is defined as

$$\mathbf{I} = \mathbf{R}\mathbf{I}_B\mathbf{R}^\intercal \tag{2.12}$$

where $\mathbf{I}_B$ is the inertia tensor with respect to the body frame. This makes the inertia tensor dependent on the orientation of the body $\boldsymbol{\Theta}$. Combining (2.6), (2.7) and (2.11), we can express the total system dynamics in terms of the states in (2.4) as

$$\begin{bmatrix} \dot{\boldsymbol{\Theta}} \\ \dot{\mathbf{r}} \\ \dot{\boldsymbol{\omega}} \\ \ddot{\mathbf{r}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{T}(\boldsymbol{\Theta}) & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Theta} \\ \mathbf{r} \\ \boldsymbol{\omega} \\ \dot{\mathbf{r}} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_3 & ... & \mathbf{0}_3 \\ \mathbf{0}_3 & ... & \mathbf{0}_3 \\ \mathbf{I}^{-1}(\boldsymbol{\Theta})[\mathbf{p}_{LF}]_\times & ... & \mathbf{I}^{-1}(\boldsymbol{\Theta})[\mathbf{p}_{RH}]_\times \\ \mathbf{1}_3/m & ... & \mathbf{1}_3/m \end{bmatrix} \begin{bmatrix} \mathbf{F}_{LF} \\ \mathbf{F}_{RF} \\ \mathbf{F}_{LH} \\ \mathbf{F}_{RH} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_3 \\ \mathbf{0}_3 \\ \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} + \mathbf{M}_{ua}\ddot{\mathbf{q}} \\ \mathbf{g} \end{bmatrix} \tag{2.13}$$

or expressed in a more compact manner

$$\dot{\mathbf{x}} = \mathbf{A}(\boldsymbol{\Theta})\mathbf{x} + \mathbf{B}(\boldsymbol{\Theta}, \mathbf{p}_{LF}, ..., \mathbf{p}_{RH})\mathbf{F} + \mathbf{E}(\boldsymbol{\omega}, \mathbf{q}, \ddot{\mathbf{q}}) \tag{2.14}$$

Assuming that we have access to all the states (which can be obtained using state estimation [70]), that we can measure $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, and that we know the specific location of the feet that are in stance at the current time, matrices[4] $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{E}$ can be fully determined. One successful approach of WBC has been implemented in [25] based on two assumptions. Firstly, given that the weight of the legs is considerably less with respect to the total weight of the robot, the cross-term product $\mathbf{M}_{ua}\ddot{\mathbf{q}}$ in (2.3) and (2.13) can be considered negligible. Secondly, it can also be assumed that the Coriolis terms related to the rotational dynamics (namely, $\boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}$) can also be neglected if the robot is not rotating at fast angular speeds, which can create effects of nutation and precession. Considering these two assumptions, the controller is designed such that it tracks desired accelerations defined by the following PD workspace control laws

$$\ddot{\mathbf{r}}_d = \mathbf{K}_{p,l}(\mathbf{r}_d - \mathbf{r}) + \mathbf{K}_{d,l}(\dot{\mathbf{r}}_d - \dot{\mathbf{r}}) \tag{2.15}$$

$$\dot{\boldsymbol{\omega}}_d = \mathbf{K}_{p,r}[\mathbf{R}_d\mathbf{R}^\intercal]_\times^{-1} + \mathbf{K}_{d,r}(\boldsymbol{\omega}_d - \boldsymbol{\omega}) \tag{2.16}$$

[4]Note that $\mathbf{B}$ in (2.2) is different from $\mathbf{B}(\boldsymbol{\Theta}, \mathbf{p}_{LF}, ..., \mathbf{p}_{RH})$ in (2.14).

where $\mathbf{K}_{p,l} \in \mathbb{R}^{3\times3}$, $\mathbf{K}_{d,l} \in \mathbb{R}^{3\times3}$, $\mathbf{K}_{p,r} \in \mathbb{R}^{3\times3}$ and $\mathbf{K}_{d,r} \in \mathbb{R}^{3\times3}$ are gain matrices; $\mathbf{r}_d \in \mathbb{R}^3$ and $\boldsymbol{\omega}_d \in \mathbb{R}^3$ are the desired position and angular velocity respectively; $\mathbf{R}_d \in SO(3)$ is the rotation matrix representing the desired orientation of the body; and the operator $[\mathbf{x}]_\times^{-1}$ is the inverse operator of $[\mathbf{x}]_\times$ such that $[\mathbf{x}]_\times^{-1} = \mathbf{x}$. Finally, if the orientation of the robot is set to remain parallel to a flat surface ($\boldsymbol{\Theta} = \mathbf{0}$), and by replacing the desired linear and angular accelerations, we can rewrite (2.13) as

$$\underbrace{\begin{bmatrix} \mathbf{I}^{-1}(\boldsymbol{\Theta})[\mathbf{p}_{LF}]_\times & \dots & \mathbf{I}^{-1}(\boldsymbol{\Theta})[\mathbf{p}_{RH}]_\times \\ \mathbf{1}_3/m & \dots & \mathbf{1}_3/m \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} \mathbf{F}_{LF} \\ \mathbf{F}_{RF} \\ \mathbf{F}_{LH} \\ \mathbf{F}_{RH} \end{bmatrix}}_{\mathbf{F}} = \underbrace{\begin{bmatrix} \dot{\boldsymbol{\omega}}_d \\ \ddot{\mathbf{r}}_d \end{bmatrix}}_{\mathbf{b}} \tag{2.17}$$

then, friction consistent GRFs can be found by solving the following optimization problem

$$\begin{aligned} \underset{\mathbf{F}}{\text{minimize}} \quad & ||\mathbf{BF} - \mathbf{b}||_{\mathbf{L}} + ||\mathbf{F}||_{\mathbf{K}} \\ \text{subject to} \quad & -\mu\mathbf{F}_z \leq \mathbf{F}_x \leq \mu\mathbf{F}_z, \\ & -\mu\mathbf{F}_z \leq \mathbf{F}_y \leq \mu\mathbf{F}_z, \\ & F_{min} \leq \mathbf{F}_z \leq F_{max} \end{aligned} \tag{2.18}$$

where $\mathbf{L}$ and $\mathbf{K}$ are weighing matrices for the 2-norm, μ is the friction coefficient between the feet and the ground, $\mathbf{F}_x$, $\mathbf{F}_y$ and $\mathbf{F}_z$ are the x, y and z components of the GRFs of the stance feet, and F_{min} and F_{max} are limits for the vertical GRFs. The optimization problem in (2.18) is defined as a quadratic program (QP), and can be solved using standard solvers such as the open source QP solver QuadProg++ [71], which implements the Goldfarb-Idnani active-set method [72]. Then, the optimized GRFs $\mathbf{F}^*$ can be mapped into joint torques with

$$\boldsymbol{\tau}^* = -\mathbf{J}_{c,u}^\intercal \mathbf{F}^* \tag{2.19}$$

This formulation has been successfully implemented for dynamic locomotion. For example, this trunk controller has been combined with the RCF from [52] and a similar implementation has been used for model predictive control (MPC) in [39].

A similar approach to [25] was applied in [73] on the humanoid robot Atlas [73], but instead of neglecting the leg inertia and the Coriolis effects, the centroidal dynamics represented by (2.13) are linearized at every iteration and set as an equality constraint. Then a QP that minimizes the difference between the desired and current states is solved. A step further was taken in [74] for the DARPA Robotics Challenge (DRC), where they set the problem in terms of the joint accelerations $\ddot{\mathbf{q}}$, the joint torques $\boldsymbol{\tau}$ and the GRFs $\mathbf{F}$ for given joint position and velocities $\mathbf{q}$ and $\dot{\mathbf{q}}$, respectively. Herein, the problem is stated as well in the form of a QP, and the equations of motion are set as equality constraints, with no need for assumptions or simplifications. The work of [74] is very similar to the whole-body controller used in [75],

which is based on [76] with some differences in the way the constraints are set and in the cost function. In [77], a whole-body controller is devised for the humanoid robot TORO [78] based on the solution of a QP, however, the dynamics are stated such that the controller allows interaction of the end-effectors with the environment in a compliant manner, although no walking motions are shown.

More recently, it has been shown for quadrupeds as well that the problem can be formulated in terms of the total wrench exerted on the body instead of the GRFs [79]. The advantage is that the joint accelerations $\ddot{\mathbf{q}}$ and the GRFs $\mathbf{F}$ can be jointly optimized by setting an equality constraint that encodes the physical consistency between these two control inputs, without having to rely on the assumptions regarding the weight of the legs and limited rotational speeds. This controller was further extended to drop the rigid terrain assumption and work also for soft terrain models [80]. Similarly in [81], a whole-body dynamic controller based on the analysis of the dynamics of the CoM for a humanoid robot and went even further by implementing the friction constraints in terms of the contact wrench cone (CWC) [82].

Whole-body controllers have demonstrated their value and have been key for the progress of dynamic locomotion. They can instantaneously guarantee balance to allow the robot to execute other tasks. The next stepping stone has been to translate this instantaneous conception of balance to a more plan-oriented scheme and provide safety guarantees (such as balance) to perform a task in the future, for example moving from point A to B. This kind of problem belongs to the realm of trajectory optimization (TO).

2.2.2 Trajectory Optimization

Unlike wheeled robots, to move from a specific pose to another, legged robots need to define a specific and feasible sequence of contacts/footholds in order to reach a goal. This scenario is depicted in Figure 2.8. Furthermore, the GRFs on each foot have an impact on the overall acceleration of the base, and they depend on the weight of the robot, the torque at each joint, the feet contact positions and the robot pose itself. The problem becomes even more complex when the flat terrain assumption is dropped. In summary, a task as simple as moving from point A to B requires to plan for a contact/foothold sequence and a trajectory of the CoM. Solving this problem for a legged robot requires to find a large number of free variables for a determined planning horizon such as:

- body orientation ($\boldsymbol{\Theta}$);
- CoM position ($\mathbf{r}$);
- contact positions ($\mathbf{p}_i$);
- gait sequence ($\mathcal{G}$);

- joint torques ($\boldsymbol{\tau}$);
- ground reaction forces ($\mathbf{F}$).

The selection of these variables have effects from one to another, and cannot be selected arbitrarily. To find a feasible combination, the model from (2.13) can be used to constrain the selection of these variables to obey the dynamics of this system. If we recall (2.13), matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{E}$ are fully determined for the current time. However, these matrices for "future" states are not defined because $\boldsymbol{\Theta}$, $\mathbf{r}$ and $\mathbf{p}_i$ are optimization variables. Moreover, $\mathbf{p}_i$ for future states depends on the gait pattern $\mathcal{G}$.

It is possible to find values for all of the variables listed above at once using optimization, but the nature of the optimization problem is nonlinear, non-convex and involves a large combinatorial space. This is done in [56], and they render the problem slightly more tractable by modeling the robot as a rigid body subject to contact forces exerted by the feet (i.e., the centroidal dynamics model [68, 69]). The gait sequence $\mathcal{G}$ is encoded in the problem by optimizing the phase durations for each of the legs. The optimization problem is defined as a feasibility problem subject to a set of nonlinear constraints that include the centroidal dynamics and contact-consistent behavior (friction). This problem belongs to the category of nonlinear programs (NLPs) and it is solved using an interior point method solver, namely ipopt [83]. The approach was tested both in simulation and hardware and it is able to find feasible contact sequences and CoM trajectories from a starting point to a goal, without pre-specifying any of the variables listed above. However, for highly uneven terrain it was found that the solver can get trapped in local minima and additionally, the solver can take up to approximately 4 s to compute a solution, which limits its application to static environments with no perturbations in between the starting point and the goal. An alternative approach to use the centroidal dynamics, is to use the sequential linear quadratic (SLQ) algorithm [84], which solves a series of QPs by sequentially linearizing the system dynamics. In [62], this method is implemented in combination with differential dynamic programming (DDP) for quadrupedal locomotion and were able to generate both static and dynamic gaits without a

Figure 2.8: Example of a contact sequence to reach a specific robot pose. The robot moves from pose A to pose B following the series of footholds denoted by the green discs on the ground.

pre-specified gait sequence. However, it also suffers from long computational times (can take up to slightly less than a minute). A similar approach is taken in [64] using SLQ, with the difference that the problem is formulated as a switched system with a specified gait sequence, only optimizing the switching times.

To reduce the computational cost, on top of specifying the gait sequence, TO approaches generally divide the problem into two separate components: reference contact/foothold optimization and CoM trajectory generation. The advantage is that instead of having one nonlinear problem with many optimization variables, as in the previous examples, you can decompose it into problems that may not be as difficult or time consuming to solve as the original problem. Such is the case of [85], where they decompose the optimization problem in two parts: one for the optimization of the CoM trajectory and one for the optimization of the foothold location. The CoM trajectory optimization is still a nonlinear problem which is solved using sequential quadratic programming (SQP) [86]. The problem is, however, of lower dimension with respect to the case that optimizes foothold locations jointly. Furthermore the optimization of the foothold location consists of a quadratic cost function with linear constraints, hence, a QP. With this formulation, they are able to compute GRFs, joint torques, and foothold locations for a horizon of one gait cycle in a time that goes up to 9 ms. The method is implemented on the quadruped robot ANYmal [12] and is able to perform multiple dynamic gaits. Other approaches that decompose the problem into contact planning and CoM trajectory include [87], [88] and [89] in humanoids, however in the case of [87] no hardware experiments are shown and in the case of [88] and [89] less dynamic motions are shown.

Solving a trajectory optimization problem at a fast rate is directly connected to its applicability to MPC. In recent years this control methodology has gained popularity in robotics because of its inherent robustness and intuitiveness to formulate complex problems such as the ones related to legged locomotion. In the following section we provide a general overview of what MPC is and some examples where it has been applied in robotics.

2.2.3 Model Predictive Control

Model predictive control (MPC) is a control methodology that is based on five key components [90]:

1. process model;
2. performance index or cost function;
3. constraints;
4. optimization;

5. receding horizon principle.

The *process model* is a mathematical description of the system that needs to be controlled. It serves two purposes: *a*) the prediction of the behavior of the future output of the process on the basis of inputs; and *b*) the calculation of the input signal to the process that minimizes the given cost function. The *performance index* or *cost function* is the criterion that reflects the reference tracking error and the control action. It is formulated measuring the reference tracking error and the control action. *Constraints* are generally set to match practical situations such as limits on the control input, states and output. Once a cost function and constraints are defined, an *optimization* technique can be used to compute the set of future control signals that minimize the cost function for a given time in the future. It can be argued that the most important component of MPC is the use of the *receding horizon principle*. This principle arises from the concept of *prediction horizon*. The *prediction horizon* is the time (or time-steps in the discrete case) in the future that the optimization will consider to compute control signals. The *receding horizon principle* states that out of all the future control signals computed using optimization, only the signal corresponding to the first time-step of the prediction horizon is applied and the horizon is shifted one time-step to restate the optimization based on the new information coming from the measurements.

MPC is rapidly gaining momentum among the legged robotics community. Some years back, MPC was thought to be not suitable for robotics applications, since the size of the optimization problems required long computational times with respect to the fast dynamics involved in the process. This limited the application of this type of framework to processes with relatively slow dynamics, such as the ones in the chemical industry [91]. However, due to the advancements in computer processing and algorithmic efficiency, MPC has made its way into the field of robotics. The spectrum of strategies when using MPC varies mostly depending on the trade-off between model accuracy and computational cost.

The initial difficulties that researchers faced when trying to implement MPC were related to computational efficiency. Using a complete nonlinear dynamics model such as the one described in (2.13) was not feasible. These challenges gave rise to the use of *template models* initially proposed by Full and Koditschek [92] after years of studying the locomotion of insects [93, 94]. Template models are mathematical representations that try to capture the essential dynamics of a system, using a minimum number of variables and parameters. Roboticists took the idea of template (or simplified) models from biology to have a more intuitive understanding of the dynamics when developing robotic systems and make their analysis and control more tractable. Technically speaking, the hypothesis is that template models are attracting invariant submanifolds on which the restricted dynamics take a form prescribed to solve the specific task at hand [95].

The seminal work of Kajita et al. [96] explains how the linear inverted pendulum (LIP) model can be used as a simplified model for the control of legged robots. Herein, a robot standing on

its support leg is modeled as a pendulum connecting the foot with the CoM of the robot. The model is linear since the motion of the center of mass is constrained to a plane perpendicular to gravity and stability is guaranteed using the ZMP criterion (see Section 2.1). The model can be described by

$$\mathbf{z} = \mathbf{c}_{xy} - \frac{1}{\omega^2}\ddot{\mathbf{c}}_{xy} \tag{2.20}$$

where $\mathbf{z}$ is the zero moment point, $\mathbf{c}_{xy}$ is the projection of the CoM onto the xy plane, $\omega = \sqrt{g/c_z}$ is the natural frequency of the pendulum, where g is the acceleration of gravity and c_z is the z component of the CoM position. A biped robot was able to walk using generated patterns based on this simple model (and modified versions of it, like the cart-table model [97]), and was also able to go through complex scenarios such as stairs and stepping stones. Some other relevant results on WBC and trajectory optimization of humanoids and quadrupeds are based on simplified models [98, 55].

The reduced complexity of simplified models allows to optimize and recompute trajectories and control signals during execution. Such is the case that MPC-based locomotion controllers have been devised for humanoids [99, 100, 101, 102] and quadrupeds [103].

However, as in the case of the LIP model, these approaches work well under a number of assumptions such as zero vertical acceleration or no angular dynamics, and some consider that friction will always be sufficient to maintain contact with the ground. Nevertheless, simplified models are still relevant to dynamic legged locomotion, not only because of their historic significance but also because they can be used to describe complex dynamics and reduce the computational load of larger optimization problems.

To drop these assumptions, the work of [39] considers a simplified version of the centroidal dynamics model, neglecting leg inertia, and ignoring the effects of non-zero roll/pitch on the dynamics of the body. The optimization problem is still convex and it is solved in real-time as a QP. The strategy keeps the robot in balance during a range of highly dynamic motions (e.g., trot, bound, and gallop) on the Cheetah 3 robot [11].

Some other approaches tackle the trade-off between computational cost and model accuracy not by simplifying the whole-body dynamics, but by relying on reducing the computational cost of the optimization problem solver. One of the very first examples where the full dynamics model was used to implement a nonlinear model predictive control (NMPC) controller was given in [104]. Therein, the MPC optimization problem is solved using the iterative linear quadratic regulator (iLQR) algorithm. Another remarkable example is the one devised in [105]. Their approach performs NMPC and relies on a custom solver based also on the iLQR algorithm and exploits automatic differentiation [60]. The work shown in [106] improved on the implementation of the SLQ algorithm for trajectory optimization used in [64] by computing the backward pass in parallel. This improvement allows them to run their trajectory optimization algorithm in an MPC fashion.

2.2.4 Data-driven approaches

Machine learning and artificial intelligence have become one of the most discussed research topics in recent years. This technology seems to be taking off due to the increasing ability to gather and process large amounts of data, which was one of its main bottlenecks in the past. Machine learning (and specifically deep learning) have proven to outperform previous approaches in areas such as image classification, natural language processing and autonomous driving. Such popularity has made its way to the field of legged locomotion.

In particular, results regarding deep reinforcement learning have caught the eye of many. In 2017, Google DeepMind [107] released a paper [108] that featured a deep reinforcement learning strategy to teach locomotion in rough environments to two and four-legged simulated agents, with only proprioceptive sensing (joint position and torque), its position with respect to the environment and a profile of the terrain ahead. The learning scenario consisted on an obstacle track that the agent traversed in the longitudinal direction. The reward was designed such that it encouraged forward velocity and penalized deviation from the center of the track. With this simple reward function, the agents were able to traverse a complex, dynamic environment, even in the presence of disturbances. Similarly, the work in [109] developed a hierarchical deep reinforcement learning framework to teach high-level locomotion skills to navigate complex environments or manipulate objects to a biped agent, and extended their work in [110] to learn complex skills such as running and performing backflips, cartwheels and sideflips, from example motion clips. Although the results of all three methods are impressive, the amount of data required is in the order of 1×10^7 examples and the learning process can take several days. Furthermore, these approaches have not yet been tested in hardware.

More recently, a deep reinforcement learning strategy to develop a feedback control law and generate locomotion to the biped robot Cassie [111] was demonstrated in [112]. The robot was able to walk robustly in simulation and the method has been recently tested on hardware [113]. Another work in deep reinforcement learning that was implemented on hardware is the work of [114]. Herein, a control policy was trained based on simulated data, allowing the robot to do self-righting and perform locomotion at high speeds.

The results in locomotion using deep reinforcement learning seem promising. However, the aim of this dissertation is to benefit from the strength of model-based approaches and the insight that they provide, and deal with the computationally expensive parts of the locomotion process (e.g., terrain evaluation) using machine learning-based techniques, and specifically convolutional neural networks (CNNs). In the next section, we provide a short overview regarding the use of CNNs in the context of robotic perception.

2.3 Convolutional Neural Networks and Robot Perception

Research in computer vision has experienced one of its most prolific decades thanks to the emergence of deep learning. One of the main turning points was the 2012 ImageNet Large Scale Visual Recognition Challenge [115], a competition designed to estimate the content of a very large hand-labeled dataset (ImageNet). The winner of the challenge used an algorithm based on a CNN, namely AlexNet [116], which outperformed the runner-up by nearly 10%. From that moment on, CNNs started to become the standard in machine vision image recognition tasks.

Two precursors of CNNs prior to AlexNet were the works of [117] and [118]. The CNN architecture proposed in [117] was one of the first to achieve high image classification accuracy in patterns with extreme variability, such as handwritten characters. On the other hand, in [118] the Rectifier Linear Unit (ReLu) activation function was introduced, and in [119] it was later proven that the use of this activation function can greatly speed up training.

There are several reasons why CNNs started to gain popularity. The "neurons" that comprise a CNN are named kernels. In essence, a kernel is a 2D convolutional filter applied to an input image. These kernels are trained to detect features using large amounts of labeled data. In the case of traditional computer vision, to extract features of an image these filters had to be designed considering the specific element that the user wanted to identify from the image. One of the major reasons why CNNs were not widely used was because of the required training times due to hardware limitations. However, the rapid evolution of computing and electronic devices reduced considerably the training times required for learning algorithms. As time went by, the time required to train a CNN started to become less compared to the time needed to design feature extraction filters. Furthermore, CNNs can be used in applications where traditional computer vision techniques already display high performance such as face detection [120] and enhance them, for example to do face recognition [121].

The field of computer vision intersects with robotics in several applications such as simultaneous localization and mapping (SLAM), object detection, grasping or scene representation. Because of this overlap, CNNs are nowadays used to perform these kinds of tasks on many robots. Some examples are: pedestrian detection for self-driving cars [122], object grasping [123] and visual odometry [124]. These use cases, however, are either designed for fixed-based robots or wheeled vehicles. There are only a few examples where CNNs are used to process visual [7, 125, 126] in the context of dynamic legged locomotion. In the next section, we explore the works related to locomotion strategies that make use of the information of the terrain, with and without vision, to understand the advantages and limitations of the state-of-the-art methods, and provide the context to support the use of a CNN for foothold selection.

2.4 Terrain Awareness in Legged Locomotion

Legged platforms are versatile over a wide variety of terrain and tasks. In parallel, sensor fusion techniques have evolved to overcome the harsh conditions typical of field operations on legged machines and provide effective pose and velocity estimates for planning, control, navigation and mapping.

Despite this progress, a real-time safe, computationally efficient way to use 3D visual feedback in dynamic legged locomotion is still challenging. The complexity lies on the high-density nature of visual information, which makes it hard to meet the fast response requirements for the control actions at dynamic locomotion regimes. In this section we provide some examples of strategies that deal with the problem of terrain awareness.

2.4.1 Trajectory Optimization and Footstep/Contact Planning

In Section 2.2.2 we discussed how trajectory optimization (TO) has improved dynamic locomotion. In this section, we focus mainly on TO strategies that consider knowledge about the morphology of the terrain, not necessarily coming from sensor data.

As mentioned in Section 2.2.2, trajectory optimization techniques usually break the problem into two parts: a footstep planner and a CoM planner. For the DRC, the WPI-CMU team implemented a trajectory optimization algorithm for the reference position of the CoM based also on the centroidal dynamics model using DDP, with predefined footstep locations. This method is reported in [74], which is similar to the work of [75], with the main difference that a footstep planner proposes a series of foot locations that the user can modify on-line. In the case of [127] an off-line footstep planner initially provides a finite number of feasible footsteps. Then, the given footstep sequence is used to generate a gait motion based on the methods presented in [102]. The method was demonstrated in simulation using the HRP4 humanoid robot walking on a set of flat surfaces at various heights. In [128], the authors devised a trajectory optimization method that constructed a cost map using an on-board RGB-D sensor, to evaluate the terrain in search of safe footholds to avoid rough areas and collisions. The problem was decomposed into two parts: a body action plan and a footstep plan. The body action plan evaluated the difficulty to traverse the terrain and the footstep plan decided where to place the feet based on the cost map. The work was extended in [55] by including a cart-table model to generate the trajectory of the CoM for the body action plan. In [63], a motion planning algorithm was showcased. The algorithm computed gait pattern, contact sequences and CoM trajectory as an outcome of a mixed-integer convex program (MICP) on several non-planar convex surfaces. The method devised in [56] (mentioned in Section 2.2.2) was able to optimize gait, CoM trajectory and contacts, considering non-flat terrain based on a simplified centroidal dynamics model using an off-the-shelf NLP solver.

In [57], an approach to decompose the problem into two steps was proposed, namely: *a*) planning a feasible root guide path and *b*) generate the contacts along the path. The feasible root guide path was a path planned in low-dimensional space based only on the reachability of the objects that may act as support for the feet and collision avoidance. Then, once the root guide path was decided, they computed the contacts based on their dynamic feasibility.

Similarly, in [58] the authors propose an approach that plans a trajectory for the CoM and then obtains a series of contacts along the CoM trajectory without specifying a specific goal position, but rather a bound around this goal position. Then a contact planner was in charge of finding feasible sets of contacts in order to reach the surrounding area of the goal. Contact locations were initially found using anytime non-parametric A* (ANA*), and then to evaluate the dynamic feasibility of the contacts, they used two neural networks that learned both the feasibility and the evolution of the dynamics between contact transitions based on a MICP-based foothold planner.

Trajectory optimization approaches have provided solutions to deal with complex scenarios using information of the morphology of the terrain. However, in the works here cited, either the trajectories are computed only once before the motion is executed or the terrain is assumed to be known and there are no experiments with vision sensors during the execution of the motion. These issues limit their application to simulation or non-dynamic environments without presence of disturbances.

2.4.2 Terrain Awareness with Vision

In this section we present some of the locomotion strategies that use visual feedback to traverse difficult terrain. Kolter et al. [3] provided one of the first applications of terrain awareness to enhance the traversing capabilities of a quadruped robot. To do so, collision probability maps and heightmaps collected a priori are used to train a Hierarchical Apprenticeship Learning algorithm to select the best footholds in accordance to an expert user.

A similar approach was taken in [4] where, in contrast to [3], visual feedback was discretized using templates, i.e., portions of terrain in the vicinity of a foothold. With a learning regression method based on expert user selection, a target foothold is associated to each template. The authors have incorporated the classification algorithm into a locomotion planner and demonstrated its validity on the robot LittleDog, traversing highly unstructured terrains.

Both approaches have proved to be powerful, but they rely on external motion capture systems for the position and maps were provided a priori, reducing their field of application to controlled and calibrated environments. In contrast, in [5] they used an on-board laser scanner to collect an elevation map of the terrain. Their method searches for useful clues related to the foothold placement, and selects the ones with minimal slippage. The optimal footholds are learned in an unsupervised fashion, inside a simulated environment.

Despite their ability to perform locomotion tasks with on-board sensors only, most of the vision-based foothold selection strategies involve slow motions, mainly to provide enough time to complete the most costly operations such as image processing and optimization. An exception was shown in [129], where the acquisition of swept-sphere-volumes allowed the biped robot Lola to avoid obstacles while moving, with no prior information about the environment. Nevertheless, this strategy was mainly demonstrated for single obstacle avoidance and self collision, and not for rough terrain.

The work of [2] is similar to the template-based foothold correction of [4], but it differs due to its implementation in a fully reactive fashion. Heightmaps around the nominal footholds are evaluated to generate continuous motion corrections for the RCF [52]. The corrections are learned from expert demonstration using a Logistic Regression classifier.

More recently, a perception-based statically stable motion planner for the quadruped robot ANYmal [12] was presented [6]. For each footstep, the algorithm generates a foothold (upon rejection of unsafe and kinematically unfeasible solutions), a collision free foot trajectory, and a body pose. However this approach has been demonstrated with (quasi-) static gaits and the foothold corrections are computed only at the moment of lift-off of the leg.

Most of the previously mentioned terrain awareness strategies that used vision sensors rely on performing locomotion at low speeds. The goal of this dissertation is to be able to break this limitation and propose a robust and reliable locomotion strategy that considers terrain while performing dynamic gaits.

2.5 Summary and Discussion

In this chapter, we have presented relevant works regarding the two main components that we want to bring together: dynamic locomotion and terrain awareness using vision. Dynamic locomotion has come a long way since the works of Raibert [36]. There is room for improvement and new interesting research directions. One of the main topics that are being tackled nowadays is the increase in computation efficiency in trajectory optimization algorithms. On one hand, one of the limitations that prevent approaches such as [62, 64, 56] is that they still take very long time to output reference trajectories, which means that they still cannot be used in an MPC-fashion while performing highly dynamic motions. MPC has been achieved thanks to algorithmic tools such as automatic differentiation [59], as well as the decomposition of the problem into CoM trajectory generation and footstep selection [39]. However, most of the MPC-based locomotion strategies do not consider information about the terrain coming from the vision sensors. It is safe to say though, that dynamic locomotion by itself has reached a state where the "deltas" of improvement are more subtle and refined with respect to the initial breakthroughs.

Conversely, terrain awareness is now one important point of discussion and improvement in the field of legged robotics. The cost of building a map and evaluating the terrain makes the problem difficult to address using state-of-the-art trajectory optimization approaches. In general, plans are computed prior to the motion and then executed [63, 55], or tested in simulation [56]. The approaches that are able to evaluate the terrain as the robot traverses have been mostly implemented in the case of gaits that are performed at low-speed ranges [6, 125].

This dissertation aims to fill the gap between dynamic locomotion and terrain awareness based on vision using on-board sensing and computing. Our methods stand on top of the achievements on dynamic locomotion in the past years and we devise methods that make use of novel control methodologies and learning algorithms such as MPC and CNNs to devise locomotion strategies that evaluate terrain morphology in the time frame that dynamic locomotion demands.

Chapter 3

System Overview

The HyQ-series of hydraulically actuated quadruped robots is nearing its 10th anniversary since HyQ's debut in 2011. Since then, newer versions have improved aspects of the original design such as aesthetics, joint range of motion, compactness and component reliability. The latest iteration of the series, HyQReal, takes all the lessons learned from its predecessors and focuses on improving power autonomy to be brought outside of the lab.

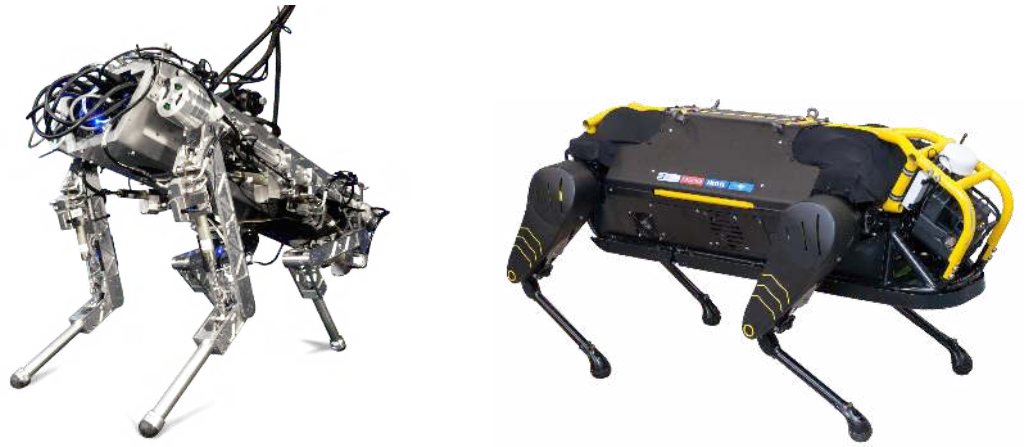
The methods devised throughout this dissertation use both the precursor (HyQ) and the avant-garde (HyQReal). In this chapter, we summarize their main features related to actuation, sensing and locomotion, to provide a better overview of the robots' capabilities and limitations.

3.1 Robots Overview

The HyQ-series is a family of quadruped robots developed by the Dynamic Legged Systems lab (DLS) at the Istituto Italiano di Tecnologia (IIT)[1]. In this dissertation we perform experiments and simulations on HyQ and HyQReal, shown in Figure 3.1. The four legs of each robot have three degrees of freedom (DOFs): a hip joint for abduction and adduction (HAA), a hip joint for flexion and extension (HFE), and a knee joint for flexion and extension (KFE). Schematic drawings of each of these robots and their leg/joint descriptions are shown in Figure 3.2.

HyQ weighs close to 100 kg and it was built mainly to run on external hydraulics and electric power. As mentioned previously, one of the main features of HyQReal is its ability to run with on-board hydraulics and electric power. To this end, the robot is equipped with two

[1] Website: https://dls.iit.it/



(a) HyQ **(b)** HyQReal

Figure 3.1: Hydraulically actuated quadruped robots HyQ and HyQReal.

hydraulic pressure units (HPUs) and a battery. The total weight of the robot including HPUs and battery totals 130 kg.

Both of these robots have the same total number of DOFs, however, there are key differences regarding both the mechanics and the software of the robots. These differences are addressed throughout this chapter and the methods devised in this dissertation take them into account. The rest of the chapter is organized as follows: Section 3.2 describes the actuators and sensors equipped both in HyQ and HyQReal; Section 3.3 gives an overview of the software/hardware architecture developed to control these robots; and Section 3.4 outlines the main locomotion capabilities of the HyQ-series robots.

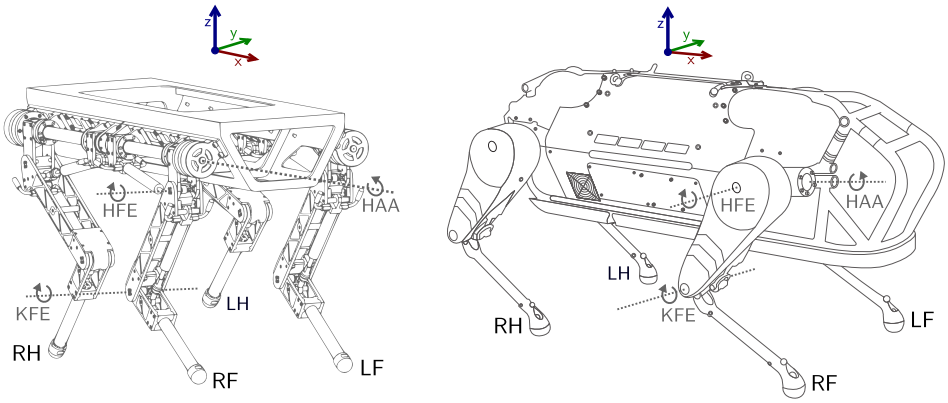


Figure 3.2: Leg, joint and frame descriptions of HyQ (left) and HyQReal (right). The legs are identified as left-front (LF), right-front (RF), left-hind (LH) and right-hind (RH). The joints are indicated on the RF leg of each robot and are identified as hip abduction/adduction (HAA), hip flexion/extension (HFE) and knee flexion/extension (KFE).

3.2 Actuators and Sensors

HyQ and HyQReal both use hydraulic cylinders and hydraulic rotary motors. However, one crucial difference between HyQ and HyQReal is related to the specific actuators and proprioceptive sensors used. On one hand, HyQ uses several off-the-shelf components including hydraulic cylinders and valves, force sensors, and relative encoders. The advantage of using off-the-shelf components is that they are generally cheaper and have better documentation, the main drawback is that these separate components require several mechanical, electrical and software interfaces, which affects the control and design of the robot. On the other hand, HyQReal uses the Integrated Smart Actuator (ISA) and smart manifolds [16] developed by IIT's industrial partner Moog, in collaboration with the DLS lab. The ISA and the smart manifolds are hydraulic actuation systems that contain all the previously mentioned sensors, hydraulic components and the necessary electronics to do hydraulic low-level control (e.g., force/torque control), all in one compact enclosing structure. This provides a much better form factor for mechanical design and facilitates the control of each of the joints.

Because of their key differences, we proceed to explain separately the actuation and proprioceptive sensing capabilities of HyQ and HyQReal. The vision sensors, however, are the same for both HyQ and HyQReal and they are explained in one single section.

3.2.1 HyQ

Each of the legs of HyQ[2] [53] has three DOFs which include the HAA, HFE and KFE joints. The HAA joints are actuated by hydraulic rotary motors, while the HFE and the KFE joints are actuated by hydraulic linear cylinders. The four HAA joints are equipped with a torque sensor, while the remaining eight HFE and KFE torques are computed measuring the force applied by the hydraulic cylinder using load cells. All of the 12 joints are equipped with relative and absolute encoders to measure position. HyQ is also equipped with a tactical-grade IMU, used by the state estimator [70] to provide the pose and velocity estimates of the body with respect to the world.

3.2.2 HyQReal

One of the main upgrades of HyQReal [132] with respect to its predecessors is the implementation of novel actuation devices [16]. Each of the HAA and HFE joints is actuated by a hydraulic rotary motor attached to what is called a *smart manifold*. The smart manifold is an actuation device that includes an additively manufactured (AM) body made with titanium, a high-performance servo valve and all the electronics and sensors needed to perform

[2]For an in depth description of the mechanical design of HyQ we refer the reader to [130] and to [131] for reference regarding the equipped sensors.

high-bandwidth torque control [16]. A custom-made hydraulic rotary actuator, a magnetic rotary encoder and a torque sensor are attached to the smart manifold to perform torque and position control. Each of the four KFE joints is driven by the Integrated Smart Actuator (ISA). The ISA includes all of the features of the smart manifold, but instead of a rotary actuator, it contains a hydraulic cylinder that is fully integrated into the actuation unit. This results in an even more compact device with respect to the smart manifold. Additionally, the ISA also provides measurements of the valve spool position, which can be used as an extra degree of freedom for control. Figure 3.3 shows both of these actuators.

The use of these types of actuators in a legged robot has several advantages: first, it reduces the overall complexity of the machine, since the various actuator components are combined into one device. Sensor wires are routed inside the AM body and several components are merged into the same electronic board (e.g., microcontroller, valve amplifier, temperature sensor). Fewer and shorter wires result in higher reliability and less signal noise; second, it reduces the total robot weight, and increases its ruggedness.

3.2.3 Vision and Exteroception

Throughout the years, various configurations of exteroceptive sensors have been used on HyQ [131] and HyQReal. In this thesis we mainly use two types of exteroceptive sensors: *a*) a depth sensor or RGB-D sensor and, *b*) a LiDAR sensor. The depth sensor, which can be an Asus Xtion or an Intel RealSense, is used for visual odometry (VO) and mapping. The LiDAR is mainly used to perform low-frequency pose corrections using a scan matching based on the iterative closest point (ICP) algorithm adapted in [70].

(a) Smart manifold **(b)** ISA

Figure 3.3: Hydraulic actuators mounted on HyQReal developed by Moog and the DLS lab. The smart manifold with the attached custom-made hydraulic motor, torque sensor and rotary encoder (a) and the ISA (b) [133].

3.3 Software/Hardware Architecture

HyQ and HyQReal are equipped with two on-board PCs: a *Control-PC* that runs a real-time Linux kernel and a *Vision-PC* running a regular kernel. The two computers are synchronized by means of an NTP server. The Control-PC executes the robot control commands (i.e., the *Supervisor task*) in a real-time environment, as well as the Extended Kalman Filter state estimator in a non-real-time thread. The Vision-PC collects the exteroceptive inputs, computes the VO and the ICP-based scan matching, and sends to the Control-PC an elevation map of the surrounding area of the robot. The elevation map is a crucial component of the strategies presented in Chapters 4 and 5. In case of failure of the Vision-PC, the controller can operate blindly with a smooth but drifting pose estimate. Both HyQ and HyQReal receive user commands from an off-board computer, the *Operator-PC*, connected via wifi or Ethernet.

There is a key difference regarding the low-level control and the hardware/software architecture. In HyQ, the final low-level control signals that are sent to the actuators (valve commands) are computed inside the Control-PC. In the case of HyQReal, since the ISA and the smart manifolds include an electronic board capable of doing the hydraulic low-level control, only the torque references are computed inside the Control-PC and they are sent to the electronic boards of the smart actuators, which compute the low-level hydraulic control commands [16]. Figures 3.4 and 3.5 show the hardware/software architectures of HyQ and HyQReal, respectively.

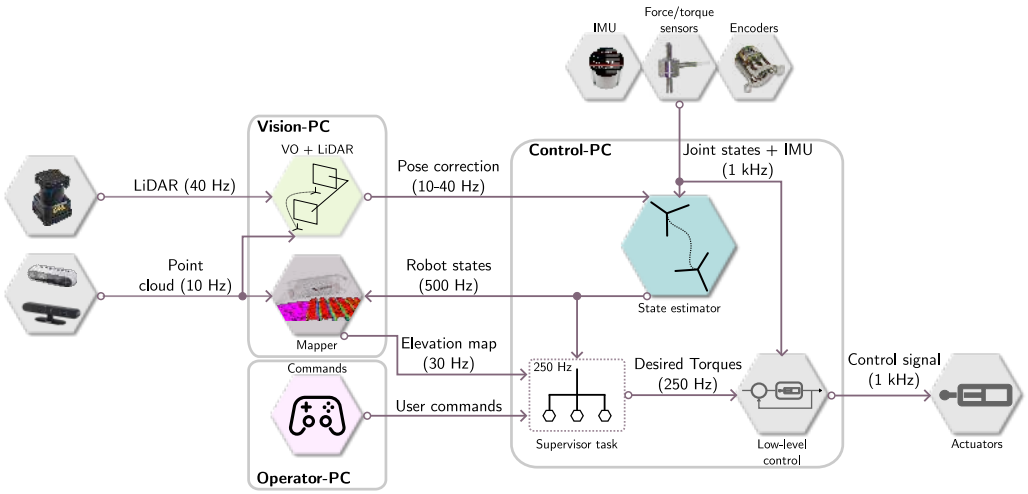


Figure 3.4: Hardware/software architecture of HyQ. The state estimator runs on the Control-PC and receives the signals from the proprioceptive sensors and low-frequency pose corrections from the visual odometry (VO) module computed in the Vision-PC. The elevation map used in Chapters 4 and 5 is also computed in the Vision-PC using the point cloud delivered from the depth sensor and the mapper module. The map and the robot states are sent to the supervisor task, which runs in a real-time thread and sends the desired torques to the low-level control module. The low-level control module sends the valve commands to every actuator.

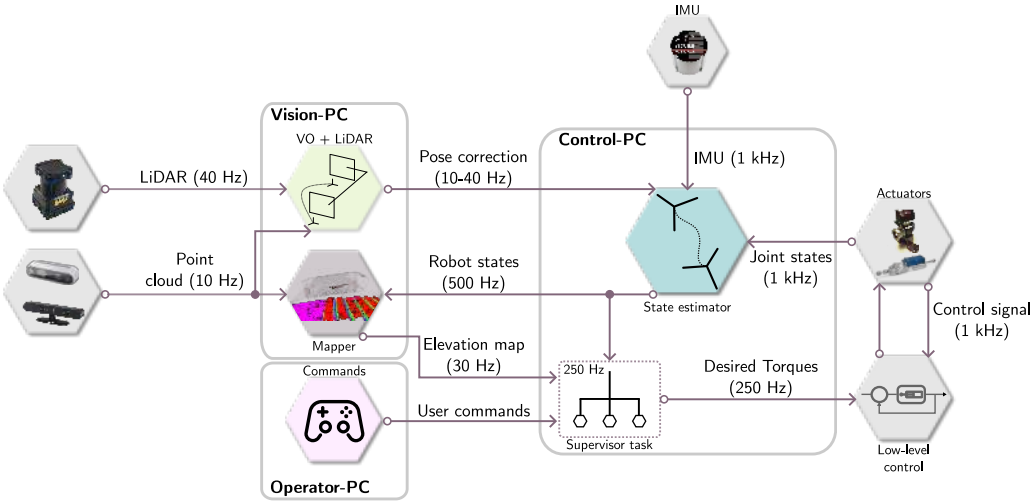


Figure 3.5: Hardware/software architecture of HyQReal. It retains a similar structure to that of HyQ, however, the low-level control and the collection of the proprioceptive signals are outsourced to the electronic boards mounted inside the ISA and the smart manifolds.

3.4 Locomotion Capabilities of the HyQ-series Robots

Several locomotion strategies have been implemented on the HyQ-series robots to perform different types of gaits including crawl [134], bound [135] and trot [52]. In this dissertation we mainly focus on trotting gaits and we specifically use the locomotion framework developed in [52], the Reactive Controller Framework (RCF), as a foundation for the building blocks that we devise in the subsequent chapters. However, the methods developed here can also be used and extended to other dynamic and static gaits. Below, a summary of the main components of the RCF is given.

3.4.1 Reactive Controller Framework

The RCF is a modular locomotion framework that is able to produce omni-directional periodic gaits and is comprised of several modules that allow the robot to perform robust dynamic locomotion on rough terrain. This framework was initially devised using proprioceptive sensors only, and in this dissertation we have extended its use to consider exteroceptive inputs as well. It consists of two main parts: a *motion generation* block and a *motion control* block.

The motion generation block is in charge of assigning the trajectories of the feet with respect to what is called the *horizontal frame*. The horizontal frame is a reference frame whose origin coincides with the body frame and shares the same yaw orientation, but its xy plane is always perpendicular to the direction of the gravity vector. Choosing such frame makes the generation of the feet independent from the trunk attitude, which is key when dealing with non-flat terrain. It helps to maintain the foot contacts even when the trunk does not

follow the desired attitude. A schematic drawing showing the horizontal frame is shown in Figure 3.6. The leg trajectories and gait pattern are assigned using a set of central pattern generator (CPG)-inspired coupled nonlinear oscillators with elliptical shapes, described in Cartesian coordinates. The motion control block provides corrective actions to track the desired trunk motion. Such actions can be defined at the kinematic level (e.g., push recovery) or at torque level (e.g., whole-body control).

In addition to the motion control and motion generation blocks, the RCF has a reactive layer that contains a number of modules that increase the reliability of the robot while crossing difficult terrain; some examples are push recovery, foot collision detection, terrain adjustment and shin collision detection. These modules send reactive components to adjust the trajectory and joint torques to the motion generation and motion control blocks. A more detailed description of this framework can be found in [52].

The Vision-based Foothold Adaptation (VFA) devised in Chapter 4 computes a trajectory adaptation that is sent to the motion generation block to adjust the landing position of the feet. The model predictive control (MPC)-based trunk controller with terrain awareness stabilizes the trunk and replaces the trunk controller component of the motion control block of the RCF. We are able to implement these novel approaches thanks to the modularity of the RCF.

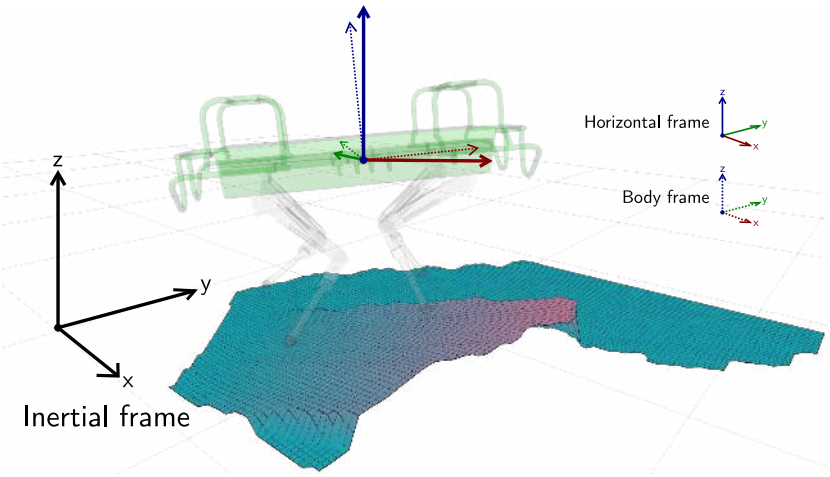


Figure 3.6: Description of the horizontal frame. The horizontal frame is attached to the body frame and shares the same yaw orientation as the body frame. The xy plane of the horizontal frame is always perpendicular to the gravity vector, i.e., roll and pitch angles are always equal to 0.

Chapter 4

Vision-based Foothold Adaptation for Dynamic Locomotion

Leveraging terrain information to improve dynamic locomotion on rough terrain scenarios is a complicated task. Building a map and computing kinematic and dynamically feasible contact sequences is computationally expensive. This cost renders the problem highly time consuming, especially in the case of complex legged systems. This is time that a robot cannot afford during dynamic locomotion.

This chapter addresses the use of visual information during dynamic locomotion for legged robots on a reactive level. Our goal is to endow the robot with the ability to overcome complex scenarios where disturbances and rough terrain may lead to unsafe footholds. We devise a strategy that adapts foothold locations in a fast and continuous fashion using only on-board sensing and computing.

4.1 Problem Statement

Consider the case depicted in Figure 4.1. A person is running through rough terrain and plans a series of footstep locations to avoid falling. Suddenly, the person is pushed from the back while one foot is in the air. The originally planned foothold cannot be reached due to the disturbance on the body, and the change in trajectory of the foot now leads to an undesired location. The runner only has the remaining time that the foot is in the air to *adapt* the landing position. In that split second, the runner is able to find a better foothold and adjust the trajectory of the leg to reach the new desired position[1].

[1] A similar situation can be caused by a slipping stance foot.

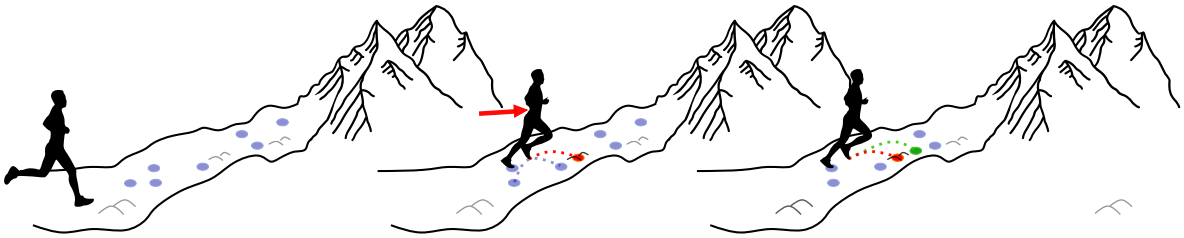


Figure 4.1: Example of a vision-based reaction. A runner is pushed and needs to adjust the landing location of the feet to prevent falling.

We would like to reproduce this fast decision-making behavior. We acknowledge that in many cases carefully planned contact sequences are required to overcome difficult terrain. However, when dealing with disturbances, changing environments and dynamic locomotion, the plan often needs to be recomputed. The robot might not have the timespan to compute a new contact sequence according to these changes.

We propose a way to reduce the risk of failure due to foothold placement in a short timescale. This timescale is defined by the total swing time of a leg. To achieve this, we enhance the haptics-based reactive layer provided by the Reactive Controller Framework (RCF) [52] (see Section 3.4.1) with foothold adaptations triggered by visual information.

In this chapter we explain how we deal with the computationally demanding processes related to visual information and safe foothold selection. We call our approach Vision-based Foothold Adaptation (VFA)[2]. The rationale behind the VFA stems from the idea of using visual information about the terrain during dynamic legged locomotion in a fast and continuous fashion. We remove the load related to the computation of safe foothold locations from the robot and outsource it to a self-supervised, off-line learning pipeline. The outcome of this process is a function approximation of an evaluation for foothold selection based on terrain morphology and foot/leg collision avoidance. The approximated function is able to compute safe foothold locations up to 350 times faster than the full-blown evaluation of the terrain. This computational gain allows the robot to compute safe landing locations continuously along the swing phase trajectory of each leg.

The rest of this chapter is structured as follows: Section 4.2 explains the machine learning pipeline to train the function approximation of the terrain evaluation; Section 4.3 details how the VFA is integrated into the locomotion control architecture of the robot; Section 4.4 shows the prediction results of the trained foothold classifier, as well as the outcome of the simulation and experimental tests of the strategy; finally Section 4.5 provides the summary and the discussion regarding the contents of this chapter.

[2]This strategy could also be referred to as Map-based Foothold Adaptation.

4.2 Self-Supervised Learning Pipeline

Our learning pipeline is inspired by [2]. Herein, image samples of different obstacles were collected using an RGB-D sensor. An expert user selected the most suitable foothold location considering terrain roughness and possible leg collisions from each image. The foothold locations and images were then used to train a logistic regression-based classifier. The trained classifier was able to continuously compute safe foothold locations allowing the legs to adapt their trajectory to avoid obstacles. A diagram describing this learning pipeline is shown in Figure 4.2.

We improved the learning pipeline of [2] focusing on three main aspects:

1. *Map generation and collection.* Collecting map samples from the vision sensors in [2] was highly time consuming. We improved this by generating artificial maps that approximated obstacles. This process was automated to account for different shapes and orientations. Furthermore, we built testing scenarios in simulation and collected additional maps from these scenarios.

2. *Autonomy of training.* We replaced the human expert foothold selection with a self-supervised evaluation based on the geometry of the terrain and the robot kinematics. This allowed us to increase the size of the training set, to a potentially unlimited number of samples. With this new evaluation we could also increase the number of possible landing positions for each map from 9 in [2] to 225.

3. *Learning model improvement.* We improved the learning algorithm by replacing the logistic classifier with a convolutional neural network (CNN), whose effectiveness for image classification has been proven [136, 116]. This allowed us to evaluate more difficult and diverse type of obstacles with respect to [2]. The improvement in accuracy enabled us to craft a carefully balanced architecture of the CNN with a low-dimensional parameterization.

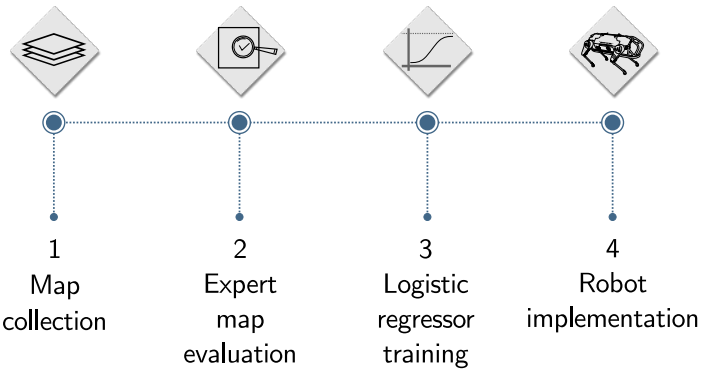


Figure 4.2: Description of the learning pipeline in [2].

Figure 4.3 shows our improved self-supervised, off-line learning pipeline. We start by generating artificial images that represent obstacles such as bars, steps and pallets of different sizes and at various orientations (see Section 4.2.1). We then evaluate these image representations to select the best landing position of a specific foot according to a number of criteria (see Section 4.2.2). We improve the robustness against sensor noise by adding maps corrupted by zero-mean Gaussian noise. Then, a first version of the CNN is trained using the set of evaluated maps. This provides a function approximation of the full-blown map evaluation. This function approximation is first implemented and tested in a simulated scenario. We then collect the maps from the simulator, and repeat the evaluation and noise addition performed for the artificial maps. A CNN with the same architecture as the first one is trained using both of the training sets corresponding to the artificially generated maps and the maps collected from simulation. The outcome of this training is the final function approximation that we implement on the real platform.

In the next section we define the image representations of the terrain used for training (namely *heightmaps*), and explain how they are artificially generated and collected from simulation.

4.2.1 Heightmap Definition, Artificial Generation and Collection from Simulation

Heightmap definition. A *foothold heightmap* (or simply *heightmap*) is a two-dimensional discrete representation of the terrain where each pixel describes the height of a specific area. The heightmap is defined such that its center corresponds to the *nominal foothold* of a specific leg. The *nominal foothold* is the predicted landing position of a foot during swing phase considering the leg trajectory and the velocity of the base (see Section 4.3.1).

The heightmap is parameterized by size and resolution. These two parameters are depicted in Figure 4.4. Both parameters are the result of a trade-off between computational expense and task requirements. We want to avoid processing large amounts of data, while retaining a

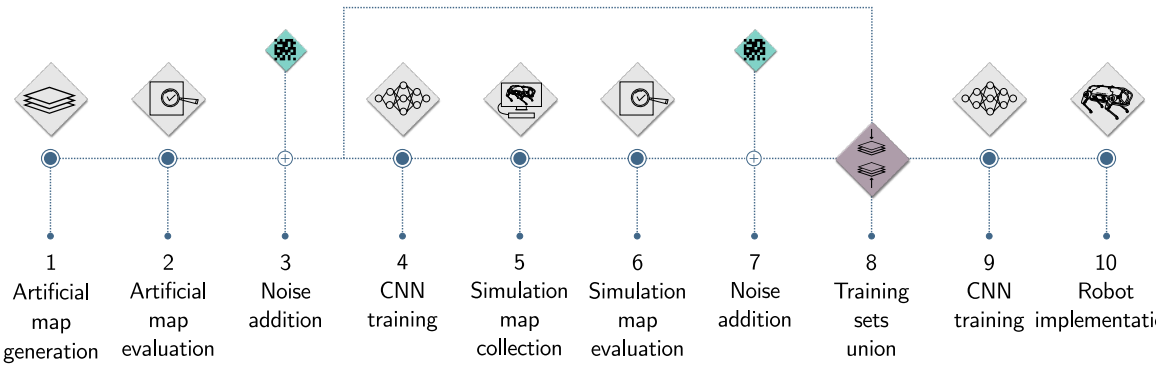


Figure 4.3: Description of learning pipeline.

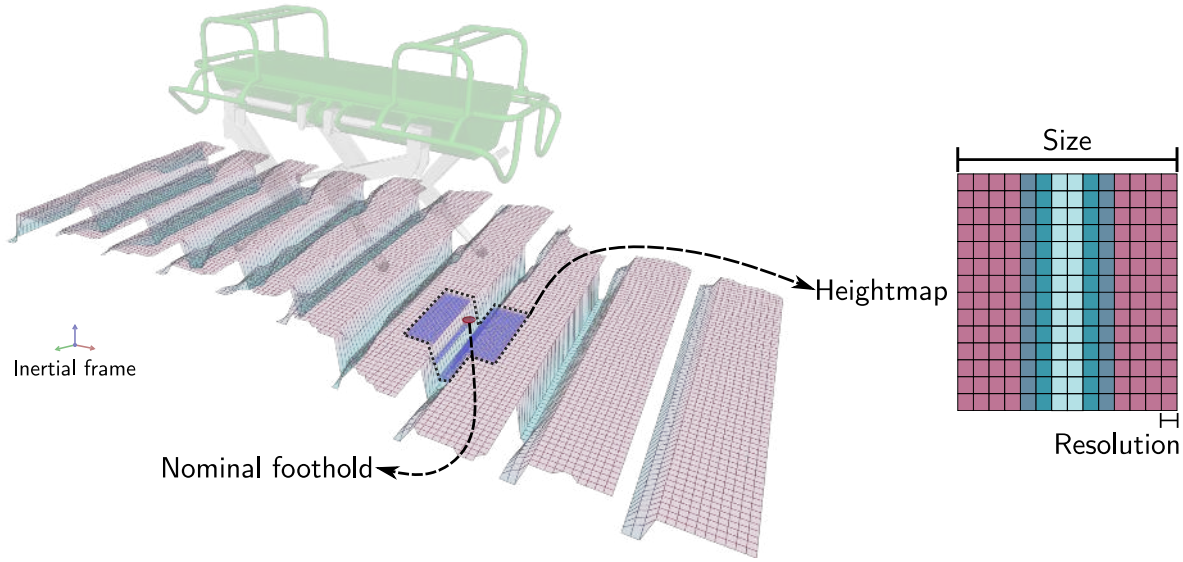


Figure 4.4: Example of an acquired heightmap from simulation.

level of detail that is meaningful to the task. A detailed discussion on appropriate parameter selection can be found in [2]. Each pixel in the image corresponds to a possible foothold. We opt for a heightmap size of $30 \times 30\,\text{cm}$ with a resolution of $4\,\text{cm}^2$ per pixel (heightmap of 15×15 pixels), chosen according to the foot radius of HyQ ($2\,\text{cm}$).

Artificial heightmap generation. An image can be represented as a matrix, and in the case of a heightmap each element of the matrix corresponds to the average height of an area of $4\,\text{cm}^2$ (see Figure 4.4). We use this representation of the heightmaps to generate artificial examples of obstacles. We reproduce bars, gaps, steps and pallets of different sizes and at various orientations. Examples of artificially generated heightmaps can be seen in Figure 4.5.

We use the artificially generated heightmaps to build a richer training set with significantly more features with respect to [2], without having to design specific simulation scenarios or collect them using the real sensors. Quantitatively, the number of examples was increased from 3300 in [2] to 35688[3] combining artificially generated maps and the ones collected from simulation.

Heightmap collection from simulation. The examples from simulation came from our software framework, that uses the Gazebo [137] simulator to reproduce the dynamics and the sensors. After training a first version of the CNN using the artificially generated data (step 4 in Figure 4.3), the robot was commanded to cross a series of wooden beams and the visual

[3]This number of training examples corresponds to the results presented in this chapter. For the simulation and experiments in Chapter 5 a larger training set was used to train an improved version of the CNN.

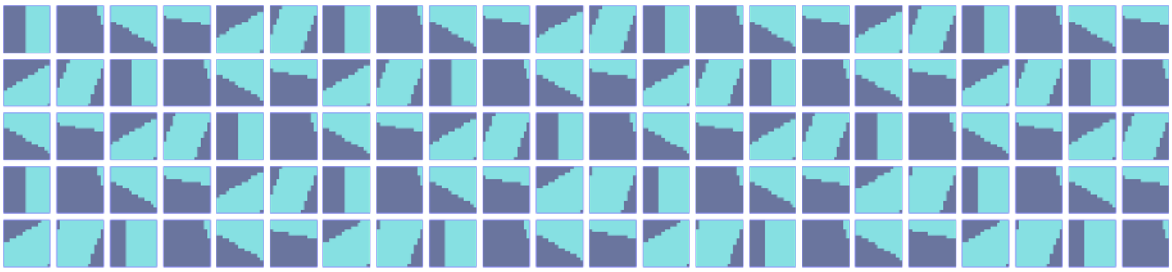


Figure 4.5: Examples of artificially generated heightmaps. The light color represents higher areas in the map.

data corresponding to the heightmaps of each of the feet were recorded for training. This scenario can be seen in Figure 4.4.

Remark 1 (Real sensor data). In the case of dynamic locomotion, state estimation, mapping and joint tracking are subject to uncertainty. We will explain where this uncertainty is originated and how we cope with it during the heightmap evaluation in Section 4.2.2. This uncertainty may lead to errors in the pose estimations, causing bad positioning of the map with respect to the world. Furthermore, joint tracking and base velocity estimation errors, lead to wrong predictions of the next nominal foothold. Obtaining data from the sensors during dynamic locomotion is complex because there is no "ground truth" data regarding the position of the robot, the map and the feet trajectory tracking. However, it is important to point out that the CNN-based classifier is able to provide safe foothold locations in experiments only based on simulation and artificially generated data. This ability to extrapolate from training data to reality strengthens our approach.

In the following section, we describe how the maps are evaluated to create the sets used to train the CNN described in Section 4.2.3.

4.2.2 Heightmap Evaluation

We evaluate the set of 35688 heightmaps to find safe footholds with respect to the terrain morphology and the robot kinematics. Given that the radius of the feet is 2 cm, we consider each element of the grid ($4\,\mathrm{cm}^2$) as a potential foothold. This results in an output space of 225 possible landing positions. In particular, we evaluate each heightmap considering single leg swing motions with respect to six criteria:

1. uncertainty;
2. terrain roughness;
3. kinematic limits;

4. foot frontal collision;

5. leg collision;

6. distance to the nominal foothold.

Figure 4.7 shows an example of a fully evaluated heightmap and Appendix A shows examples of evaluations with respect to specific criteria. We proceed to explain each one of the criteria.

Uncertainty margin. Barasuol et al. [2] identified three main sources of uncertainty that affected the reactive foothold placement therein implemented. The first source of uncertainty is *swing leg trajectory tracking errors*. This is mainly related to the low-level control of the joints. The second source of uncertainty is *map-drift*. This drift in the map is mainly caused by errors in the robot's pose provided by the state estimator. Phenomena such as impacts, link flexibility [138] and inertial measurement unit (IMU) drift, affect negatively the pose estimation. Finally, a third source of uncertainty comes from the *foothold prediction errors*. These errors are also influenced by the inaccuracies of the state estimator, however, they also depend on the chosen prediction model of the next landing position (see Section 4.3.1).

We conducted a series of experiments on flat terrain to identify the total error due to the three aforementioned factors. The experiment consisted on commanding the robot to trot approximately for 2 m, with a forward velocity $\mathbf{V}_f$ with a magnitude of 0.3 m/s along the x direction of the horizontal frame, a duty factor $D_f = 0.65$ and a step frequency $f_s = 1.4$ Hz. We recorded the data from these experiments and measured the error caused by inaccuracies in our foothold prediction model (see Section 4.3.1). We also measured the error caused by state estimator drift during swing. We compared the longitudinal position of the robot computed by our on-board state estimator [70] against the position provided by a Vicon motion capture system. Areas of the map that are no longer in the field of view of the vision sensors are affected by this drift. The results of this measurements are reported in Figure 4.6. We identified an uncertainty of 3 cm around the nominal foothold, related to errors in the foothold prediction. The short term map drift added an error of 2.5 cm after traversing the distance corresponding to a swing phase. To account for the total of errors we define an *uncertainty margin*. This margin defines the size of the neighborhood considered when evaluating terrain roughness. We set an uncertainty margin of 6 cm (three cells) to also consider errors due to leg trajectory tracking.

Terrain roughness. For each possible foothold (pixel), we compute the mean and the standard deviation of the slope relative to its neighborhood. The size of this neighborhood is defined by the uncertainty margin. We define a specific threshold for the sum of the standard deviation and the mean of the slope, according to the foot radius. The footholds with values

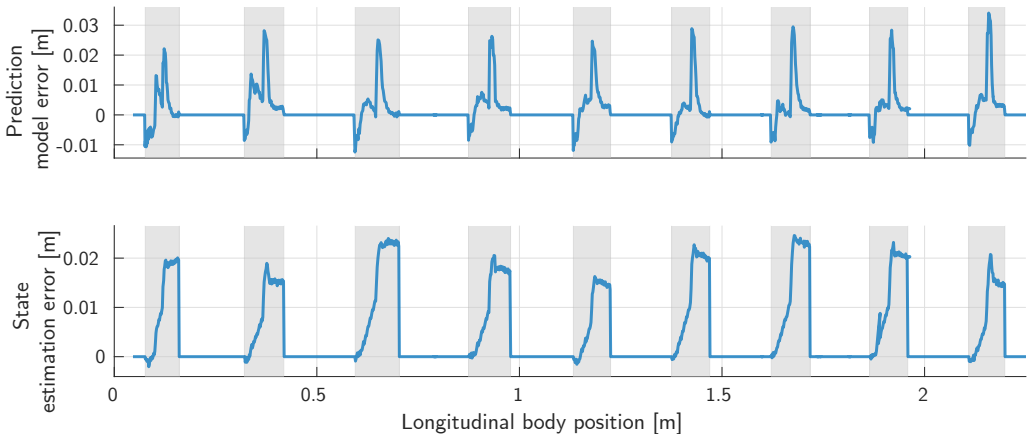


Figure 4.6: Error plots for the experiments on flat terrain to evaluate uncertainty for the left-front (LF) leg. Gray areas represent the moments when foothold predictions are performed (i.e., during swing phase).

above this threshold are discarded. In Figure 4.7 the footholds marked with a light-blue ✖ symbol are discarded due to terrain roughness.

Kinematic limits. Footholds are discarded if they are located outside of the workspace of the leg. We determine if a foothold is reachable using the leg's inverse kinematics.

Foot frontal collision. We evaluate potential frontal collisions along the predefined trajectory from the lift-off position. We consider a distance threshold for possible collisions along the trajectory considering the radius of the feet. In Figure 4.7 the discarded footholds due to foot collisions are marked with a black ✖ symbol.

Leg collision. Similarly to frontal collisions, we evaluate the intersection between the terrain and both of the leg limbs throughout the whole step cycle (i.e., stance and swing phases). We mark the discarded footholds due to leg collisions with a red ✖ symbol in Figure 4.7.

Distance to nominal foothold. In the case that multiple footholds satisfy all of the previous criteria, we define the *optimal foothold* as the one that is closest to the nominal foothold. We do this to minimize the deviation from the original trajectory.

Let $\mathbf{H} \in \mathbb{R}^{15\times15}$ be an input heightmap. We denote[4] $g_i(\mathbf{H}) : \mathbb{R}^{15\times15} \rightarrow \mathbb{Z}_2^{15\times15}$ as a mapping that takes a heightmap $\mathbf{H}$ as input and outputs a matrix of binary values indicating the

[4]Herein, $\mathbb{Z}_2 = \{0, 1\}$, where 0 corresponds to an unsafe foothold and 1 to a feasible foothold, and subindex i in $g_i(\mathbf{H})$ stands for a possible evaluation criteria.

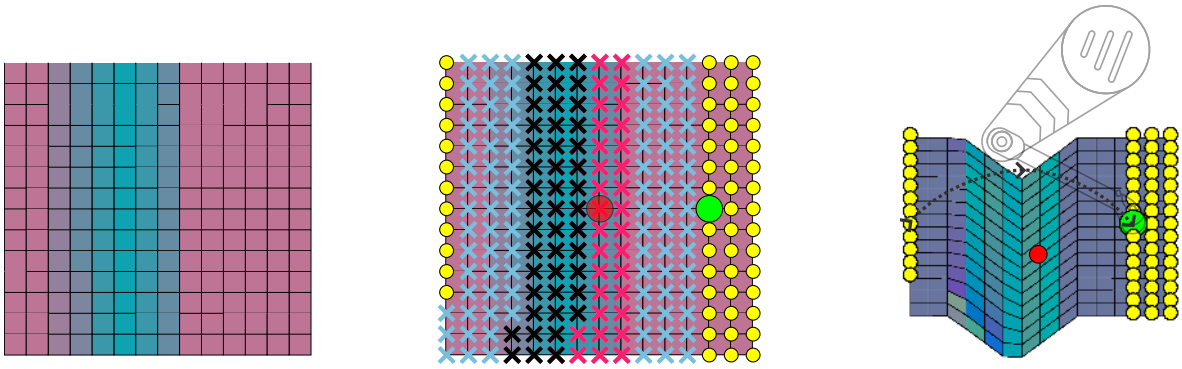


Figure 4.7: Example of a heightmap evaluation. The map prior to evaluation is seen on the left. In the center the evaluated map is shown. Yellow discs indicate feasible footholds, the red disc indicates the nominal foothold and the green disc indicates the optimal foothold. Crosses indicate discarded footholds according to terrain roughness and uncertainty margin (light-blue); shin collision and kinmeatic limits (red); and foot frontal collision (black). On the right, the leg trajectory is shown along with the feasible, the nominal and the optimal footholds.

elements in $\mathbf{H}$ that correspond to safe footholds according to one of the previous criteria. We define four mappings $g_t(\mathbf{H})$, $g_k(\mathbf{H})$, $g_f(\mathbf{H})$ and $g_l(\mathbf{H})$, corresponding to: terrain roughness and uncertainty margin; kinematics; frontal collision; and leg collision, respectively. A *feasible foothold* is defined as a foothold that is deemed as safe according to all four mappings. Furthermore, we define $g(\mathbf{H})$ as

$$g(\mathbf{H}) = g_k(\mathbf{H}) \wedge g_t(\mathbf{H}) \wedge g_f(\mathbf{H}) \wedge g_l(\mathbf{H}), \tag{4.1}$$

where the operator $\wedge$ represents a coefficient-wise logical *AND*. The elements that are equal to 1 of the matrix coming out of $g(\mathbf{H})$ correspond to feasible footholds. The mapping $h(\mathbf{H}) : \mathbb{R}^{15\times15} \rightarrow \{0,1\}^{225}$, computes the feasible footholds according to $g(\mathbf{H})$ and outputs a "one-hot vector"[5] that represents the optimal landing point, as the one with the smallest Euclidean distance to the nominal foothold. In Figure 4.7 feasible footholds are marked with yellow discs and the optimal foothold is marked with a larger green disc.

Remark 2 (Definition of the uncertainty margin). The selection of the size of the uncertainty margin is the result of a trade-off between safety and number of available feasible footholds. We would like to consider an uncertainty margin that accounts for all the possible sources of error. However, if we choose a highly conservative margin the number of feasible footholds can be significantly reduced. This situation is illustrated in Figure 4.8. In our evaluation, if no solution is found, then the optimal foothold is set to be the nominal one and rely on proprioception to deal with the complexity of the terrain. If a large number of examples with a nominal solution in the center are provided to the CNN, then the trained function might be biased to choose solutions always located in the center of the grid. We opt for an uncertainty

[5] A "one-hot vector" is a vector of zeros except for one of its entries, which contains the value of one.

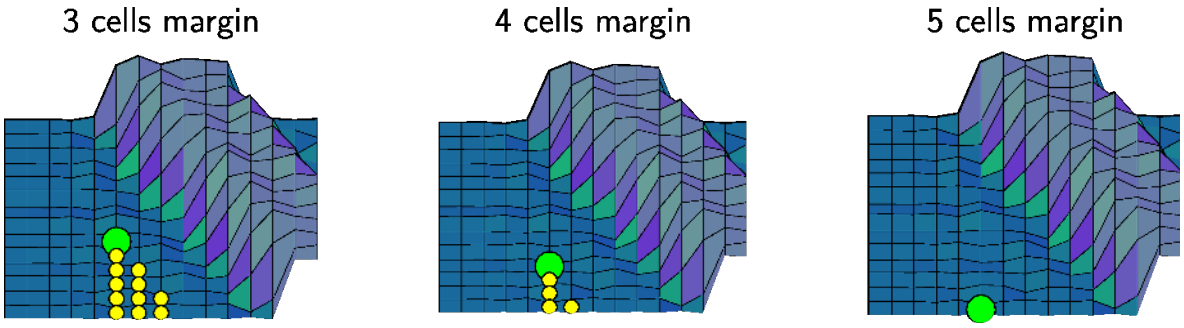


Figure 4.8: Number of feasible footholds for different values of uncertainty margin.

margin of three cells to maintain a level of robustness in the foothold selection, while also keeping a meaningful number of feasible footholds.

4.2.3 Convolutional Neural Network Architecture and Training

We decided to use a CNN to approximate the mapping $g(\mathbf{H})$ since it is better suited for image classification with respect to the logistic regression used in [2]. In particular, the convolution operation uses advantageously adjacent pixel information, similarly to traditional filter-based image classification methods [117]. Our input to the CNN is a heightmap $\mathbf{H} \in \mathbb{R}^{15\times15}$. As output, we obtain a "one hot vector" $\mathbf{v} \in \{0,1\}^{225}$ that represents the optimal landing position within the heightmap as shown in Figure 4.7.

To generate the training set we evaluate the heightmap data (see Sections 4.2.1 and 4.2.2). Let again $\mathbf{H} \in \mathbb{R}^{15\times15}$ be an input heightmap. We denote by $h_{\mathbf{w}}(\mathbf{H}) : \mathbb{R}^{15\times15} \rightarrow [0,1]^{225}$ the CNN classifier parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^d$, where $d = 8238$ is the number of parameters of the CNN. Using the mapping $g(\mathbf{H})$ outlined in Section 4.2.2, we built a dataset of N_e labeled examples $\mathbb{D} = (\mathbf{H}_i, g(\mathbf{H_i}))$ for $i = 1, .., N$, from simulated and artificially generated data. To improve robustness against noisy heightmaps, the training set is corrupted by zero-mean Gaussian noise with a standard deviation of 3 cm (steps 3 and 7 in Figure 4.3).

The CNN is trained to minimize the cross-entropy loss on $\mathbb{D}$. Namely, we approximately solve the following optimization problem

$$\underset{\mathbf{w}}{\text{minimize}} \quad -\frac{1}{N}\sum_{i=1}^{N} g(\mathbf{H}_i) \circ \log h_{\mathbf{w}}(\mathbf{H}_i) \tag{4.2}$$

by stochastic gradient descent, where $\circ$ denotes the element-wise product.

To choose the architecture of the CNN, we started from an off-the-shelf architecture (similar to LeNet [117]) and trained it using the data from [2]. This architecture provided a 100% prediction accuracy on the test set of [2]. We then decreased the size of the CNN to the minimum that maintained the 100 % accuracy. We trained the new architecture on the

dataset containing the simulated and artificially generated heightmaps. After training, the CNN from the total number of predictions, 99% corresponded to feasible footholds, from which 76% were optimal. These prediction results are summarized in Section 4.4.1. We concluded that this architecture is suitable for our application. Highly accurate predictions are not needed as long as the selected foothold is safe according to the evaluation described in Section 4.2.2. The final architecture of the CNN is depicted in Figure 4.9

The input to the CNN is a 15×15 matrix corresponding to a heightmap around the vicinity of a foothold. There are 2 convolutional layers, the first one performs convolution of the heightmap with 4 kernels and the second takes the output of the first layer and performs convolution with 8×4 kernels. Both convolutions are performed to retain the input size with the appropriate zero-padding and all kernels are 5×5. Max-pooling is used in both layers to downsample the data, providing feature maps of 8×8 for the first layer and 4×4 for the second layer. The activation function is the Rectified Linear Unit (ReLu).

4.3 Control Architecture and Implementation of the VFA

Our goal is to create an intermediate layer between planned motions and reflex-like actions. We do not intend to rely on visual information only, but rather have an interface with the reactive layer of our locomotion controller, the RCF. The main idea is to enhance such layer with reliable feedback obtained from exteroceptive sensing and to adapt the foothold locations according to the terrain during the swing phase of a specific leg. We call our approach *Vision-based Foothold Adaptation* (VFA). Figure 4.10 shows how the VFA interacts with the RCF and how it is integrated into our locomotion control framework. We divide the foothold adaptation process into four stages:

1. prediction of the nominal foothold for each leg;
2. acquisition of the terrain information in the vicinity of the nominal foothold;
3. terrain evaluation (CNN inference);

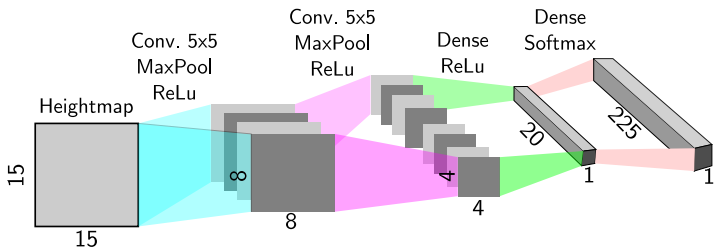


Figure 4.9: Architecture of the convolutional neural network. The input to the CNN is a 15×15 pixel heightmap and the output is a "one-hot vector" containing the optimal foothold.

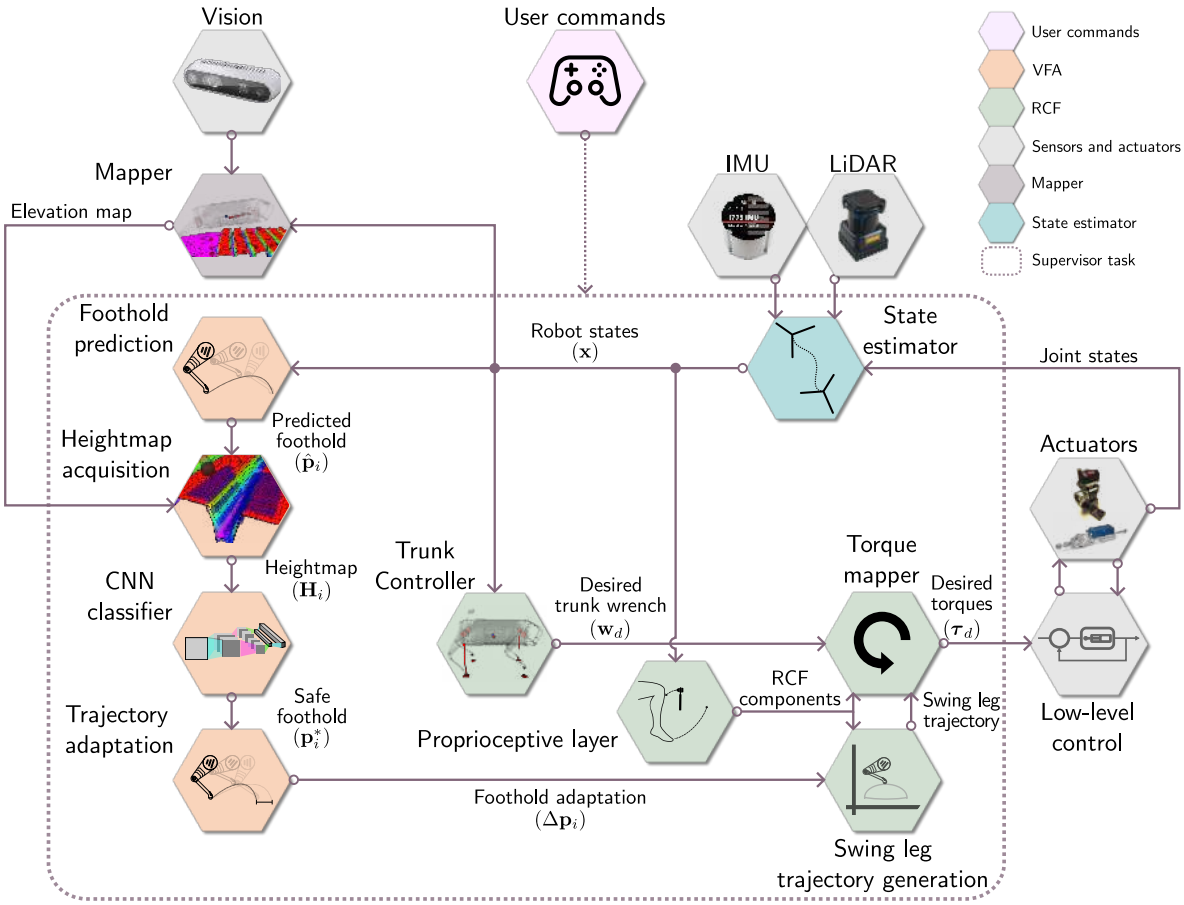


Figure 4.10: Description of the locomotion control architecture. The user commands are: forward velocity $\mathbf{V}_f$, desired yaw rate $\dot{\psi}_d$, duty factor D_f, step frequency f_s, gait sequence $\mathcal{G}$ and step height h_s. The safe foothold is defined as $\mathbf{p}_i^* = h_{\mathbf{w}}(\mathbf{H})$.

4. foot trajectory adjustment.

These stages are sequentially executed and depicted in Figure 4.11. The foothold adjustment is sent to the swing leg trajectory module of the RCF (see [52]). In the following, we explain each of the stages of the VFA.

4.3.1 Foothold Prediction

We define *foothold prediction* as the estimation of the landing position of a foot during the swing phase of its respective leg. This landing position corresponds to the *nominal foothold*. The prediction of a nominal foothold differs significantly depending on the motion of the trunk.

If there is no trunk motion during the swing of the legs (e.g., [25]), the nominal foothold can be computed at lift-off according to the desired direction of motion. Therefore, the only source

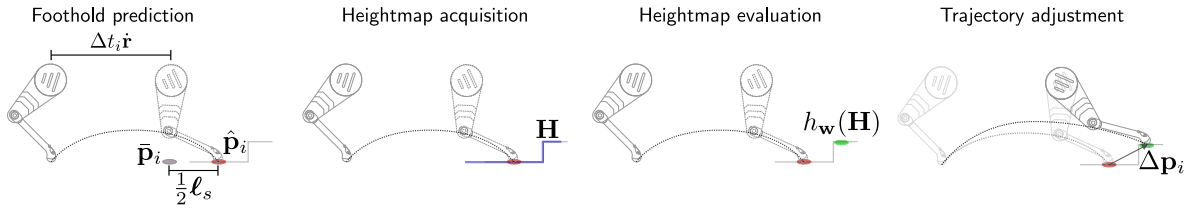


Figure 4.11: Phases of the VFA. From left to right, initially the nominal foothold is predicted based on the motion of the trunk and the predefined trajectory of the leg. Then, a heightmap is acquired in the vicinity of the nominal foothold. From the heightmap, an optimal foothold is inferred using the CNN-based foothold classifier. Once the location of the optimal foothold is computed, the trajectory of the leg is continuously adjusted to adapt to the new foothold location.

of error between the nominal and the actual foothold comes from foot trajectory tracking.

On the other hand, some gaits yield motion of the trunk while the legs are in swing phase (e.g., a diagonal trot). We refer to these kind of gaits as *dynamic gaits*. In the case of dynamic gaits the prediction of the nominal foothold has to account for the trunk velocity, in addition to the foot trajectory tracking. Hence there are two sources of uncertainty: trajectory tracking and trunk state estimation (position and velocity).

In the RCF, the swing foot trajectory is described by a half ellipse expressed in the horizontal frame of the robot[6], where the major axis corresponds to the step length. To compute the nominal foothold, we use the following approximation

$$\hat{\mathbf{p}}_i = \bar{\mathbf{p}}_i + \frac{1}{2}\boldsymbol{\ell}_s + \Delta t_i \dot{\mathbf{r}} \tag{4.3}$$

where $\hat{\mathbf{p}}_i \in \mathbb{R}^3$ is the nominal foothold position of leg i, $\bar{\mathbf{p}}_i \in \mathbb{R}^3$ is the center of the ellipse of leg i, Δt_i is the remaining swing time of leg i (for $i = LF, RF, LH, RH$), $\boldsymbol{\ell}_s \in \mathbb{R}^3$ is the step length vector, and $\dot{\mathbf{r}} \in \mathbb{R}^3$ corresponds to the velocity of the base. All vector variables are given in world coordinates. In the case of the next touchdown of a swing leg, $\Delta t_i = \frac{1-D_f}{f_s} - t_{sw,i}$, where D_f is the duty factor, f_s is the step frequency and $t_{sw,i}$ is the elapsed swing time since the latest lift-off of leg i. The first two terms in (4.3) are related to the leg trajectory, while the third term is related to the displacement of the base. A schematic drawing showing the variables involved in the foothold prediction is shown on the left-hand side of Figure 4.11.

In (4.3), the predicted foothold $\hat{\mathbf{p}}_i$ is computed with respect to the world, thus, is subject to uncertainty due to pose estimation errors. Furthermore, the term $\Delta t_i \dot{\mathbf{r}}$ corresponds to the estimation of the displacement of the base for the remaining time of the swing phase (Δt_i). The base velocity during swing $\dot{\mathbf{r}}$ is continuously updated at task frequency (250 Hz), but the foothold prediction given by (4.3) assumes that the velocity of the trunk will remain constant from the moment that it was received from the state estimator until the end of the swing phase of leg i. This is in general not true, since the velocity of the body might

[6]For an explanation on the horizontal frame, see Section 3.4.1 of Chapter 3.

change due to acceleration peaks caused by the wrenches exerted on the body due the inertia of the legs in swing phase and external disturbances. This situation leads to errors in the foothold prediction, and thus adds uncertainty to the process. These errors caused by the foothold prediction model were studied in the experiments explained in Section 4.2.2. A comparison between the actual foot landing position and the predicted one along the swing phase from (4.3) showed an average error of approximately 3 cm. One of the main goals of the strategy explained in Chapter 5 is to reduce the error due to inaccuracies in the foothold prediction model.

4.3.2 Heightmap Acquisition

The definition and parameterization of a heightmap is provided in Section 4.2.1. Given a predicted nominal foothold, the heightmap around it can be extracted from the elevation map constructed on-board by the Vision-PC (see Chapter 3). The elevation map is obtained using the Grid Map interface from [139]. Figure 4.4 shows an example of an acquired heightmap centered at a nominal foothold. We decide to express the heightmaps for each of the legs with respect to a frame located at their corresponding hip positions. We do this to be able to evaluate each leg separately (see Section 4.2.2).

4.3.3 Convolutional Neural Network Foothold Inference

To find safe footholds we continuously evaluate the acquired heightmaps of the feet during swing phase using the CNN trained as described in Section 4.2 and depicted in Figure 4.9. We compute the mapping $h_{\mathbf{w}}(\mathbf{H})$ using the heightmaps acquired from the on-board sensors as inputs.

Our goal is to find a safe foothold location within the time window of a swing phase. The CNN is up to 350 times faster with respect to the full-blown evaluation described in Section 4.2.2 and the variation in computation time is low, regardless of the shape of the map. The results regarding the computational time and the prediction accuracy of the CNN-based foothold classifier are reported in Section 4.4.

4.3.4 Foothold Adjustment

After the CNN has been trained, it can infer a foothold adaptation from a previously unseen heightmap sample. The difference between the optimal foothold and the nominal one is sent to the trajectory generator module as a relative displacement $\Delta\mathbf{p}_i$ to adapt the original foot swing trajectory. This adjustment is shown on the right-hand side of Figure 4.11.

To avoid aggressive control actions, the foothold corrections are collected from the lift-off to the apex of the swing trajectory only. Then, the controller considers the latest adjusted foothold available before the apex.

4.4 Results

In this section we present the results regarding the computational performance and the locomotion robustness we achieved with the proposed strategy, both in simulation and experiments[7].

4.4.1 Convolutional Neural Network Prediction Results

We compared the time required to evaluate the heightmaps using the full-blown evaluation $g(\mathbf{H})$ and the CNN-based classifier $h_{\mathbf{w}}(\mathbf{H})$ as the robot crosses the scenario depicted in Figure 4.12. As a metric for comparison, we use the number of clock tick counts between the beginning and the end of each computation divided by the computer clock frequency. To achieve real-time safe performances, these times should be low on average and display little variance.

To speed up the computation of the full-blown evaluation, the algorithm incrementally expands the search radius from the center of the heightmap (i.e., from the nominal foothold) and stops once a feasible foothold is found. In the case of the computation of the full-blown heuristic algorithm, the computational times range from 0.1 ms to 35 ms. The prediction from the CNN takes from 0.1 ms to 0.2 ms. The CNN-based model is therefore up to 350 times

Figure 4.12: Simulation scenario to test the VFA. The scenario consists of ten beams with a height of 15 cm and a width of 20 cm. The beams are equally spaced with a gap distance of 10 cm.

.

[7]Link to video: https://www.youtube.com/watch?v=IyuCw5QbbU8

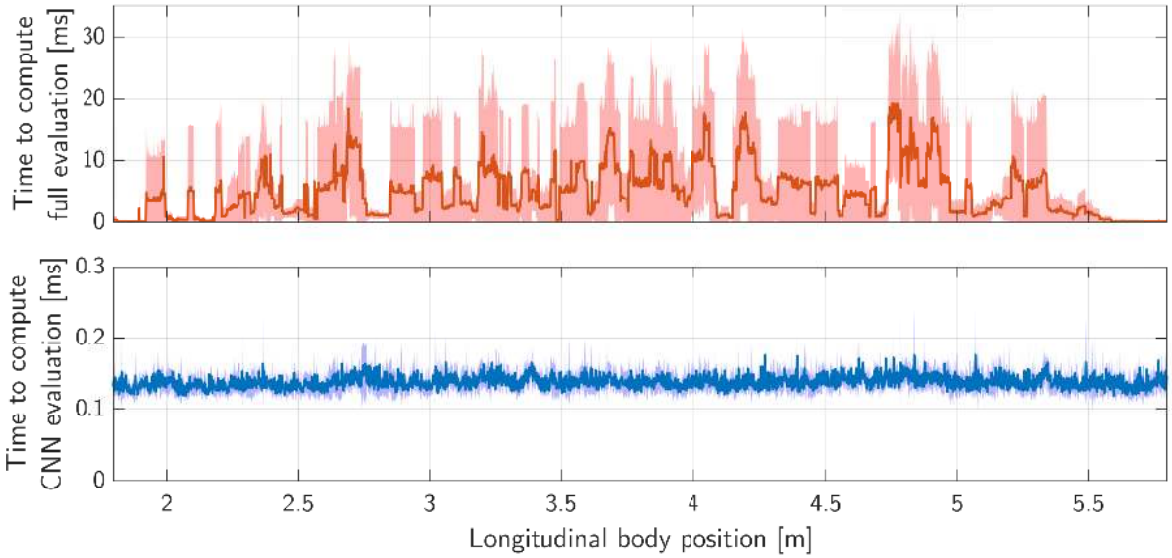


Figure 4.13: Mean and standard deviation of computation times during gap crossing simulation corresponding to five different simulations. The mean computation time of the five trials is represented by the solid line, while the shaded area around the solid line corresponds to the standard deviation of the measured time at the specific longitudinal position.

faster than the full-blown evaluation. Indeed, the computational time increases in a nonlinear fashion along with the complexity of the heightmap if the full evaluation is performed, while the neural network presents a computational time that is less sensitive to the input. This is because the function $h_{\mathbf{w}}(H)$ contains a fixed and predefined number of operations, regardless of the input. Furthermore, the duration of the supervisor task control loop of our system is 4 ms (see Figure 3.4) , making the computation of the CNN at least 20 times faster than the task rate, allowing the CNN to run in real-time. Figure 4.13 shows the mean and standard deviation of the computation times corresponding to five different trials, for both the full-blown evaluation and the CNN as the robot crosses the scenario of Figure 4.12. It can be seen that the times for the CNN are significantly shorter and the time variation shows no dependency on the complexity of the terrain.

In Table 4.1, we summarize the results of the CNN performance when predicting the optimal footholds. From a total number of 35688 examples (for both front and hind legs) we used half dataset as training set and half as testing set. It is worth noting that the percentage of accurate predictions is not notably high (about 76% for both legs). Nevertheless, by looking closely to the false positives, approximately 99% of them were deemed as safe with respect to the criteria defined in Section 4.2.2 (i.e., uncertainty margin, terrain roughness, kinematic limits and collision avoidance), although they were sub-optimal according to the distance criterion, since their distance was not the shortest to the nominal foothold. This means that even if the foothold is not optimal, the chosen foothold is still safe.

Regarding the quality of predicted footholds with respect to our previous work [2], we initially compared the results of both learning algorithms (CNN vs logistic regression) applied to the same training set used to train the logistic regression classifier, consisting of 3300 examples with 9 possible footholds. The output layer of the CNN was initially set to be 9×1, matching the number of possible footholds. We improved the prediction accuracy to nearly a 100% using the CNN-based classifier, compared to a 90% using the logistic regression. This result and the automation of the training process drove us to increase the number of outputs (from 9 to 225). We compared the CNN-based classifier with a logistic regression using 225 possible outputs and a much larger number of examples. The CNN classifier proved to have better prediction accuracy (76% vs 68%) and yielded a much lower number of parameters (8238 vs 51076).

4.4.2 Simulation and Experimental Results

To assess improvements in terms of locomotion robustness, we created challenging scenarios composed of a series of gaps. These terrain templates are built up from short beams (15 cm height and 20 cm width), equally spaced with a gap distance of 10 cm, and pallets (15 cm height). This type of scenario is used for both simulation and experimental tests: a nine-gap template for the simulation tests (see Figure 4.12) and a four-gap template for the experimental ones (see Figure 4.17).

In both simulation and experimental tests the locomotion robustness is evaluated while the robot is performing a trotting gait over the beams at different velocities (0.3 m/s and 0.5 m/s) and under external disturbances. The locomotion robustness is evaluated by observing the tracking of the robot desired velocity and trunk height. To evaluate the performance repeatability, we considered the data of 5 trials for each desired velocity. The trotting gait is performed with step frequency of 1.4 Hz, duty factor of 0.65 and a default step height of 12 cm.

Simulation results. Figure 4.14 depicts the details of the simulation scenario showing the elevation map computed by the perception system and the resulting feet trajectories during

Table 4.1: Results of prediction coming from the neural network on the test set.

Leg	Perfect foothold prediction	Non-feasible foothold prediction	Feasible foothold prediction
Front	13718/17844 (76.88%)	47/17844 (0.26%)	17797/17844 (99.74%)
Hind	13700/17844 (76.78%)	21/17844 (0.12%)	17823/17844 (99.88%)

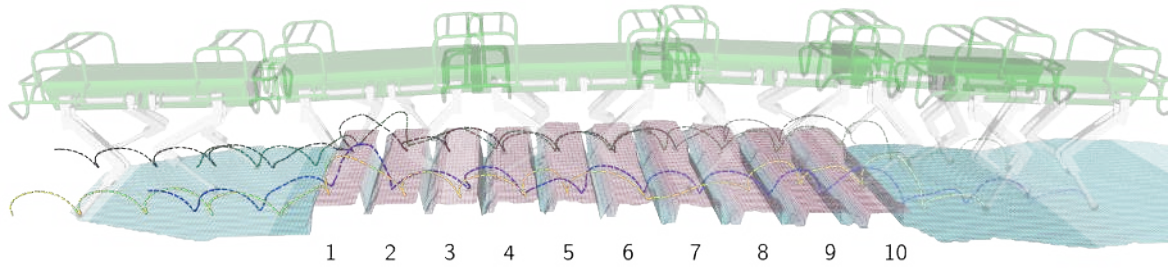


Figure 4.14: Example of HyQ crossing the gap scenario. Dashed lines correspond to feet trajectories corrected by the VFA, for a $0.5\,\mathrm{m/s}$ trotting gait. The feet trajectories are identified as: left-front (green), right-front (blue), left-hind (black) and right-hind (yellow).

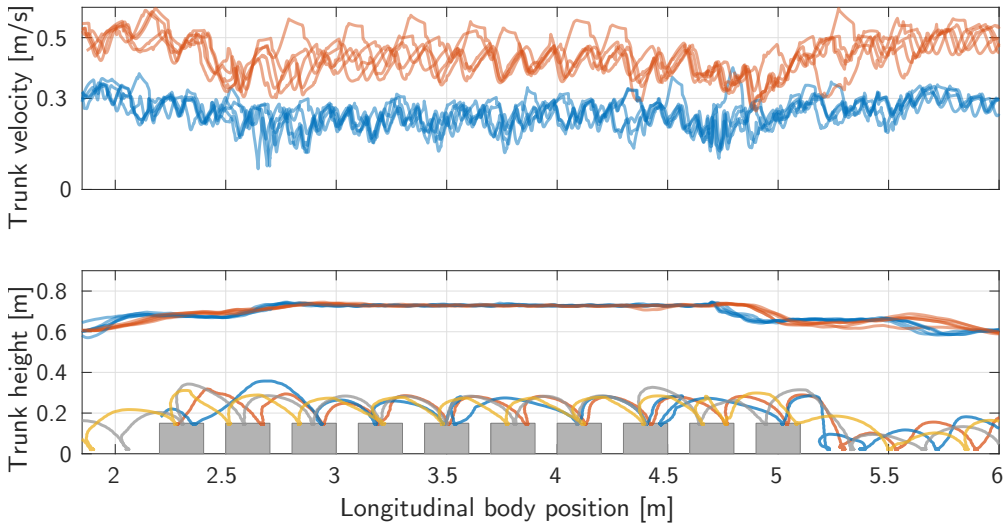


Figure 4.15: Results of gap-crossing scenario simulation. Blue lines correspond to $0.3\,\mathrm{m/s}$ and red lines correspond to $0.5\,\mathrm{m/s}$. The feet trajectories corresponding to one of the trials can be seen at the bottom of the figure.

the simulation. The footprints (dashed lines) show the effect of the foothold adaptation on the original feet trajectories.

Using the beam numbers shown at the bottom of Figure 4.14, several examples of corrections that avoided stepping inside the gap can be easily identified: left-front foot stepping on top of beam 2 to avoid landing in the gap (green line); right-hind foot stepping over beam 7 (yellow line). The right-front foot steps on beam 4 to avoid landing in the gap (blue line). The left-hind leg steps over beam 10 to avoid landing close to the edge (black line). The results of five trials at different velocities are shown in Figure 4.15.

As a last simulation example, we test the capabilities of our strategy to respond against external perturbations and show the benefits of having a continuous adaptation. During the same gap-crossing task shown before, with a $0.3\,\mathrm{m/s}$ trotting, we apply a series of perturbations

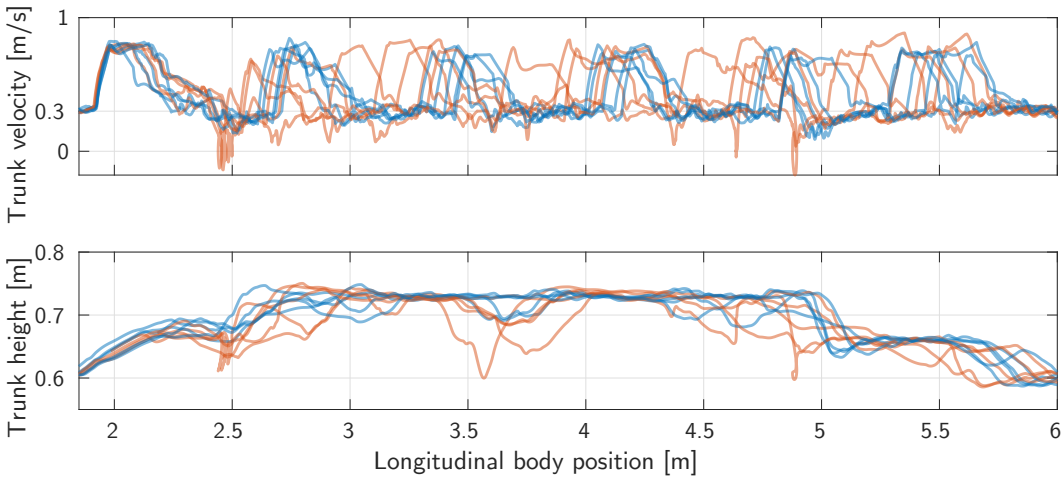


Figure 4.16: Results of gap crossing under disturbances. Red lines correspond to the trials where the correction was only computed at lift-off and blue lines correspond to the trials where the correction was continuously updated until the swing leg apex.

of 500 N every 2 s with a duration of 0.1 s each. The perturbations are applied on the base longitudinal direction disturbing the forward motion. Two cases are studied in this setup: the first corresponds to the case when the adaptation is only computed at the lift-off, and the trajectory is not corrected during the swing phase (red lines in Figure 4.16); in the second case, the foothold adaptation is continuously computed and can be modified along the swing phase (blue lines in Figure 4.16). It can be seen that in some cases, when the adaptation is only computed at the lift-off, the velocity of the trunk decreases considerably and the trunk height is less stable. Such tracking errors, caused by undesired impacts with the beams, happen due to the lack of foothold adaptation during the swing phase.

Experimental results. We prepared the setup depicted in Figure 4.17 for the gap-crossing experiments. To show how challenging this task can be for a blind robot, we also present the results of five trials using the RCF without the foothold adaptation (i.e., only with proprioceptive feedback). The results of these trials are shown in Figure 4.18.

As it can be seen, the robot was not able to complete the task without the VFA. With the VFA the goal was achieved with similar performances between the 0.3 m/s and 0.5 m/s trials. The resulting feet trajectories of one of the trials at 0.5 m/s can also be seen at the bottom part of Figure 4.18.

The robustness of the robot against external disturbances was also experimentally tested on top of the multi-gap terrain template. For such test, we placed the robot on top of one of the pallets, displayed on the bottom-left of the snapshots of Figure 4.19, and commanded the robot to keep its position on it while trotting. Then, we disturbed the robot by pulling it and

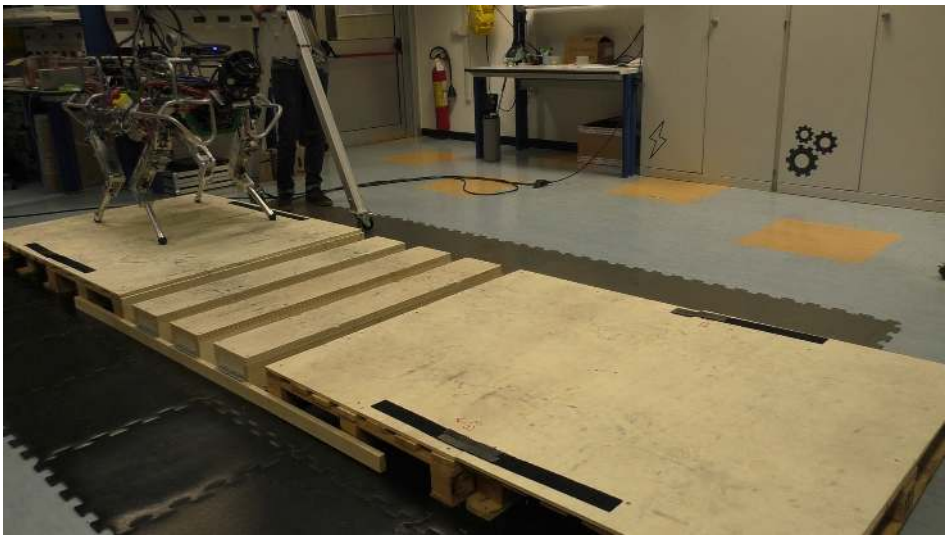


Figure 4.17: Experimental scenario used to test the VFA. The setup consists of one large pallet (width $1\,\mathrm{m}$), three wooden beams (width $20\,\mathrm{cm}$) equally spaced by $10\,\mathrm{cm}$ and two pallets with the same dimensions as the first one at a distance of $10\,\mathrm{cm}$ from the last beam. The total distance of the track is approximately $3.5\,\mathrm{m}$.

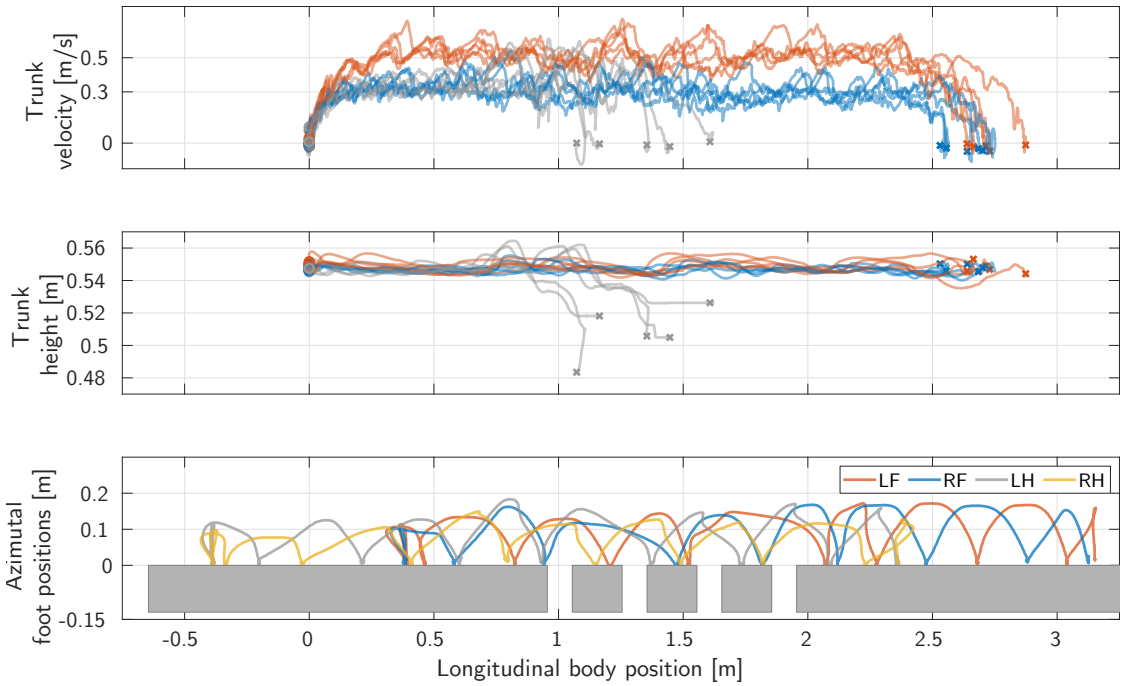


Figure 4.18: Results of the gap crossing experiments. Red lines correspond to the trials at $0.5\,\mathrm{m/s}$, blue lines correspond to the trials at $0.3\,\mathrm{m/s}$ and gray lines correspond to the trials at $0.3\,\mathrm{m/s}$ without visual feedback. The circles and crosses indicate the beginning and end of the trajectory for each of the trials, respectively. The feet trajectories corresponding to one of the succesful trials at $0.5\,\mathrm{m/s}$ can be seen at the bottom of the figure.

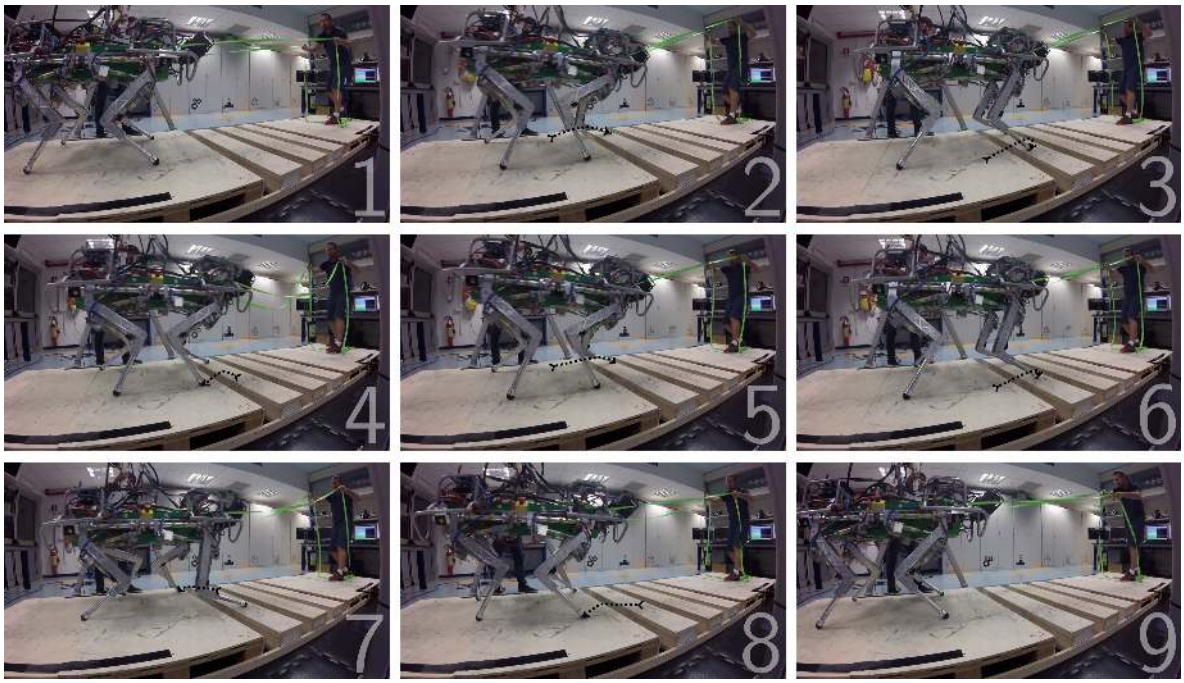


Figure 4.19: Snapshots of the robot being disturbed while avoiding to step inside the gaps.

forcing it to go repeatedly over the gaps. Figure 4.19 shows video snapshots of the experiments where the robot is subject to strong pulling forces (estimated to be around 500 N). It can also be observed that the robot is able to keep its balance and come back to its commanded position without stepping inside the first gap. It is worth highlighting the robust autonomy of the closed-loop system, since the robot is only commanded to keep its global position and no base trajectory is pre-designed while it is disturbed.

4.5 Discussion and Conclusion

4.5.1 Discussion

In this section we discuss the results reported in this chapter and possibilities of improvement regarding the CNN-based foothold classifier and the uncertainty present in the foothold adaptation process.

Foothold classifier and Learning Pipeline. The CNN-based foothold classifier demonstrated to approximate the evaluation described in Section 4.2.2. This evaluation only considers terrain morphology and robot kinematics. However, in more complex scenarios, appropriate foothold selection may also require knowledge of the robot dynamics. In Chapter 5, we account for the dynamic effects using an model predictive control (MPC)-based whole-body

controller to select a set ground reaction forces (GRFs) considering the adapted foothold locations. Thus, even if a foothold might be not "dynamically" ideal, an appropriate selection of the GRFs helps the robot to maintain its balance. An alternative would be to consider more criteria in the evaluation such as robot dynamics, surface normals, gait parameter modulation and a two-step horizon. The current compact architecture of the CNN might not be able to approximate the evaluation if more criteria are considered. Since the classifier provides such computational gain, we can design a more sophisticated architecture to take into account new criteria, without risking to exceed the available computation time. Additionally, with this low computational requirements, we can increase the size of the output space, currently consisting of 225 possible footholds, to consider larger maps and also to provide inference of other type of variables such as gait parameters.

Uncertainty. As mentioned in Section 4.2.2, there are three main sources of uncertainty that affect the foothold adaptation process, swing foot trajectory tracking, map-drift and foothold prediction errors. These three effects can cause errors that might lead to wrong estimations of the landing location of the feet. These problems can be tackled separately in order to reduce the overall errors caused by the total sum of their effects.

The swing foot trajectory tracking errors are related to the joint low-level control. A better joint tracking would consequently improve the trajectory tracking. A way to do this is by having stiffer leg and joint impedances, however, compliant legs deal better with unwanted impacts. This trade-off between joint tracking and leg compliance makes it difficult to compensate optimally for this source of uncertainty.

The uncertainty related to map-drift is directly linked to the drift present in the state estimator. Once an area of the elevation map is out of the field of view of the vision sensors, it is kept at its location based on the position of the robot with respect to the inertial frame. This means that if there is drift in the state estimator the map positioning will also be affected by this drift. Reducing state estimation drift in legged locomotion is complex due to phenomena such as impacts, link flexibility and IMU drift. In particular for the degree of precision that we are handling for foothold adaptation (2 cm) it is still very challenging.

The third source of uncertainty is related to the foothold prediction errors. This is also linked to the state estimation problem, but it is also affected by the foothold prediction model in (4.3). In this model, the body velocity $\dot{\mathbf{r}}$ and the reference position on the ground $\bar{\mathbf{p}}_i$ depend both on the state estimator. Both of these are continuously updated every 4 ms (i.e., every loop run) during the swing phase of the legs, but assumed constant from the moment that they are read to the moment that the leg touches the ground (i.e., the remaining Δt_i). This situation in general is not true. In particular, $\dot{\mathbf{r}}$ might change due to the inertia of the limbs in swing phase or due to disturbances. In Chapter 5, we aim to reduce the errors in foothold prediction by implementing an MPC-based trunk controller to reduce velocity

variations along the swing phase of the legs due to acceleration peaks with an improved selection of the GRFs.

Limitations of the method. The method presented in this chapter proved to be effective to safely choose foothold locations while performing dynamic locomotion, however, we would like to discuss its limitations in order to propose future improvements. The CNN was trained for a specific set of gait parameters, including forward velocity, duty factor and step frequency. The robot was able to complete the tasks for two different forward velocities and also for various types of gaits (as shown in the video linked). However, if the gait parameters used were very far from those used during training, the robot was not able to predict safe footholds. This limitation can be overcome by including the gait parameters as an output of the learning algorithm. Moreover, as it was mentioned before, if the errors due to foothold prediction, map drift and leg trajectory tracking are larger than the uncertainty margin (e.g., due to the lack of proper tuning of the low-level control and the state estimator), the strategy fails. This situation might occur, for example, during outdoor experiments. Furthermore, another limitation is related to the vision hardware. If the robot performs yaw rotations at a very fast rate, or moves sideways, the next foothold may not be captured within the field of view of the vision sensor mounted on the front of the robot. This could be dealt by improving the positioning of the sensor are adding extra sensors on the sides.

4.5.2 Conclusion

In this chapter, we have presented a novel strategy for continuous foothold adaptation using a CNN-based foothold classifier. The strategy allows the robot to traverse difficult scenarios while performing dynamic locomotion. We account for the motion of the base while the legs are in swing phase to compute future foothold locations and a CNN-based classifier is trained to evaluate the terrain in search of safe landing locations. Such classifier is capable of computing safe footholds up to 350 times faster than the full-blown evaluation and 20 times faster than the task frequency.

We implemented the classifier according to the control architecture in Figure 4.10 and evaluated the performance of the approach by executing dynamic trotting on challenging scenarios, where proprioception-based techniques are likely to fail. The various simulation and experimental trials, at different forward velocities and under external disturbances, demonstrated the robustness of the strategy and its repeatability. The work presented in this chapter has been published in [7].

Chapter 5

Perception Aware Model Predictive Control for Dynamic Locomotion

Model predictive control (MPC) is nowadays widely used to deal with the complex dynamics involved in the control of legged robots. Besides providing an intuitive formulation, it also yields a great level of robustness, since it can compute control actions considering the system's limitations. However, in many cases future contact locations do not consider information about the shape of the terrain.

In this chapter, we present a novel control strategy for dynamic legged locomotion that considers information about the morphology of the terrain. The strategy is built on top of two main elements: first, a contact sequence task that provides safe foothold locations based on the Vision-based Foothold Adaptation (VFA); then, a model predictive controller that considers the foothold locations given by a contact sequence task to optimize target ground reaction forces. This allows the robot to handle more difficult scenarios, such as obstacles at different elevations and orientations, in a safer and more reliable fashion.

5.1 Motivation and Rationale

Our goal is to produce robust and dynamic locomotion in complex scenarios using terrain information provided by on-board vision sensors. We combine an model predictive control (MPC)-based trunk controller with the Vision-based Foothold Adaptation (VFA) from Chapter 4. The combination of these two approaches is mutually beneficial for each other. On one hand, we exploit the low computational requirements of the VFA to provide safe contact

sequences. This enables the MPC to reason about the effects of future contacts, without having to consider them as decision variables within the optimization problem. On the other hand, accounting for future states to compute the ground reaction forces (GRFs) improves the prediction of the next foothold location, which is crucial for the accuracy of the VFA.

We start from the premise that optimizing GRFs accounting for future states using MPC will lead to better foothold predictions, since they do not only depend on foot trajectories, but also on the robot states (see Section 4.3.1). Using the control architecture depicted in Figure 4.10, the robot states are affected by acceleration peaks caused by the intrinsic reactivity of the workspace proportional-derivative (PD) control actions used to compute the desired body wrench. If the robot states have large acceleration peaks, the foothold prediction described in (4.3) is less accurate. A better management of the GRFs that considers future footstep locations would reduce acceleration peaks and improve foothold predictions.

The mathematical model that we use for prediction is a simplified centroidal dynamics that considers a number of assumptions (see Section 5.4.2), included disregarding the wrench exerted by the legs during swing on the body. We use this model to reduce the complexity of the optimization problem and still capture the meaningful dynamics of the process. However, one of the drawbacks of using this model is that when the total weight of the legs is significant with respect to that of the robot, the wrench exerted on the body due to the inertia of the legs plays a significant role in the robot dynamics. To account for these effects, we compute the wrench exerted on the body due to the desired joint accelerations of the legs and compensate for it, which is similar to a feed-forward action. We test our strategy using the quadruped robots HyQ [53] and HyQReal [132].

When the strategy is implemented the result is a more stable locomotion, which is robust to a wider range of disturbances and acts preemptively to obstacles based on visual information while performing dynamic gaits.

The rest of this chapter is organized as follows: an overview of the locomotion control strategy is given in Section 5.2; Section 5.3 explains how the contact sequences are computed; Section 5.4 is focused on explaining the reference generation and tracking of the center of mass (CoM); implementation results are shown in Section 5.5; and the discussion and conclusions are presented in Section 5.6.

5.2 Overview of the Locomotion Strategy

The block diagram shown in Figure 5.1 describes our locomotion strategy. It retains a similar structure to the strategy shown in Figure 4.10. However, there are two key differences. Firstly, instead of using the VFA only to adapt the foothold locations of the swing legs, we also compute the footholds corresponding to the subsequent two gait cycles. We define this

process as the *contact sequence task*. We explain the contact sequence task in detail in Section 5.3. Secondly, we replace the trunk controller block of the Reactive Controller Framework (RCF) with an MPC-based trunk balance controller that uses the foothold locations computed from the contact sequence task to compute optimal GRFs for the subsequent two gait cycles. Furthermore, we directly compensate for the wrench exerted by the legs during swing phase, since they are not considered in the model used for prediction by the MPC (see Section 5.4.4). This compensation is done outside of the MPC formulation to keep the optimization computationally light. We obtain a total desired wrench $\mathbf{w}_d$ from both the MPC-based controller and the leg inertia compensation and send it to the torque mapper. The torque mapper solves a similar optimization problem to the one shown in (2.18) to find a set of desired GRFs[1]. In essence, we find the set of GRFs $\mathbf{F}$ that satisfy the following expression

$$\underbrace{\begin{bmatrix} [\mathbf{p}_{LF}]_\times & [\mathbf{p}_{RF}]_\times & [\mathbf{p}_{LH}]_\times & [\mathbf{p}_{RH}]_\times \\ \mathbf{1}_3 & \mathbf{1}_3 & \mathbf{1}_3 & \mathbf{1}_3 \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} \mathbf{F}_{LF} \\ \mathbf{F}_{RF} \\ \mathbf{F}_{LH} \\ \mathbf{F}_{RH} \end{bmatrix}}_{\mathbf{F}} = \mathbf{w}_d \tag{5.1}$$

subject to friction constraints. Namely, we solve the following optimization problem

$$\begin{aligned} \underset{\mathbf{F}}{\text{minimize}} \quad & ||\mathbf{BF} - \mathbf{w}_d||_{\mathbf{L}} + ||\mathbf{F}||_{\mathbf{K}} \\ \text{subject to} \quad & -\mu\mathbf{F}_z \leq \mathbf{F}_x \leq \mu\mathbf{F}_z, \\ & -\mu\mathbf{F}_z \leq \mathbf{F}_y \leq \mu\mathbf{F}_z, \\ & F_{min} \leq \mathbf{F}_z \leq F_{max} \end{aligned} \tag{5.2}$$

After finding the GRFs $\mathbf{F}$, the desired joint torques $\boldsymbol{\tau}_d$ are obtained as

$$\boldsymbol{\tau}_d = -\mathbf{S}\mathbf{J}_c^\intercal \mathbf{F} \tag{5.3}$$

where $\mathbf{S}$ is a selection matrix that defines the legs in contact with the ground, $\mathbf{J}_c$ is a matrix with the stacked contact Jacobians for each leg. We define the computation of the desired wrench $\mathbf{w}_d$ as the *CoM tracking task*. The description of the CoM tracking task is given in Section 5.4.

The state predictions in the MPC controller are computed using the centroidal dynamics model, since it gives us a detailed enough description of the dynamics of the robot and allows us to solve the optimization problem at a sufficiently fast rate (25 Hz) during dynamic locomotion. Safe contact sequences are based on the VFA and future footholds can be continuously computed by the VFA approximately every 0.5 ms, enabling the MPC to reason about the effects of future contacts, without having to consider them in the optimization[2].

[1] In (5.1) and (5.2) all variables are defined the same as in (2.17) and (2.18), except for $\mathbf{w}_d$.

[2] Computing contact sequences using the VFA takes longer with respect to pairs of footholds (as in Chapter 4) since more footholds are inferred by the convolutional neural network (CNN) at the same time.

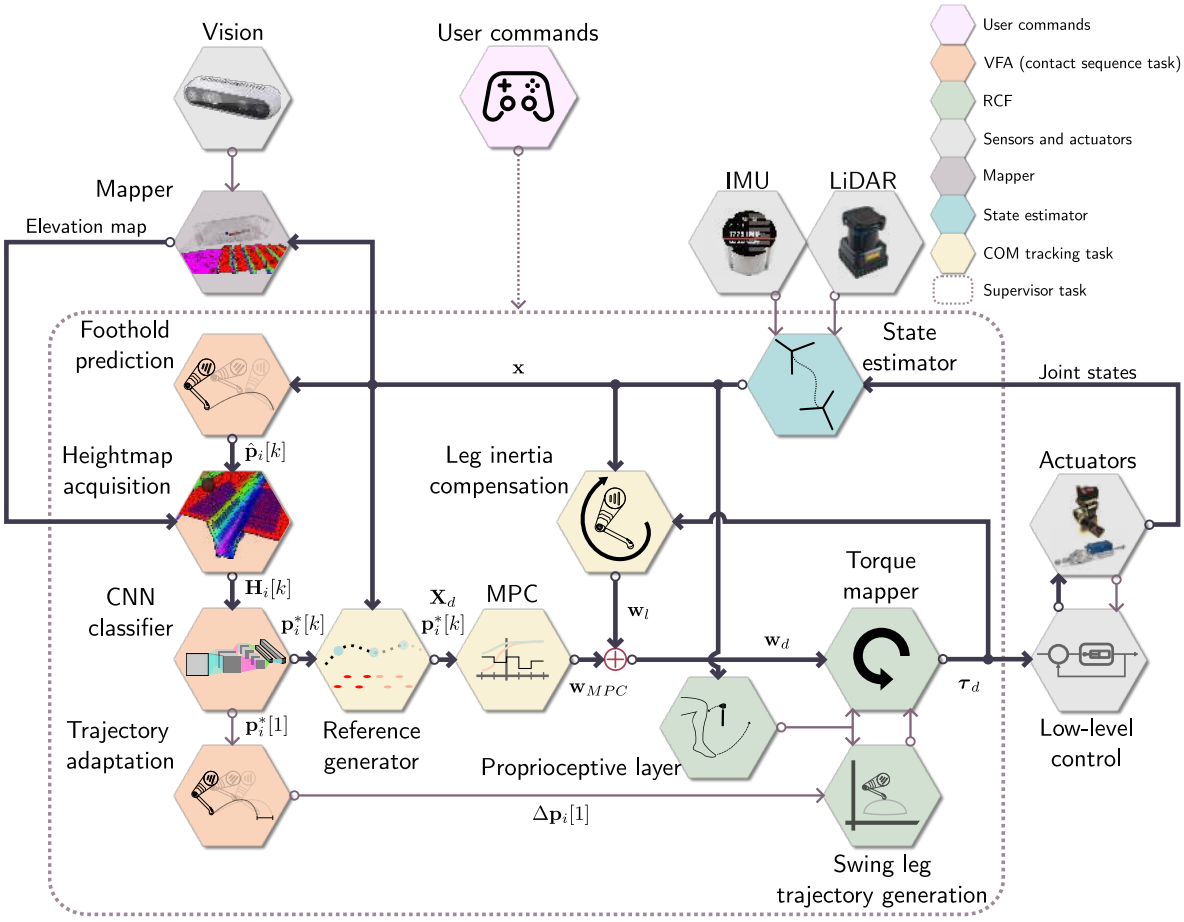


Figure 5.1: Description of the locomotion control architecture. The user commands are: forward velocity $\mathbf{V}_f$, desired yaw rate $\dot{\psi}_d$, duty factor D_f, step frequency f_s, gait sequence $\mathcal{G}$ and step height h_s. Variable k defines the discrete-time step and its length is determined by the prediction horizon (i.e., $k = 1, ..., N$). Thicker lines indicate the path of interaction between the VFA and the MPC-based controller. The non-labeled signals are the same as in Figure 4.10.

To generate position and attitude references along the prediction horizon N, we build upon the contacts computed by the VFA and fit planes for the sets of contacts at each specific sample in the prediction horizon to define the attitude.

5.3 Contact Sequence Task

We extend the usage of the VFA to provide footholds for the subsequent two strides. These two strides comprise 8 reference footholds and 16 stance changes. A stance change corresponds to the moment when a leg leaves (lift-off) or touches (touchdown) the ground. We consider both lift-off and touchdown because these events determine the instants when the contact configuration changes. This affects the number of applied GRFs and thus, the number of

decision variables in the optimization for the MPC-based trunk controller. In this dissertation we perform trotting gaits, which means that the diagonal legs share lift-off and touchdown times, and the number of stance changes could be reduced from 16 to 8. However, we keep the total number of stance changes to 16 to keep the formulation general enough to apply it to other type of gaits such as crawl, gallop, pace, or bound.

The sets of footholds at each stance change are also used to generate the CoM reference trajectory and provide the contacts to be used in the model described in Section 5.4.

The computational gain obtained by the VFA allows us to evaluate further ahead of the robot. We modify (4.3) to compute footholds at future stance changes, namely

$$\hat{\mathbf{p}}_i[k] = \bar{\mathbf{p}}_i[k] + \frac{1}{2}\boldsymbol{\ell}_s + \Delta t[k]\dot{\mathbf{r}} \tag{5.4}$$

where $\hat{\mathbf{p}}_i[k] \in \mathbb{R}^3$ is the predicted foothold of leg i at stance change k, for $k = 1, ..., 16$, $\bar{\mathbf{p}}_i[k] \in \mathbb{R}^3$ is the reference position on the ground for leg i at stance change k determined by the elliptical trajectory of the feet as in (4.3), $\boldsymbol{\ell}_s \in \mathbb{R}^3$ is the step length vector, $\Delta t[k] \in \mathbb{R}$ is the time for a stance change to happen from time instant $k = 0$ and $\dot{\mathbf{r}} \in \mathbb{R}^3$ is the body velocity at the current time instant $k = 0$. All vector variables are given with respect to the world frame. We dropped the leg index i in $\Delta t[k]$ since it represents the time for stance change to happen, irrespective of the leg that is changing its stance state.

In (5.4), $\dot{\mathbf{r}}$ is assumed constant in between future stance changes. In this way, since the gait is periodic and defined by the step frequency f_s and the duty factor D_f, we can estimate the timings for the non-immediate foot contacts. These timings are used to compute the predicted foothold locations for each of the legs at every stance change. A stance change can be a lift-off $\mathcal{L}_i$ or a touchdown $\mathcal{T}_i$, for leg $i = LF, RF, LH, RH$. We replace these stance changes for $\Delta t[k]$ in (5.4) to compute new foothold predictions $\hat{\mathbf{p}}_i[k]$. An example of stance change timings associated to their respective legs is shown in Table 5.1 for the case of $f_s = 1.4\,\text{Hz}$ and $D_f = 0.65$ and a trotting gait. Table 5.2 shows the timings ordered with respect to time stance k. After computing the sequence of predicted foothold locations, we use our CNN-based foothold classifier to adjust the predicted foothold location. This is done for the next two gait cycles (8 contacts in total and 16 stance changes). Figure 5.2 depicts an example of a generated contact sequence.

The CNN continuously provides safe contact sequences at task frequency (250 Hz). These contact sequences are used both as future foot positions and to inform the MPC controller to improve the CoM regulation, as explained in Section 5.4. This interaction is depicted in Figure 5.1 by the thicker arrows in the block diagram. One key feature of this approach is that the safe footholds are computed without including them as optimization variables in the MPC controller, which significantly decreases the complexity of the problem.

Table 5.1: Example of gait timings for $f_s = 1.4\,\text{Hz}$, $D_f = 0.65$ during trot. The touchdown times $\mathcal{T}_i$ are shown on the left and the lift-off times $\mathcal{L}_i$ on the right for all four legs. The sequence starts at $t = 0\,\text{s}$ and the legs that start the motion are the left-front and the right-hind.

Gait cycle	$\mathcal{T}_{LF}$	$\mathcal{T}_{RF}$	$\mathcal{T}_{LH}$	$\mathcal{T}_{RH}$	$\mathcal{L}_{LF}$	$\mathcal{L}_{RF}$	$\mathcal{L}_{LH}$	$\mathcal{L}_{RH}$
0	-	-	-	-	0	0.36	0.36	0
1	0.25	0.61	0.61	0.25	0.72	1.07	1.07	0.72
2	0.97	1.32	1.32	0.97	-	-	-	-

Table 5.2: Example of stance change times for $f_s = 1.4\,\text{Hz}$, $D_f = 0.65$ during trot. The top row indicates the stance change event (touchdown times $\mathcal{T}_i$ and lift-off times $\mathcal{L}_i$). The second row indicates the time instant for $k = 0, .., 15$ and the bottom row indicates the value of $\Delta t[k]$ for each time instant. The sequence starts at $t = 0\,\text{s}$ and the legs that start the motion are the left-front and the right-hind.

Stance change	$\mathcal{L}_{LF}$	$\mathcal{L}_{RH}$	$\mathcal{T}_{LF}$	$\mathcal{T}_{RH}$	$\mathcal{L}_{RF}$	$\mathcal{L}_{LH}$	...	$\mathcal{T}_{LF}$	$\mathcal{T}_{RH}$	$\mathcal{L}_{RF}$	$\mathcal{L}_{LH}$	$\mathcal{T}_{RF}$	$\mathcal{T}_{LH}$
k	0	1	2	3	4	5	...	10	11	12	13	14	15
$\Delta t[k]$	0	0	0.25	0.25	0.36	0.36	...	0.97	0.97	1.07	1.07	1.32	1.32

5.4 Center of Mass Tracking Task

In this section we detail how the trajectory of the CoM is generated and the methods that we use to track this trajectory.

5.4.1 Center of Mass Reference Trajectory

To provide the reference trajectory for the CoM along the prediction horizon, we compute its location at every stance change based on the desired gait timings using f_s and D_f. In the case of two gait cycles, there are a total of 16 stance changes, so we compute a total of 16

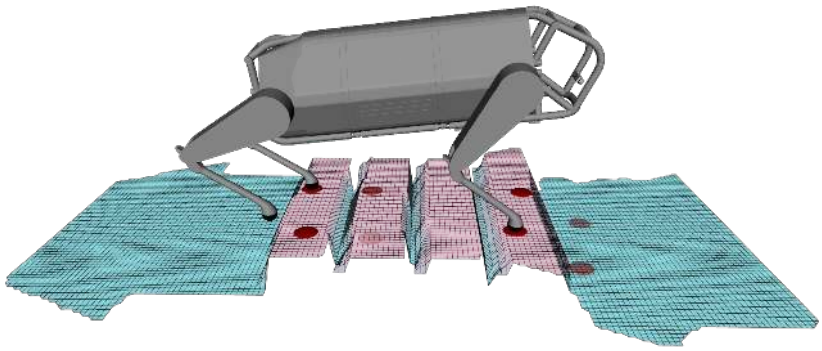


Figure 5.2: Example of a generated contact sequence. Foothold positions are indicated by red discs on the map. The more transparent the disc, the further it is in the prediction horizon.

CoM positions. Similarly to the third term of (5.4), we compute the reference yaw using the desired yaw rate $\dot{\psi}_d \in \mathbb{R}$ as

$$\psi_d[k] = \psi + \Delta t[k]\dot{\psi}_d[k] \tag{5.5}$$

where $\psi_d[k] \in \mathbb{R}$ is the yaw reference at stance change k, $\psi \in \mathbb{R}$ is the current yaw angle of the body, and $\Delta t[k]$ is defined the same as in (5.4). Using the reference for the yaw angle, we compute the reference position of the CoM with respect to the world

$$\mathbf{r}_d[k] = \mathbf{r} + \mathbf{R}_z(\Delta\psi[k])\Delta t[k]\dot{\mathbf{r}}_d[k] \tag{5.6}$$

where $\mathbf{r}_d[k] \in \mathbb{R}^3$ is the reference position for the CoM at stance change k, $\mathbf{R}_z(\Delta\psi[k]) \in SO(3)$ is the rotation matrix around the z axis about $\Delta\psi[k]$ (with $\Delta\psi[k] = \psi_d[k] - \psi$) and $\dot{\mathbf{r}}_d \in \mathbb{R}^3$ is the 3D reference velocity computed using the xy velocity reference given by the user $\mathbf{V}_f \in \mathbb{R}^2$ (initially 0 velocity in z is assumed), namely

$$\dot{\mathbf{r}}_d[k] = \begin{bmatrix} V_{f,x} & V_{f,y} & 0 \end{bmatrix}^\intercal \tag{5.7}$$

where $V_{f,x} \in \mathbb{R}$ and $V_{f,y} \in \mathbb{R}$ are the x and y components of the forward velocity $\mathbf{V}_f$ set by the user. We can then define $\mathbf{r}_d[k]$ for the next x and y positions of the CoM with respect to the world using (5.6). The reference for the body pitch angle $\theta_d \in \mathbb{R}$ and roll angle $\phi_d \in \mathbb{R}$ relies on the contact configuration at each stance change. We approximate the terrain surface as a plane and compute its orientation based on the stance legs at stance change k. We also use the contacts to define a z reference position for the body (namely, $r_{d,z}[k] \in \mathbb{R}$), setting it to remain parallel to the z world axis, at a constant distance Δz from the center position of the approximated plane $\mathbf{p}_c$. An example of a reference orientation and height for the CoM can be seen in Figure 5.3. To obtain $\dot{r}_{d,z}[k]$, $\dot{\phi}_d[k]$ and $\dot{\theta}_d[k]$ we numerically differentiate $r_{d,z}[k]$, $\phi_d[k]$ and $\theta_d[k]$, respectively. To perform the optimization required for the MPC-based trunk controller, it is required to maintain a constant sampling time along the prediction horizon N. We define a sampling time $\Delta t_s \in \mathbb{R}$ and evenly sample the 16 reference points given by the stance changes, filling the gaps in between stance changes using a zero-order hold (ZOH). We define a reference vector at evenly sampled time Δt_s as

$$\mathbf{x}_d[k] = \begin{bmatrix} \boldsymbol{\Theta}_d^\intercal[k] & \mathbf{r}_d^\intercal[k] & \dot{\boldsymbol{\Theta}}_d^\intercal[k] & \dot{\mathbf{r}}_d^\intercal[k] \end{bmatrix}^\intercal \tag{5.8}$$

where the time in between time instants is always $\Delta t_s = t[k] - t[k-1]$ and with $\boldsymbol{\Theta}_d[k] = [\theta_d[k]\ \phi_d[k]\ \psi_d[k]]^\intercal$ and $\mathbf{r}_d[k] = [r_{d,x}[k]\ r_{d,y}[k]\ r_{d,z}[k]]^\intercal$. A series of CoM references can be seen in Figure 5.4.

5.4.2 Simplified Centroidal Dynamics Model

Our MPC-based trunk controller is inspired by the work of [39]. We also model the robot as a rigid body subject to contact patches at each stance foot and we neglect the effects of

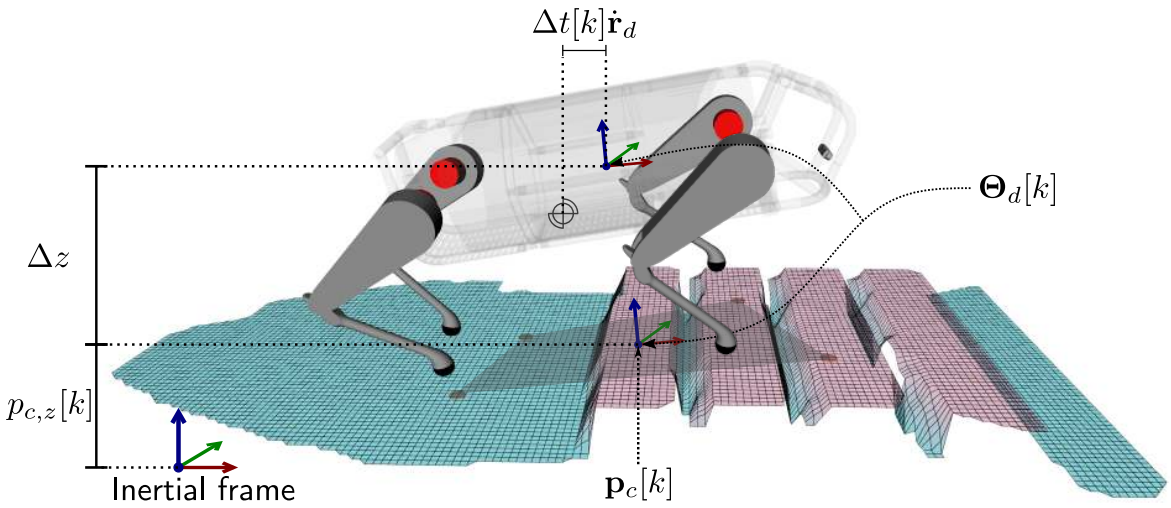


Figure 5.3: Example of the orientation and height reference. The figure shows a set of footholds (denoted by the red discs at the corners of the plane) for a stance change k in the future. The current location of the CoM is denoted by its symbol located in the trunk. The orientation of the plane formed in between the foothold locations (indicated by the coordinate frame attached to its center) specifies the orientation of the body at stance change k, $\boldsymbol{\Theta}_d[k]$. The center of the plane is indicated by $\mathbf{p}_c[k]$. The desired height Δz is a constant distance from $\mathbf{p}_c[k]$ used to define the reference height of the body $r_{d,z}[k]$. The reference height of the body $r_{d,z}[k]$ is defined such that $r_{d,z}[k] = p_{c,z}[k] + \Delta z$.

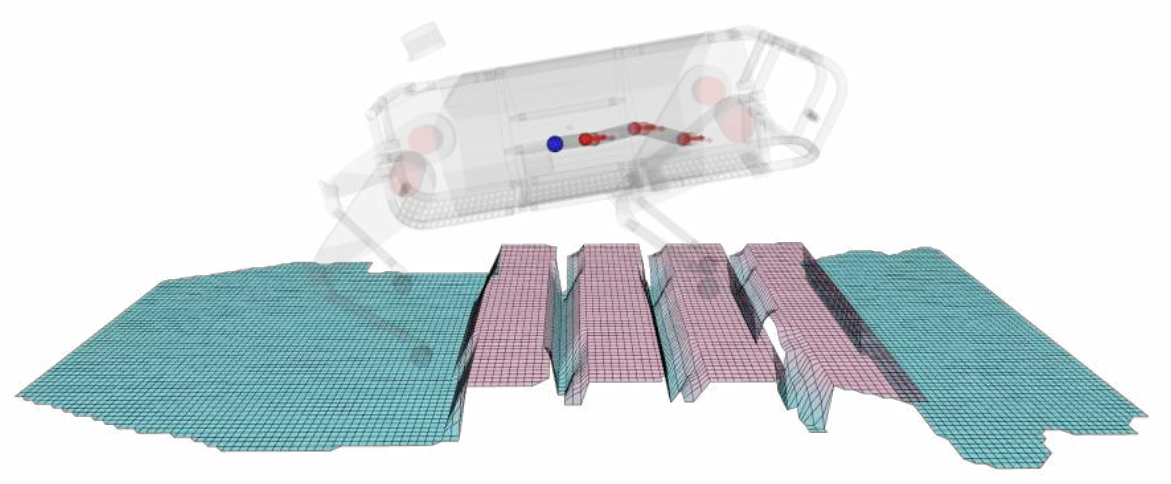


Figure 5.4: Example of the a series of CoM reference positions and orientations. The current position of the CoM is denoted by the blue sphere, the future reference positions of the CoM are denoted by smaller red spheres and the arrows connected to every sphere indicate the orientation of the body at that specific reference position.

precession and nutation as in [25]. However, there are two key differences in our approach: firstly, we do not define the reference roll and pitch angles to be zero. Additionally, although we do not explicitly consider the leg inertia in our model for control, we compensate for it by computing the wrench exerted by the legs using the actuated part of the joint-space inertia matrix and the desired accelerations of the joints. We explain how this is done in Section 5.4.4.

The dynamic model of the rigid body and its rotational kinematics are given by

$$\ddot{\mathbf{r}} = \frac{\sum_{i=1}^{n} \mathbf{F}_i}{m} + \mathbf{g} \tag{5.9}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}} = \sum_{i=1}^{n} \mathbf{p}_i \times \mathbf{F}_i \tag{5.10}$$

$$\dot{\mathbf{R}} = [\boldsymbol{\omega}]_\times \mathbf{R} \tag{5.11}$$

where $\mathbf{r} \in \mathbb{R}^3$ is the position of the CoM, $\mathbf{F}_i \in \mathbb{R}^3$ is the GRF at foot i, $m \in \mathbb{R}$ is the robot mass, $\mathbf{g} \in \mathbb{R}^3$ is the gravitational acceleration, $\mathbf{I} \in \mathbb{R}^{3\times3}$ is the inertia tensor of the robot with respect to the world, $\mathbf{p}_i \in \mathbb{R}^3$ is the i-th foot contact position, $\mathbf{R} \in SO(3)$ is the rotation matrix from body to world coordinates according to pitch θ, roll ϕ and yaw ψ angles and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the robot's angular velocity. All quantities are given with respect to the world frame unless indicated otherwise. The operator $[\mathbf{x}]_\times$ is the skew-symmetric matrix such that $[\mathbf{x}]_\times \mathbf{y} = \mathbf{x} \times \mathbf{y}$. In (5.10) we are neglecting precession and nutation effects, namely $\omega \times \mathbf{I}\omega \approx 0$. The inertia tensor $\mathbf{I}$ is defined as

$$\mathbf{I} = \mathbf{R}\mathbf{I}_B\mathbf{R}^\intercal \tag{5.12}$$

where $\mathbf{I}_B$ is the inertia tensor with respect to the body frame. This makes the inertia tensor dependent on the orientation of the body $\boldsymbol{\Theta}$. We rewrite equations (5.9), (5.10) and (5.11) in state-space form. To do this, we initially can obtain the angular velocity in terms of the body's Euler angles from (5.11) as

$$[\boldsymbol{\omega}]_\times = \dot{\mathbf{R}}\mathbf{R}^\intercal \tag{5.13}$$

which can be rewritten as

$$\boldsymbol{\omega} = \begin{bmatrix} \cos\theta\cos\psi & -\sin\psi & 0 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{5.14}$$

then, the angular rates can be obtained from (5.14) as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi/\cos\theta & \sin\psi/\cos\theta & 0 \\ -\sin\psi & \cos\psi & 0 \\ \tan\theta\cos\psi & \tan\theta\sin\psi & 1 \end{bmatrix} \boldsymbol{\omega} \tag{5.15}$$

Note that the 3 by 3 matrix on the right-hand side in (5.15) is defined as long as $\theta \neq \pi/2$, which in practice does not happen (it implies that the robot is pointed vertically). Defining

$\boldsymbol{\Theta} = [\phi\ \theta\ \psi]^\intercal$ and the matrix that maps from $\dot{\boldsymbol{\Theta}}$ to $\boldsymbol{\omega}$ as $\mathbf{T}$, we can rewrite (5.15) as

$$\dot{\boldsymbol{\Theta}} = \mathbf{T}(\boldsymbol{\Theta})\boldsymbol{\omega} \tag{5.16}$$

We define the state vector as

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\Theta} \\ \mathbf{r} \\ \boldsymbol{\omega} \\ \dot{\mathbf{r}} \\ \mathbf{g} \end{bmatrix} \tag{5.17}$$

Rearranging (5.9), (5.10) and (5.15) and defining additional states for $\mathbf{g}$ we can write the dynamics in state-space form as

$$\begin{bmatrix} \dot{\boldsymbol{\Theta}} \\ \dot{\mathbf{r}} \\ \dot{\boldsymbol{\omega}} \\ \ddot{\mathbf{r}} \\ \dot{\mathbf{g}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{T}(\boldsymbol{\Theta}) & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Theta} \\ \mathbf{r} \\ \boldsymbol{\omega} \\ \dot{\mathbf{r}} \\ \mathbf{g} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_3 & ... & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & ... & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{I}^{-1}(\boldsymbol{\Theta})[\mathbf{p}_{LF}]_\times & ... & \mathbf{I}^{-1}(\boldsymbol{\Theta})[\mathbf{p}_{RH}]_\times & \mathbf{0}_3 \\ \mathbf{1}_3/m & ... & \mathbf{1}_3/m & \mathbf{0}_3 \\ \mathbf{0}_3 & ... & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \begin{bmatrix} \mathbf{F}_{LF} \\ \mathbf{F}_{RF} \\ \mathbf{F}_{LH} \\ \mathbf{F}_{RH} \\ 0 \end{bmatrix} \tag{5.18}$$

where $\mathbf{0}_3 \in \mathbb{R}^{3\times3}$ and $\mathbf{1}_3 \in \mathbb{R}^{3\times3}$ are a zero matrix and the identity matrix, respectively. Equation (5.18) can be rewritten in a more compact way as

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\boldsymbol{\Theta})\mathbf{x}(t) + \mathbf{B}(\boldsymbol{\Theta}, \mathbf{p}_{LF}, ..., \mathbf{p}_{RH})\mathbf{u}(t) \tag{5.19}$$

where $\mathbf{u}$ is the vector of GRFs. Note that no assumptions are made about the orientation of the robot[3], except for $\theta \neq \pi/2$, and that we explicitly denote the dependence of $\mathbf{T}$ and $\mathbf{I}$ with respect to $\boldsymbol{\Theta}$.

In a similar fashion to [39], we approximate the system dynamics in (5.19) to a discrete-time linear system. Namely, for each reference vector $\mathbf{x}_d[k]$, we compute the approximate linear discrete system matrices $\mathbf{A}_d[k]$ and $\mathbf{B}_d[k]$, for $k = 1, ..., n$, where n is the number of samples in the prediction horizon.

We first substitute the feet locations $\mathbf{p}_i[k]$ obtained from the contact sequence task into matrix $\mathbf{B}(\boldsymbol{\Theta}, \mathbf{p}_{LF}, ..., \mathbf{p}_{LH})$ for every contact configuration at time instant k. However, matrices $\mathbf{A}(\boldsymbol{\Theta})$ and $\mathbf{B}(\boldsymbol{\Theta}, \mathbf{p}_{LF}, ..., \mathbf{p}_{LH})$ are still nonlinearly dependent on the body orientation. To obtain the linearized, discrete time versions of these matrices, we follow a similar argument as [39]. Assuming that the MPC-based controller will follow sufficiently close the reference trajectory given by $\mathbf{x}_d[k]$, we substitute the values of $\boldsymbol{\Theta}_d[k]$ into system matrices $\mathbf{A}(\boldsymbol{\Theta})$ and $\mathbf{B}(\boldsymbol{\Theta}, \mathbf{p}_{LF}, ..., \mathbf{p}_{LH})$. We then discretize the system matrices using a ZOH. The discretized linear system dynamics can be described as

$$\mathbf{x}[k+1] = \mathbf{A}_d[k]\mathbf{x}[k] + \mathbf{B}_d[k]\mathbf{u}[k] \tag{5.20}$$

[3] If $\theta \approxeq \phi \approxeq 0$ then: $\dot{\mathbf{x}}(t) = \mathbf{A}(\psi)\mathbf{x}(t) + \mathbf{B}(\psi, \mathbf{p}_{LF}, ..., \mathbf{p}_{RH})\mathbf{u}(t)$

5.4.3 Model Predictive Control

We can obtain a discrete time evolution of the system by successive substitution of states $\mathbf{x}[k]$ into (5.20) to obtain the state evolution from $k = 0$ to $k = n$. Then, we can describe the dynamics as

$$\mathbf{X} = \bar{\mathbf{A}}\mathbf{x}_0 + \bar{\mathbf{B}}\bar{\mathbf{u}} \tag{5.21}$$

where $\mathbf{X} \in \mathbb{R}^{15n}$ is the stacked vector of states along the prediction horizon $\mathbf{X} = [\mathbf{x}^\intercal[1],\ ...,\ \mathbf{x}^\intercal[n]]^\intercal$, $\bar{\mathbf{A}} \in \mathbb{R}^{15n\times 15n}$ and $\bar{\mathbf{B}} \in \mathbb{R}^{15n\times 12n}$ are the matrices built by successive substitution along the prediction horizon, $\mathbf{x}_0 \in \mathbb{R}^{15}$ is the current state vector and $\bar{\mathbf{u}} \in \mathbb{R}^{12n}$ is the stacked vector of ground reaction forces along the prediction horizon $\bar{\mathbf{u}} = [\mathbf{u}^\intercal[0],\ ,...,\ \mathbf{u}^\intercal[n-1]]^\intercal$. We formulate the optimization problem to minimize the weighted least-squares error between the states and the reference along the prediction horizon. We enforce the gait pattern $\mathcal{G}$ and friction consistency by setting appropriate constraints. Namely, we solve the following optimization problem

$$\begin{aligned} \underset{\bar{\mathbf{u}}}{\text{minimize}} \quad & ||\mathbf{X} - \mathbf{X}_d||_{\mathbf{L}} + ||\bar{\mathbf{u}}||_{\mathbf{K}} \\ \text{subject to} \quad & -\mu\bar{\mathbf{u}}_z \leq \bar{\mathbf{u}}_x \leq \mu\bar{\mathbf{u}}_z, \\ & -\mu\bar{\mathbf{u}}_z \leq \bar{\mathbf{u}}_y \leq \mu\bar{\mathbf{u}}_z, \\ & \mathbf{u}_{min} \leq \bar{\mathbf{u}}_z \leq \mathbf{u}_{max}, \\ & \mathbf{G}(\mathcal{G}) = \mathbf{0} \end{aligned} \tag{5.22}$$

where $\mathbf{X}_d \in \mathbb{R}^{15n}$ is the stacked vector of desired states along the prediction horizon[4], vectors $\bar{\mathbf{u}}_x \in \mathbb{R}^{4n}$, $\bar{\mathbf{u}}_y \in \mathbb{R}^{4n}$ and $\bar{\mathbf{u}}_z \in \mathbb{R}^{4n}$ correspond to the components of vector $\bar{\mathbf{u}}$ associated to coordinates x, y and z, respectively, of the GRFs, $\mu \in \mathbb{R}$ is the friction coefficient between the ground and the feet, $\mathbf{u}_{min} \in \mathbb{R}^{4n}$ and $\mathbf{u}_{max} \in \mathbb{R}^{4n}$ are limits for the z component of the GRFs, matrix $\mathbf{G} \in \mathbb{R}^{12n\times 12n}$ is a matrix that selects the components of the GRFs that are in contact according to gait $\mathcal{G}$, and matrices $\mathbf{L}$ and $\mathbf{K}$ are weighting matrices. The optimization problem defined by (5.22) is a quadratic program (QP) and can be efficiently solved by several off-the-shelf solvers. After solving the problem in (5.22), we take the first 12 entries of the optimized control input vector $\bar{\mathbf{u}}_*$, which correspond to the set of GRFs at time instant $k = 1$ and compute the desired wrench coming from the MPC-based controller as

$$\mathbf{w}_{MPC} = \sum_{i=LF}^{RH} \begin{bmatrix} \mathbf{p}_i \times \mathbf{F}_{i,*} \\ \mathbf{F}_{i,*} \end{bmatrix} \tag{5.23}$$

where $\mathbf{F}_{i,*}$ is the optimized GRF of foot i.

[4]We redefine $\mathbf{x}_d[k] = \begin{bmatrix} \boldsymbol{\Theta}_d^\intercal[k] & \mathbf{r}_d^\intercal[k] & (\mathbf{T}(\boldsymbol{\Theta}_d[k])\dot{\boldsymbol{\Theta}}_d[k])^\intercal & \dot{\mathbf{r}}_d^\intercal[k] \end{bmatrix}^\intercal$ to match the state definition of $\mathbf{x}$.

5.4.4 Leg Inertia Compensation

The MPC model used for prediction neglects leg inertia. This assumption is acceptable for quasi-static motions. However, if the leg-body weight ratio is significantly large, the wrench exerted by the legs on the body plays a significant role in the dynamics. We compensate for these effects in a simple, yet effective, manner. The floating base dynamics of a robot can be described by

$$\begin{bmatrix} \mathbf{M}_u & \mathbf{M}_{ua} \\ \mathbf{M}_{au} & \mathbf{M}_a \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\nu}} \\ \ddot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \mathbf{h}_u \\ \mathbf{h}_a \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \mathbf{J}_{c,u}^{\intercal} \\ \mathbf{J}_{c,a}^{\intercal} \end{bmatrix} \mathbf{F} \tag{5.24}$$

where $\boldsymbol{\nu} \in \mathbb{R}^6$ is the floating-base robot velocity, $\mathbf{q} \in \mathbb{R}^{n_j}$ is the joint configuration, $\mathbf{M}_u \in \mathbb{R}^{6\times 6}$ and $\mathbf{M}_a \in \mathbb{R}^{6\times n_j}$ are the direct un-actuated and actuated parts of the joint-space inertia matrix, respectively, whereas $\mathbf{M}_{ua} \in \mathbb{R}^{6\times n_j}$ and $\mathbf{M}_{au} \in \mathbb{R}^{n_j\times 6}$ correspond to the cross terms between actuated and un-actuated parts of the joint-space inertia matrix, respectively, $\mathbf{h}_u \in \mathbb{R}^6$ and $\mathbf{h}_a \in \mathbb{R}^{n_j}$ are the un-actuated and actuated vectors of Coriolis, centrifugal and gravitational terms, respectively, $\boldsymbol{\tau} \in \mathbb{R}^{n_j}$ is the vector of joint torques, $\mathbf{J}_{c,u} \in \mathbb{R}^{n_c\times 6}$ and $\mathbf{J}_{c,a} \in \mathbb{R}^{n_c\times n_j}$ are the un-actuated and actuated contact Jacobians and $\mathbf{F}$ is the vector of GRFs. The cross-term matrix $\mathbf{M}_{ua}$ maps the joint accelerations to the robot spatial force acting on the floating-base of the robot, namely

$$\mathbf{w}_l = \mathbf{M}_{ua}\ddot{\mathbf{q}} \tag{5.25}$$

In (5.25), $\mathbf{w}_l$ can be computed using measurements coming from the sensors. However, using the actual joint acceleration might lead to high frequency wrench signals. Instead, we use the desired joint acceleration $\ddot{\mathbf{q}}_d$. Thus, the leg inertia compensation wrench is given by

$$\mathbf{w}_l = \mathbf{M}_{ua}\ddot{\mathbf{q}}_d \tag{5.26}$$

and the total desired wrench is computed as

$$\mathbf{w}_d = \mathbf{w}_{MPC} + \mathbf{w}_l \tag{5.27}$$

5.5 Results

We performed simulations considering the dynamic models of HyQ [53] and HyQReal [132], two hydraulically actuated quadruped robots developed at the Dynamic Legged Systems lab (DLS) of the Istituto Italiano di Tecnologia (IIT)[5]. We use Gazebo [137] to perform our simulations. Control commands are executed at a task frequency of 250 Hz. Wrench values from the MPC-based controller $\mathbf{w}_{MPC}$ are sent at a maximum frequency of 25 Hz and we use a ZOH in between control signals. The prediction horizon is set to comprise 2

[5]Website: https://dls.iit.it/

gait cycles, partitioned in 20 samples. We solve the QP in (5.22) with a modified version of uQuadProg++ [71] to work with the C++ linear algebra library Eigen [140]. The leg inertia compensation wrench $\mathbf{w}_l$ is computed at task frequency. The mapping is done using the Grid Map interface from [139]. Weighting matrices are chosen as $\mathbf{L} = \mathbf{1}_{15n}$ and $\mathbf{K} = (1 \times 10^{-9}) \times \mathbf{1}_{12n}$, where $\mathbf{1}_a$ defines the a by a identity matrix.

5.5.1 Simulation Results

We performed three different simulations to assess the improvements in foothold prediction and locomotion robustness[6]. Below we explain in detail the outcome of these tests[7].

Leg Inertia Compensation. We performed simulations both with HyQ and HyQReal to compare the effect of the leg inertia compensation on robots with different leg-to-body weight ratios. The robots were commanded to trot on flat terrain with a forward velocity $\mathbf{V}_f$ with a magnitude of 0.5 m/s along the x direction of the horizontal frame, a step frequency f_s of 1.4 Hz, and a duty factor D_f of 0.6. We performed the simulation for six different combinations of the following control components:

1. QP: standard PD-action trunk controller;
2. LI: stance leg impedance;
3. GC: gravity compensation;
4. IC: leg inertia compensation; and
5. MPC: model predictive controller.

We call each combination of a control components *control configuration*. Table 5.3 shows all of the different control configurations.We would like to stress that configuration C_3 acts merely as a transition between the standard PD-action based controller and the MPC. This is because the MPC controller already compensates for the gravitational effects in the model and the PD-based controller needs a gravity compensation term. We start the simulation with the trunk controller used in Chapter 4, i.e., C_1. We keep the commanded $\mathbf{V}_f = 0.5\,\text{m/s}$ and change the controller configuration as the robot continues to trot. Figure 5.5 shows the error in velocity with respect to the commanded $\mathbf{V}_f$ for both HyQ and HyQReal.

[6]Link to video: https://www.youtube.com/watch?v=CqlLRdohFwM&feature=youtu.be

[7]The proposed method is able to perform locomotion while rotating around the yaw angle, however, we only evaluate our simulations with variations in the roll and pitch angle. We omit changes in the yaw angle since the yaw rate $\dot{\psi}$ only affects the z component of the angular velocity (see (5.18)).

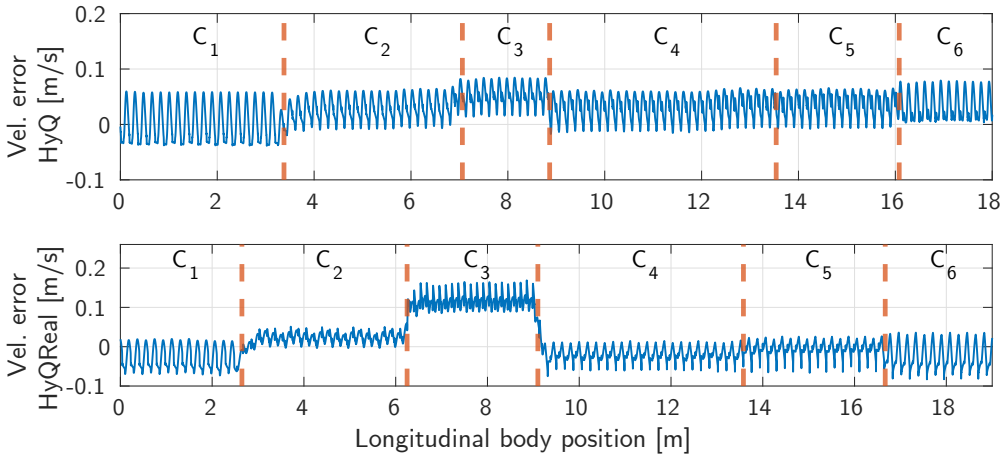


Figure 5.5: Velocity error during trot on flat terrain at constant commanded $\mathbf{V}_f = 0.5\,\mathrm{m/s}$. The gait parameters for both HyQ and HyQReal are: $D_f = 0.6$ and $f_s = 1.4\,\mathrm{Hz}$.

It can be noticed from Figure 5.5 that when the inertia compensation wrench is applied, the unwanted accelerations of the body are reduced. If we compare the effects of the leg inertia compensation between HyQ and HyQReal, it can be seen that the reduction on the variation of the velocity is more significant for HyQReal with respect to HyQ green. In both cases, the best performing configuration corresponds to the combination C_5 (fifth portion of the graphs in Figure 5.5). Under this configuration, the robot dynamics resemble more those of the MPC model (which neglects leg inertia), since the leg inertia is being considered outside of the optimization. Finally, when the MPC-based trunk controller is on with no inertia compensation (C_6); HyQ green's performance is still better compared to the initial configuration C_1. However, in the case of HyQReal, just having the MPC controller on results in a slightly larger variation in velocity with respect to C_1, since the MPC has to constantly compensate for the model discrepancies (e.g., neglected leg inertia) with the simplified centroidal dynamics model described in Section 5.4.2.

Table 5.3: Definition of the different control configurations used in the leg inertia compensation tests.

Configuration	C_1	C_2	C_3	C_4	C_5	C_6
Components	QP	QP	QP			
	LI	LI	LI	LI		
	GC	GC				
		IC	IC	IC	IC	
				MPC	MPC	MPC

Table 5.4: Root mean square and maximum absolute value of the foothold prediction error. All values are given in meters.

		LF	RF	LH	RH
QP + LI + GC	RMS(e)	0.012	0.012	0.012	0.012
	max|e|	0.095	0.095	0.088	0.082
MPC + IC	RMS(e)	0.009	0.007	0.007	0.009
	max|e|	0.074	0.061	0.060	0.076

Foothold Predictions and Robustness in the Presence of Disturbances. For the second simulation HyQReal is commanded to trot on flat terrain with the same gait parameters as in the first simulation. This time we perturb it three times with 700 N of force for a duration of 0.1 s applied on the x direction of the body frame. Table 5.4 shows the root mean square error, denoted by RMS(e), and the maximum absolute value of the foothold prediction error, denoted by max(e), for this simulation. The table helps us to compare the previous controller configuration C_1 with the MPC-based controller with leg inertia compensation C_5. The root mean square error when using MPC and leg inertia compensation (C_5) is between 25% and 41% lower with respect to the previous controller configuration (C_1). This represents between 3 mm and 5 mm of improvement. However, even if the average of the error is low in both cases, a single wrong prediction might compromise the robot's stability. In this scope, the maximum absolute value of the error is more representative of the reliability of the prediction under disturbances. In this case, the reduction of the error is between 7% and 36%. This represents between 0.6 cm and 3.5 cm of reduction of the foothold prediction error when using the MPC-based controller in combination with the leg inertia compensation. Due to the resolution that we need from the map (2×2 cm) to avoid obstacles and gaps using vision, this decrease is significant. We discuss the impact of this improvement in the discussion in Section 5.6.1.

We also point out that the inherent robustness of the MPC-based strategy also helps to improve the stabilization of the robot when being perturbed. Figure 5.6 shows the linear component in the x direction of the wrench, the angular component of the wrench around the pitch angle θ, the velocity error, the roll and pitch angles of the robot when trotting while being pushed multiple times. It can be seen that the error in velocity is less when the new strategy (C_5) is applied (right) with respect to our previous trunk controller (C_1) (left). Also the variations in the roll and pitch angles are less, since the robot shows better tracking of the orientation references given by the inclination of the terrain. The control action of the wrench rejects the disturbance along the longitudinal direction (F_x), and the moment applied about the pitch angle keeps the body from tilting forward to track the reference in orientation ($\theta_d = 0$).

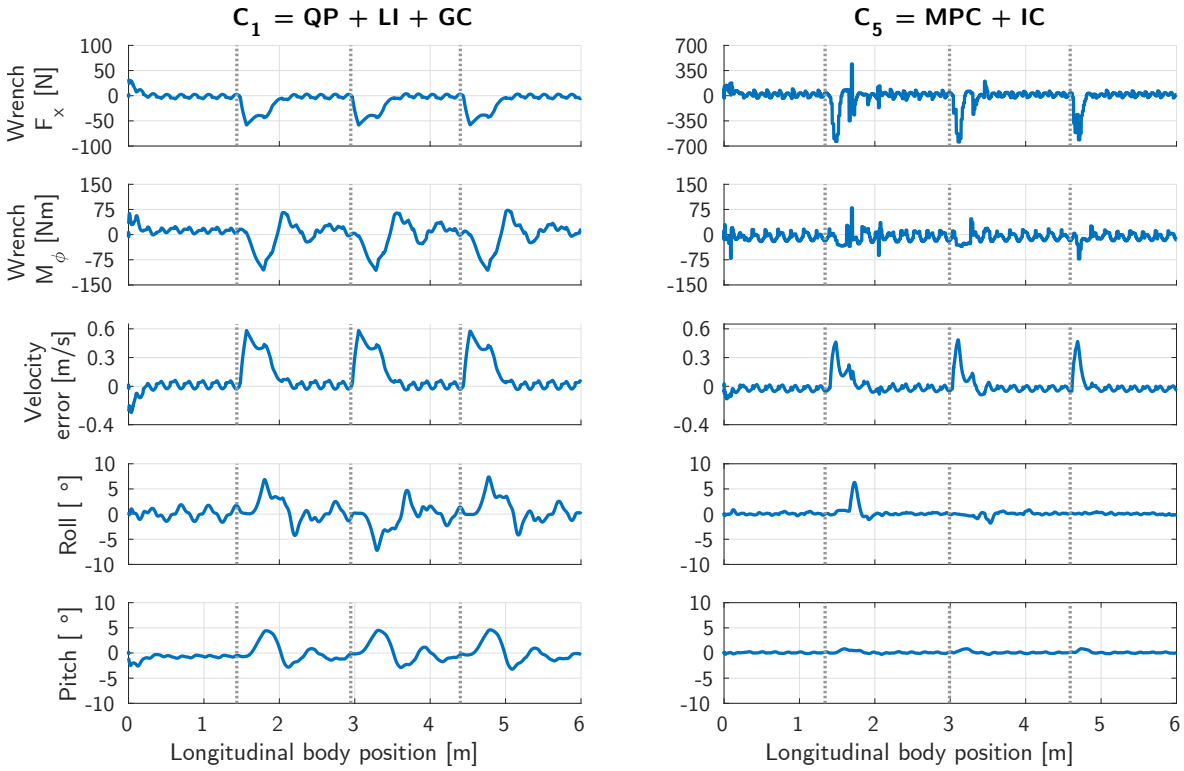


Figure 5.6: Plots corresponding to the experiment with three disturbances during trot on flat terrain. The plots on the left correspond to the control configuration used in Chapter 4 (C_1) and the plots on the right correspond to the MPC-based controller with leg inertia compensation (C_5). The vertical gray, dashed lines correspond to the moments when the disturbances were applied.

Locomotion on Challenging Terrain. To verify the improvement in performance regarding locomotion on difficult terrain, we designed the challenging scenario shown in Figure 5.7. The robot is commanded to trot with a forward velocity of 0.4 m/s, a step frequency of 1.4 Hz and a duty factor of 0.6. In order to select appropriate footholds, we use the VFA with the same control configurations of the previous experiment, namely

1. QP + LI + GC = C_1
2. MPC + IC = C_5

To test repeatability, we did four trials with each control configuration both on HyQ and HyQReal. Figure 5.8 contains the plots corresponding to pitch angle, forward velocity, body height and an example of the foot trajectories for one of the trials with configuration C_5.

From Figure 5.8, it can be seen that in the case of HyQ, the robot is able to cross the scenario for both of the control configurations in all four trials. However, the body height and body

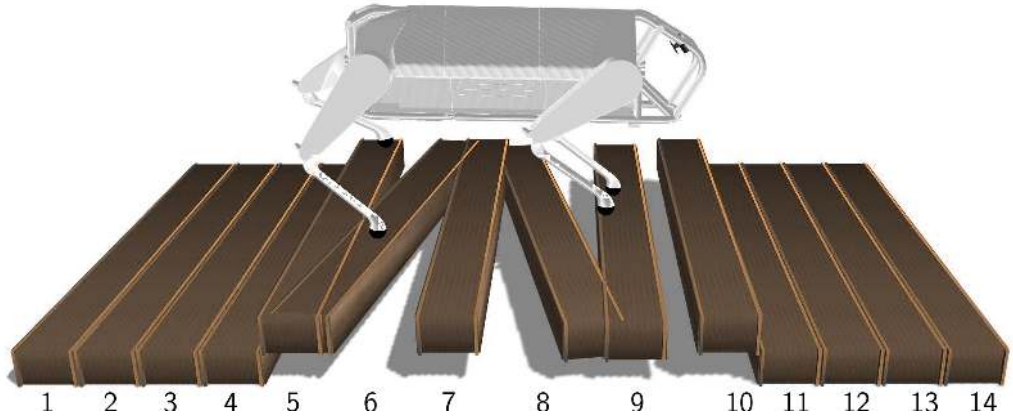


Figure 5.7: Scenario used to test the new locomotion strategy. The scenario is built up from beams with dimensions $20 \times 120 \times 15$ cm (W×L×H) at different heights and orientations.

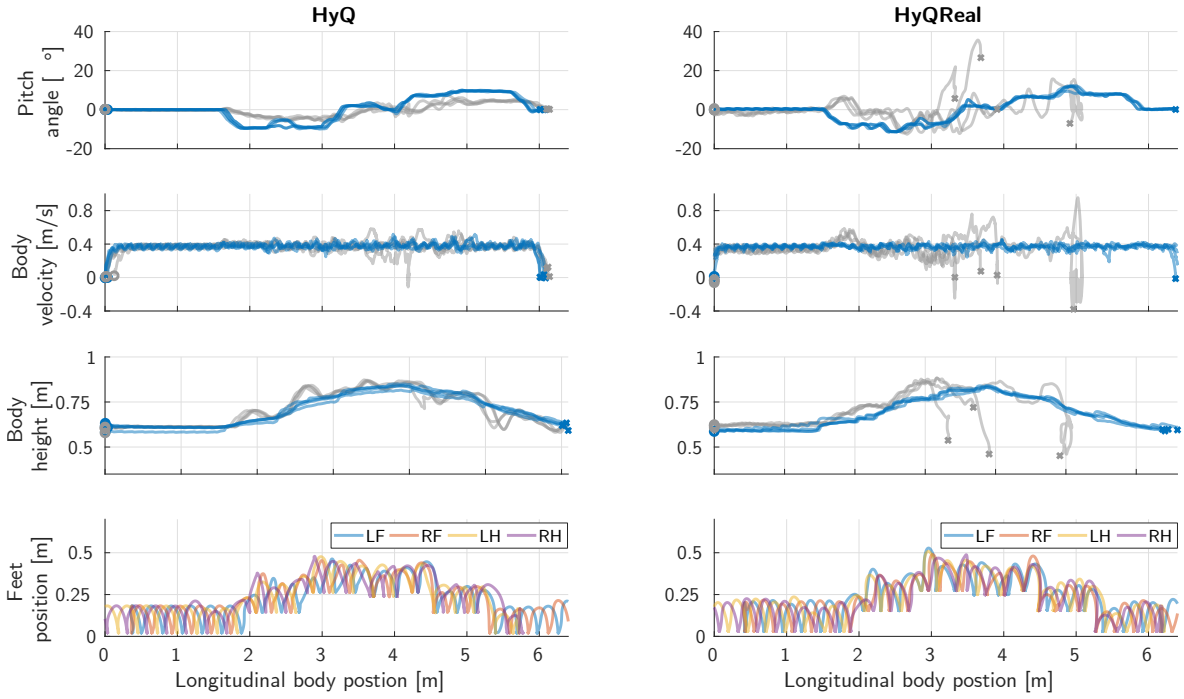


Figure 5.8: Results of the scenario crossing simulation with HyQ (left) and HyQReal (right). The top three plots show pitch angle, body velocity and body height. Gray lines correspond to trials with control configuration C_1 and blue lines correspond to control configuration C_5. Four different trials are shown for each case. The foot trajectories for all four legs corresponding to one of the successful trials with configuration C_5 are shown in the bottom plot.

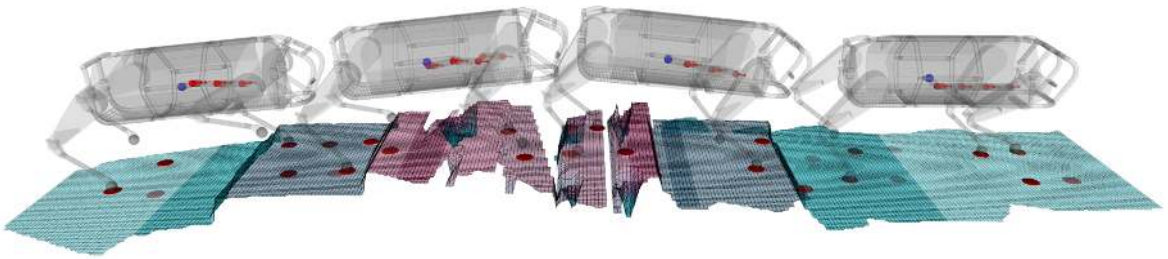


Figure 5.9: Series of snapshots of the HyQReal robot moving through the scenario. Blue spheres correspond to the position of the center of mass at the moment when the snapshot was taken and the red spheres show the reference position for the CoM along the prediction horizon. The foothold references are represented by the red discs on the ground, the more transparent the disc, the further it is in the prediction horizon.

velocity oscillate more in configuration C_1 with respect to configuration C_5. In one of the trials of configuration C_1, the velocity drops to zero. This is because the robot was trapped momentarily while crossing the scenario, but then it was able to adapt its foothold location and finish the task. In the case of HyQReal, it can be seen that all four different trials using configuration C_5 strategy were successful and the variations in linear velocity, pitch and body height are significantly reduced with respect to the configuration C_1. For C_1, the robot was not able to reach the end of the scenario for any of the trials.

Two important conclusions can be drawn from this task. Firstly, the fact that HyQ was able to cross the scenario with configuration C_1 (and HyQReal was not) is a clear indication that the leg-to-body weight ratio is playing a role in the dynamics and that compensating for it has a clear benefit in the case of HyQReal. Secondly, it shows the mutual benefits between the VFA and the MPC-based controller. The foothold prediction error is reduced when using the strategy presented here. Specifically, in the case of C_5, for all four trials and all of the legs, the maximum absolute value of the error in foothold prediction was 10 cm, while in the case of C_1 the error was up to 14 cm.

Figure 5.9 shows four overlapped snapshots of the RViz [141] visualization for HyQReal as the robot crosses the scenario, builds a map, and adjusts its footholds on the fly. Figure 5.9 also shows the reference trajectory of the CoM given at the specific moment when the snapshot was taken, and the reference foothold locations as the robot moves through the scenario

5.6 Discussion and Conclusion

5.6.1 Discussion

In this section we revisit the impact of the reduction in foothold prediction error and various routes for improvement of the strategy devised throughout this chapter.

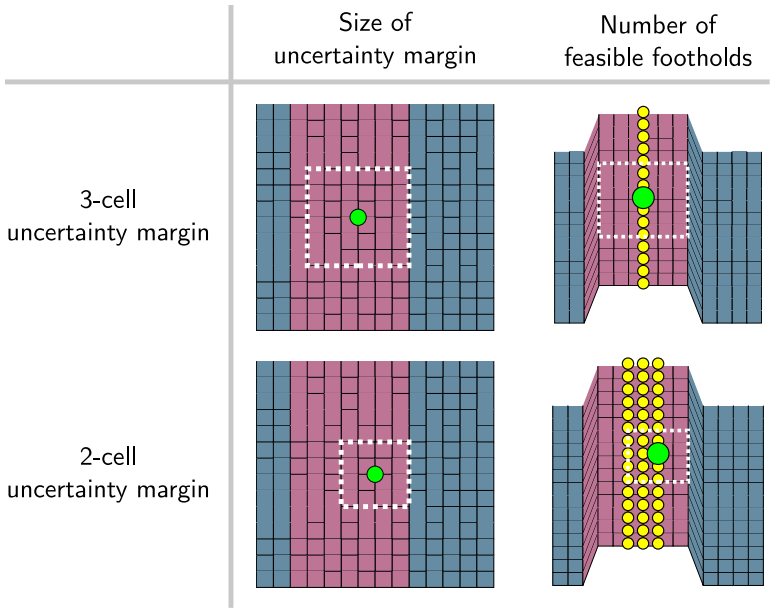


Figure 5.10: Comparison of a reduced uncertainty margin due to better foothold prediction. The uncertainty margin is indicated by the white dashed square around the optimal foothold (large green disc). Feasible footholds are denoted by the small yellow discs. Each cell is 2×2 cm.

Decrease in Foothold Prediction Error. We would like to stress the impact of the reduction of the foothold prediction error with an example. Figure 5.10 shows a comparison between an evaluated heightmap with two different uncertainty margins. The top row shows the number of feasible (safe) footholds if an uncertainty margin of 3 cells (6 cm) is used. The bottom row shows the feasible footholds for an evaluation using 2 cells (4 cm) as uncertainty margin. It can be seen that the number of feasible footholds in the bottom row triples the amount of those in the top row. If we compare the maximum absolute value of the error of each of the legs from Table 5.4, this 2 cm reduction in the error is given by the newly developed strategy. One consequence of being able to increase the confidence in your foothold prediction is the ability to find a larger number of possible choices of footholds, which also allows the robot to deal with more complex obstacles, in more difficult scenarios.

Simplified Dynamics for MPC and Whole Body Control for Torque Mapping. In the control architecture depicted in Figure 5.1, the desired wrench $\mathbf{w}_d$ is sent directly to the torque mapper block. This torque mapper solves a QP to map $\mathbf{w}_d$ to the desired GRFs as described in [25]. However, more recently in [79], a new trunk controller was devised. It makes no assumptions on the effects of the inertia of the legs or nutation and precession effects. We consider that using this whole-body controller instead of the current torque mapper block in Figure 5.1 would be beneficial for the approach presented in this chapter. In the future we would like to do a comparative study between the leg inertia compensation done in this

chapter and the use of a whole-body controller instead of the torque mapper.

Limitations of the approach. Although we are able to evaluate the terrain for the subsequent eight contacts, our simulations showed that the maximum number of time steps in the future that can be considered for the discretized MPC controller is 20, computed at a frequency of 25 Hz, which was the minimum frequency at which we could send control signals without going unstable. This limits the length of the prediction horizon and consequently the number of steps in the future that can be considered. We believe that we can speed even further the computation if we outsource the inference of the adapted footholds by the CNN to a graphics processing unit (GPU).

5.6.2 Conclusion

In this chapter, we presented a strategy based on the combination of an MPC controller and a CNN-based foothold adaptation (namely the VFA presented in Chapter 4). We showed that the interaction between these approaches is mutually beneficial and improves locomotion reliability and robustness. Firstly, we took advantage of the fast foothold computations of the VFA to provide safe contact sequences to the MPC model. These contact sequences are updated at task frequency (250 Hz) and constantly sent to the controller without compromising real time execution. Secondly, the maximum foothold prediction error was reduced by between 7% and 36% with respect to our previous strategy, equivalent to 0.6 cm and 3 cm, thanks to a better selection of the GRFs based on predictions of the future states. We also demonstrated that considering a compensation term accounting for the wrench due to the inertia of the legs, improves the performance of the MPC-based controller, due to a closer resemblance to the model used for state prediction. The various simulations validated these improvements and it remains as future work to implement the current strategy on HyQReal. The work presented in this chapter is currently under review for publication.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Legged robots might soon enter our everyday lives. They are becoming more versatile, agile and dexterous and the expectations regarding their performance in a wide variety of applications are on the rise. These higher expectations are opening doors to enter new and exciting research fields that involve their interaction with the environment. In this dissertation we focused on interactions based on visual information and we posed the problem in terms of the use of vision to increase the capability of legged robots to overcome difficult terrain. Specifically, the purpose was to generate both dynamic locomotion and foothold adaptation strategies based on visual feedback, using only on-board sensing and computing.

In Chapter 4 we devised a self-supervised machine learning pipeline for foothold classification (see Section 4.2). We trained a convolutional neural network (CNN) to approximate the evaluation of potential footholds in an area of $30 \times 30\,\text{cm}$ (i.e., a heightmap). The evaluation considered criteria such as terrain roughness, collision avoidance, process uncertainty and robot kinematics. This procedure was done off-line and in an automated manner using a large number of evaluation examples (35688). The examples were artificially generated or collected from simulation (see Section 4.2.1). The CNN-based foothold classifier was able to compute safe foothold locations within 0.1 ms, which made it suitable for dynamic locomotion.

We used the CNN-based foothold classifier to find safe landing positions for the feet and adapt the leg trajectory to reach the new footholds during dynamic locomotion of the quadruped robot HyQ both in simulation and experiments. We performed trotting at speeds as high as $0.5\,\text{m/s}$ using the Reactive Controller Framework (RCF) from Barasuol et al. [52] and the hardware/software architecture explained in Section 3.3, which computes high-level control

actions at a frequency of 250 Hz (i.e., every 4 ms). The foothold adaptations were computed within 0.2 ms, which allow the robot to adapt its foothold location continuously and in real-time. The robot was able to cross scenarios with gaps and beams that was not able to traverse when visual feedback it was not provided and only reactions based on proprioception and haptics were used. We named this Vision-based Foothold Adaptation (VFA).

In Chapter 5 we developed a strategy that combined the VFA with an model predictive control (MPC)-based whole-body controller and exploited the mutual benefits between them. Firstly, we reduced the complexity of the MPC formulation by computing future safe foothold locations using the VFA instead of including them as optimization variables. Secondly, since the VFA is subject to uncertainty related to foothold prediction errors (see Section 4.2.2), we used the MPC-based controller to reduce the error in foothold prediction. When the MPC controller was used, the error was between 0.6 cm and 3 cm less with respect to the strategy without MPC thanks to the computation of ground reaction forces (GRFs) based on the prediction of future states.

In addition, we provided empirical demonstration for the influence of the wrench exerted on the body of the robot due to the inertia of the swing legs during dynamic locomotion when it is not considered in the prediction model. We devised a simple, yet effective way to compensate for the effects of the swing legs by computing a feedforward wrench based on the desired joint acceleration and the joint-space inertia matrix. This compensation reduced the body acceleration, providing better pose and velocity tracking.

We performed simulations of the MPC-based controller with leg inertia compensation on HyQ and HyQReal. We verified the improvements with respect to previously implemented controller configurations, both in terms of foothold prediction and locomotion reliability and robustness.

We combined the VFA, the MPC-based whole-body controller and the leg inertia compensation. The outcome of this combination is a terrain-aware MPC-based dynamic locomotion strategy that: is able to overcome complex scenarios that require fast and careful selection of the feet landing positions; reduces the sensitivity to foothold prediction errors; is robust against perturbations due to changes on the terrain and external forces; and compensates for disturbances due to the inertia of the swing legs. All of these features were devised considering only on-board sensing and computing capabilities.

We strongly believe that the contributions of this dissertation constitute a step forward to perform dynamic locomotion in scenarios that require careful foothold selection. However, these advancements have given us a glance to new challenges that we would like to address in the future to improve even further the methods proposed in this work. In the following section we provide a summary of these future challenges and provide some insight on how to tackle them.

6.2 Future Work

We identified specific areas that can be improved in the future. Below we provide a summary of these improvements and categorize them according to how they were presented.

Self-Supervised Machine Learning Pipeline

- *Extension of the foothold evaluation criteria.* In Section 4.2.2 we provided a series of foothold evaluation criteria based mostly on terrain morphology and robot kinematics. Although these criteria provided good results when training and implementing the classifier, we believe that further improvements can be achieved such as considering dynamics in the evaluation criteria and a two-step horizon.

- *Use gait parameters and forward velocity as inputs to the CNN.* The input to the CNN-based classifier was a 30×30 cm heightmap. The training data was generated considering specific values of forward velocity, step frequency and duty factor[1]. Although the CNN was able to select proper footholds even when changing the forward velocity and the duty factor, we believe that the accuracy of the predictions during runtime would improve if we consider these parameters as inputs to the CNN. In this way, we could train a CNN that is able to classify foothold locations for a wider range of gait parameters.

- *Testing of new CNN architectures and overall improvement of learning algorithm.* If we want to evaluate new criteria, most likely the current compact architecture of the CNN might not be capable of properly classifying footholds. We would like to perform in depth studies comparing the trade-off between the complexity of the architecture, the foothold classification accuracy and the computational cost to understand the limitations that we may encounter considering our current computing hardware.

Vision-based Foothold Adaptation During Dynamic Locomotion

- *Method to identify occlusion.* One problem that we identified during experiments is related to occlusion. In particular, it is difficult to generate a rich enough training set for the CNN classifier considering that there will be occluded areas in the heightmap. An interesting improvement would be to find a systematic way to generate training data that also considers the possibility of occluded areas in the map.

- *Foot trajectory generation based on final and initial foot positions and clearance.* To implement the foothold adaptation, we defined a window in the trajectory that allowed

[1] In our case we trained using $\mathbf{V}_f = 0.5\,\text{m/s}$, $f_s = 1.4$ and $D_f = 0.65$ and the CNN was also able to select footholds adequately when changing both $\mathbf{V}_f$ and D_f to $0.3\,\text{m/s}$ and 0.6, respectively.

the foot to have enough clearance at the beginning of the motion and that prevented from having sudden changes of trajectory at the end of the motion. This heuristic way of choosing the trajectory was useful in most cases, but sometimes the initial clearance was not enough to overcome the obstacle or the correction was not implemented in time towards the end of the motion. To overcome this, we would like to design the trajectory according to the initial and final position of the foot, with a certain clearance from the terrain.

MPC-based Model Predictive Controller

- *Experiments.* Unfortunately by the time of submission of this dissertation, there was not enough time to perform experimental testing of the MPC-based controller with inertia compensation. It remains as future work to perform the experimental tests of the simulated results presented on this dissertation on HyQ and HyQReal.

- *Use trajectory optimization for the pose reference of the MPC.* The reference trajectory for the body pose of the robot along the prediction horizon of the MPC was based on the subsequent contact locations and its definition is based on experience regarding the robot capabilities. Because of this, we were able to define the problem as a quadratic program (QP) and solve it at a sufficiently fast rate. However, in the future we would like to investigate how to use trajectory optimization to define the pose of the robot along the prediction horizon and not rely on heuristics to define it, while keeping the optimization computationally feasible during runtime.

Leg Inertia Compensation

- *Concatenate the MPC wrench computation with a more detailed whole-body controller.* The computation of the desired wrench $\mathbf{w}_d$ is passed to a GRFs distribution and then mapped to joint torques (see Section 5.2). An alternative to this method would be to use a whole-body controller that does not have assumptions on the leg inertia and the influence of the Coriolis terms in the dynamics, such as the one used by Fahmi et al. [80]. Therein, the desired wrench $\mathbf{w}_d$ is computed based on a proportional-derivative (PD) control action in wrench space. The MPC based controller would replace this control action. Furthermore, it would be interesting to see what would be the effects of the interaction between the feedforward leg inertia compensation when implemented with a whole-body controller that has assumptions regarding the influence of the swing legs.

- *Drop assumptions in the MPC-based controller.* We chose to keep the assumptions on the influence of the swing legs and the Coriolis terms in the centroidal dynamics

model used for prediction in the MPC to achieve a fast optimization of GRFs. However, the ideal scenario is to be able to meet the computational requirements for real-time execution without having to make assumptions on the model. Reducing the number of decision variables and using algorithmic tools such as differential dynamic programming (DDP) or sequential linear quadratic (SLQ), might be reasonable solutions to solve this computationally expensive optimization problem.

Appendix A

Example of Evaluated Heightmap According to Specific Criteria

To illustrate how each criteria affects the complete heightmap evaluation, in this appendix we provide an example of a heightmap evaluated according to three different sets of criteria, and show the differences between them with respect to the optimal foothold, the number of feasible footholds and the number of discarded footholds.

The examples for each set of criteria can be seen in Figure A.1. We show the leg kinematics of HyQReal and a motion direction from left to right. We proceed to explain each of these examples of sets of criteria.

Terrain roughness and uncertainty margin

As explained in Section 4.2.2, the terrain roughness is evaluated by computing the mean and the standard deviation of the slope of a specific foothold with respect to its neighboring footholds and the size of the neighborhood is determined by the uncertainty margin. In Figure A.1, the top row shows the example heightmap considering only these two criteria. The light-blue crosses indicate the discarded footholds according to these criteria. These two criteria help to avoid stepping close to the edges of obstacles. Figure A.1 shows a discarded foothold close to the edge of the obstacle on the right-most column.

Shin collision and kinematic limits

We jointly evaluate shin collision and kinematic limits, since we use the inverse kinematics of the legs to compute the leg position when reaching every possible foothold. We discard

footholds that cannot be reached according to the leg's kinematic limits. Then, we search for intersections of the leg with the heightmap along the leg trajectory. We set a distance threshold for these intersections to account for the diameter of the lower leg tube (4.5 cm). The example of the discarded foothold on the right-most column shows how the lower-leg intersects with the surface at the moment of touchdown, which is the position with highest chances of shin collision (the leg is at its lowest position).

Frontal collision

The initial position of the foot is defined by default on the center of the left-most column of the heightmap. Then, we draw elliptical trajectories from the initial foothold to every possible foothold in the map, according to a specific step height. A schematic drawing explaining this kind of trajectory can be seen in Figure 2.3. We then look for intersections of the foot trajectory with the heightmap. An example of the discarded foothold in the bottom row of Figure A.1 shows a foot trajectory with a step height $h_s = 8\,\text{cm}$. It can be seen that the trajectory intersects the heightmap on the left-most edge of the obstacle.

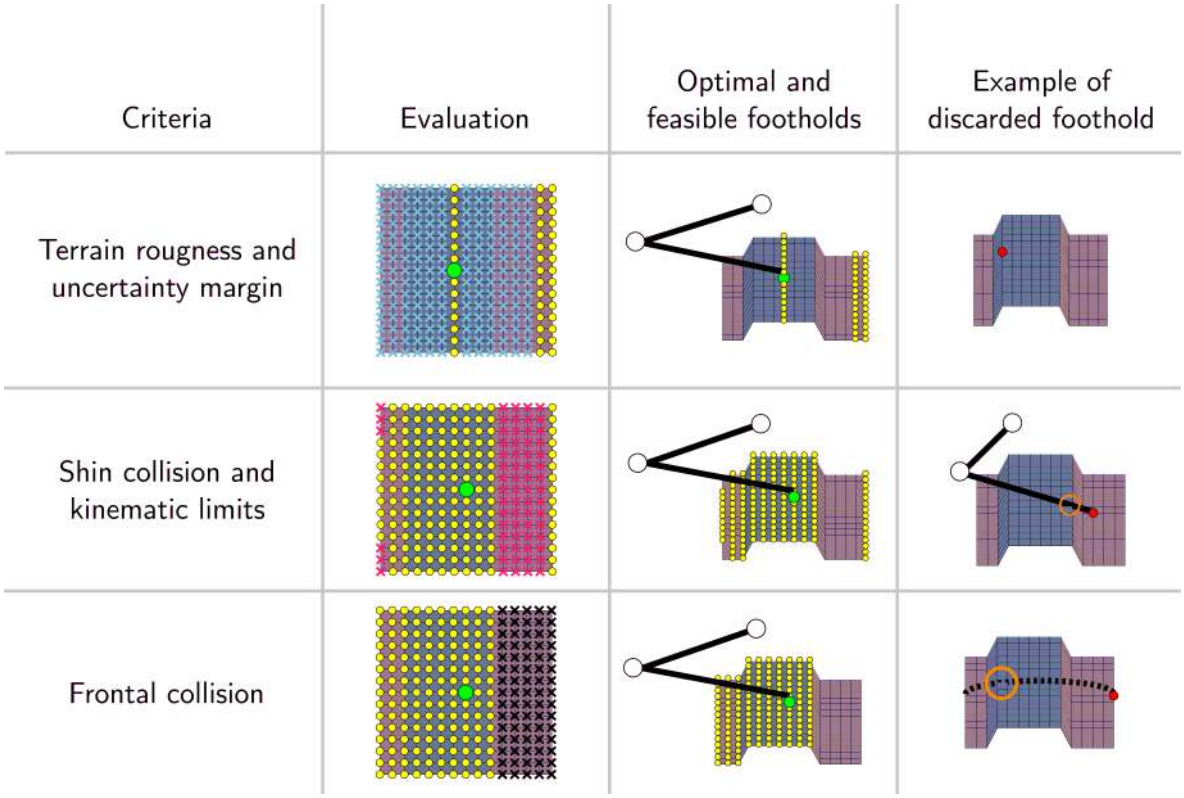


Figure A.1: Examples of different heightmap evaluation criteria. The image shows the evaluations corresponding to the leg kinematics of HyQReal and a motion direction from left to right. There are three sets of criteria, terrain roughness and uncertainty margin (top row); shin collision and kinematic limits (middle row); and frontal collision (bottom row). For each of the sets, the optimal foothold (green disc), the number of feasible footholds (yellow discs) and the discarded footholds (crosses) are shown under the evaluation column. The colors in the crosses indicate:

- discarded footholds according to terrain roughness and uncertainty margin (light-blue);
- discarded footholds according to shin collision and kinematic limits (red); and
- discarded footholds according to foot-frontal collisions (black).

In the third column, the optimal and feasible footholds are shown along with the position of the leg when the optimal foothold has been reached. The right-most column shows an example of a discarded foothold represented by a red disc and the orange circles indicate the intersections of the lower limb (middle row) and the foot trajectory with the terrain (bottom row).

References

[1] "Roy Featherstone's Website," 2019, accessed on 10.12.2019. [Online]. Available: http://www.royfeatherstone.org/

[2] V. Barasuol, M. Camurri, S. Bazeille, D. G. Caldwell, and C. Semini, "Reactive trotting with foot placement corrections through visual pattern classification," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2015, pp. 5734–5741. [Online]. Available: https://doi.org/10.1109/IROS.2015.7354191

[3] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *2008 IEEE International Conference on Robotics and Automation (ICRA)*, May 2008, pp. 811–818. [Online]. Available: https://doi.org/10.1109/ROBOT.2008.4543305

[4] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal, "Learning locomotion over rough terrain using terrain templates," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2009, pp. 167–172. [Online]. Available: https://doi.org/10.1109/IROS.2009.5354701

[5] D. Belter and P. Skrzypczyński, "Rough terrain mapping and classification for foothold selection in a walking robot," *Journal of Field Robotics*, vol. 28, no. 4, pp. 497–528, June 2011. [Online]. Available: http://dx.doi.org/10.1002/rob.20397

[6] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5761–5768. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8460731

[7] O. Villarreal, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini, "Fast and Continuous Foothold Adaptation for Dynamic Locomotion Through CNNs," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2140–2147, April 2019. [Online]. Available: https://doi.org/0.1109/LRA.2019.2899434

[8] O. Villarreal, V. Barasuol, P. Wensing, and C. Semini, "MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion," in *2020 International Conference on Robotics and Automation (ICRA)*, May 2020, in press. [Online]. Available: https://arxiv.org/abs/1909.13842

[9] S.-M. Song and K. J. Waldron, *Machines That Walk: The Adaptive Suspension Vehicle*. Cambridge, MA, USA: MIT Press, 1988. [Online]. Available: https://doi.org/10.1002/acs.4480040308

[10] Boston Dynamics, 2019, accessed on 10.12.2019. [Online]. Available: https://www.bostondynamics.com/

[11] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018, pp. 2245–2252. [Online]. Available: https://doi.org/10.1109/IROS.2018.8593885

[12] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "ANYmal - a highly mobile and dynamic quadrupedal robot," pp. 38–44, October 2016. [Online]. Available: https://doi.org/0.1109/IROS.2016.7758092

[13] C. Semini, V. Barasuol, J. Goldsmith, M. Frigerio, M. Focchi, Y. Gao, and D. G. Caldwell, "Design of the Hydraulically Actuated, Torque-Controlled Quadruped Robot HyQ2Max," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 635–646, April 2017. [Online]. Available: https://doi.org/10.1109/TMECH.2016.2616284

[14] D. Türk, L. Triebe, and M. Meboldt, "Combining Additive Manufacturing with Advanced Composites for Highly Integrated Robotic Structures," *Procedia CIRP*, vol. 50, pp. 402–407, August 2016, 26th CIRP Design Conference. [Online]. Available: https://doi.org/10.1016/j.procir.2016.04.202

[15] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, "Proprioceptive Actuator Design in the MIT Cheetah: Impact Mitigation and High-Bandwidth Physical Interaction for Dynamic Legged Robots," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 509–522, June 2017. [Online]. Available: https://doi.org/10.1109/TRO.2016.2640183

[16] V. Barasuol, O. Villarreal, D. Sangiah, M. Frigerio, M. Baker, R. Morgan, G. A. Medrano-Cerda, D. G. Caldwell, and C. Semini, "Highly-integrated hydraulic smart actuators and smart manifolds for high-bandwidth force control," *Frontiers in Robotics and AI*, vol. 5, pp. 1–15, June 2018. [Online]. Available: https://doi.org/10.3389/frobt.2018.00051

[17] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, February 1987. [Online]. Available: https://doi.org/10.1109/JRA.1987.1087068

[18] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2009, pp. 814–820. [Online]. Available: https://doi.org/10.1109/IROS.2009.5354681

[19] M. Focchi, G. A. Medrano-Cerda, T. Boaventura, M. Frigerio, C. Semini, J. Buchli, and D. G. Caldwell, "Robot impedance control and passivity analysis with inner torque and velocity feedback loops," *Control Theory and Technology*, vol. 14, no. 2, pp. 97–112, May 2016. [Online]. Available: https://doi.org/10.1007/s11768-016-5015-z

[20] P. González-de Santos, E. Garcia, and J. Estremera, *Quadrupedal Locomotion: An Introduction to the Control of Four-legged Robots*. Berlin, Heidelberg: Springer-Verlag, 2006. [Online]. Available: https://doi.org/10.1007

[21] R. McGhee and A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, pp. 331–351, August 1968. [Online]. Available: https://doi.org/10.1016/0025-5564(68)90090-4

[22] M. Vukobratović and D. Juricic, "Contribution to the synthesis of biped gait," *IEEE Transactions on Biomedical Engineering*, vol. BME-16, no. 1, pp. 1–6, January 1969. [Online]. Available: https://doi.org/0.1109/TBME.1969.4502596

[23] R. Hodoshima, T. Doi, Y. Fukuda, S. Hirose, T. Okamoto, and J. Mori, "Development of TITAN XI: a quadruped walking robot to work on slopes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2004. [Online]. Available: https://doi.org/10.1109/IROS.2004.1389449

[24] A. Roennau, G. Heppner, M. Nowicki, and R. Dillmann, "LAURON V: A versatile six-legged walking robot with advanced maneuverability," in *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, July 2014, pp. 82–87. [Online]. Available: https://doi.org/10.1109/AIM.2014.6878051

[25] M. Focchi, A. del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Autonomous Robots*, vol. 41, no. 1, pp. 259–272, January 2017. [Online]. Available: https://doi.org/10.1007/s10514-016-9573-1

[26] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, and C. Semini, "Feasible Region: an Actuation-Aware Extension of the Support Region," *ArXiv*, March 2019. [Online]. Available: https://arxiv.org/abs/1903.07999

[27] 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*, vol. 3, no. 4, pp. 3363–3370, October 2018. [Online]. Available: https://doi.org/10.1109/LRA.2018.2836441

[28] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, August 2008. [Online]. Available: https://doi.org/10.1109/TRO.2008.2001360

[29] M. Vukobratović and B. Borovac, "Zero-moment point - thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004. [Online]. Available: https://doi.org/10.1142/S0219843604000083

[30] S. Caron, Q. Pham, and Y. Nakamura, "ZMP Support Areas for Multicontact Mobility Under Frictional Constraints," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 67–80, February 2017. [Online]. Available: https://doi.org/10.1109/TRO.2016.2623338

[31] "Ghost Robotics," 2019, accessed on 10.12.2019. [Online]. Available: https://www.ghostrobotics.io/

[32] M. Hildebrand, "Vertebrate Locomotion: An Introduction: How does an animal's body move itself along?" *BioScience*, vol. 39, no. 11, pp. 764–765, December 1989. [Online]. Available: https://doi.org/10.1093/bioscience/39.11.764

[33] R. M. Alexander, *Principles of Animal Locomotion*, stu - student edition ed. Princeton University Press, 2003. [Online]. Available: http://www.jstor.org/stable/j.ctt4cg9j1

[34] L. D. Maes, M. Herbin, R. Hackert, V. L. Bels, and A. Abourachid, "Steady locomotion in dogs: temporal and associated spatial coordination patterns and the effect of speed," *Journal of Experimental Biology*, vol. 211, no. 1, pp. 138–149, December 2008. [Online]. Available: https://doi.org/10.1242/jeb.008243

[35] P. E. Hudson, S. A. Corr, and A. M. Wilson, "High speed galloping in the cheetah (acinonyx jubatus) and the racing greyhound (canis familiaris): spatio-temporal

and kinetic characteristics," *Journal of Experimental Biology*, vol. 215, no. 14, pp. 2425–2434, June 2012. [Online]. Available: https://doi.org/10.1242/jeb.066720

[36] M. Raibert, *Legged Robots That Balance*. The MIT Press, 1986. [Online]. Available: https://doi.org/10.1016/0025-5564(87)90056-3

[37] J. G. Nichol, S. P. Singh, K. J. Waldron, I. Luther R. Palmer, and D. E. Orin, "System design of a quadrupedal galloping machine," *The International Journal of Robotics Research*, vol. 23, no. 10-11, pp. 1013–1027, October 2004. [Online]. Available: https://doi.org/10.1177/0278364904047391

[38] C. Chevallereau, E. R. Westervelt, and J. W. Grizzle, "Asymptotically stable running for a five-link, four-actuator, planar bipedal robot," *The International Journal of Robotics Research*, vol. 24, no. 6, pp. 431–464, June 2005. [Online]. Available: https://doi.org/10.1177/0278364905054929

[39] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018, pp. 1–9. [Online]. Available: https://doi.org/10.1109/IROS.2018.8594448

[40] M. H. Raibert, "Trotting, pacing and bounding by a quadruped robot," *Journal of Biomechanics*, vol. 23, pp. 79–98, March 1990, international Society of Biomechanics. [Online]. Available: https://doi.org/10.1016/0021-9290(90)90043-3

[41] J. Pratt, P. Dilworth, and G. Pratt, "Virtual model control of a bipedal walking robot," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 193–198, April 1997. [Online]. Available: https://doi.org/10.1109/ROBOT.1997.620037

[42] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, February 2001. [Online]. Available: https://doi.org/10.1177/02783640122067309

[43] H. Park, M. Y. Chuah, and S. Kim, "Quadruped bounding control with variable duty cycle via vertical impulse scaling," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3245–3252, September 2014. [Online]. Available: https://doi.org/10.1109/IROS.2014.6943013

[44] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, March 2008, robotics and Neuroscience. [Online]. Available: https://doi.org/10.1016/j.neunet.2008.03.014

[45] S. Inagaki, H. Yuasa, and T. Arai, "CPG model for autonomous decentralized multi-legged robot system-generation and transition of oscillation patterns and dynamics of oscillators," *Robotics and Autonomous Systems*, vol. 44, no. 3, pp. 171–179, June 2003. [Online]. Available: https://doi.org/10.1016/S0921-8890(03)00067-8

[46] P. Arena, L. Fortuna, M. Frasca, and G. Sicurella, "An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 4, pp. 1823–1837, August 2004. [Online]. Available: https://doi.org/10.1109/TSMCB.2004.828593

[47] S. Inagaki, H. Yuasa, T. Suzuki, and T. Arai, "Wave cpg model for autonomous decentralized multi-legged robot: Gait generation and walking speed control," *Robotics and Autonomous Systems*, vol. 54, no. 2, pp. 118–126, November 2006, intelligent Autonomous Systems. [Online]. Available: ttps://doi.org/10.1016/j.robot.2005.09.021

[48] L. Righetti and A. J. Ijspeert, "Pattern generators with sensory feedback for the control of quadruped locomotion," in *2008 IEEE International Conference on Robotics and Automation (ICRA)*, May 2008, pp. 819–824. [Online]. Available: https://doi.org/10.1109/ROBOT.2008.4543306

[49] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, June 2013. [Online]. Available: https://doi.org/10.1177/0278364913489205

[50] L. Righetti and Auke Jan Ijspeert, "Programmable central pattern generators: an application to biped locomotion control," in *Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 1585–1590. [Online]. Available: https://doi.org/10.1109/ROBOT.2006.1641933

[51] R. Héliot and B. Espiau, "Multisensor Input for CPG-Based Sensory—Motor Coordination," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 191–195, February 2008. [Online]. Available: https://doi.org/10.1109/TRO.2008.915433

[52] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2554–2561, May 2013. [Online]. Available: https://doi.org/10.1109/ICRA.2013.6630926

[53] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ – a hydraulically and electrically actuated quadruped

robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, August 2011. [Online]. Available: https://doi.org/10.1177/0959651811402275

[54] C. Boussema, M. J. Powell, G. Bledt, A. J. Ijspeert, P. M. Wensing, and S. Kim, "Online gait transitions and disturbance recovery for legged robots via the feasible impulse set," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1611–1618, April 2019. [Online]. Available: https://doi.org/10.1109/LRA.2019.2896723

[55] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, "Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1096–1103. [Online]. Available: https://doi.org/10.1109/ICRA.2017.7989131

[56] 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. [Online]. Available: https://doi.org/10.1109/LRA.2018.2798285

[57] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, June 2018. [Online]. Available: https://doi.org/10.1109/TRO.2018.2819658

[58] Y. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient Humanoid Contact Planning using Learned Centroidal Dynamics Prediction," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 5280–5286. [Online]. Available: https://doi.org/10.1109/ICRA.2019.8794032

[59] M. Neunert, M. Giftthaler, M. Frigerio, C. Semini, and J. Buchli, "Fast derivatives of rigid body dynamics for control, optimization and estimation," in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, December 2016, pp. 91–97. [Online]. Available: https://doi.org/10.1109/SIMPAR.2016.7862380

[60] M. Giftthaler, M. Neunert, M. Stäuble, M. Frigerio, C. Semini, and J. Buchli, "Automatic differentiation of rigid body dynamics for optimal control and estimation," *Advanced Robotics*, vol. 31, no. 22, pp. 1225–1237, November 2017. [Online]. Available: https://doi.org/10.1080/01691864.2017.1395361

[61] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Computational co-optimization of design parameters and motion trajectories for robotic systems," *The International*

Journal of Robotics Research, vol. 37, no. 13-14, pp. 1521–1536, June 2018. [Online]. Available: https://doi.org/10.1177/0278364918771172

[62] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1502–1509, July 2017. [Online]. Available: https://doi.org/10.1109/LRA.2017.2665685

[63] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, July 2018. [Online]. Available: https://doi.org/10.1109/LRA.2017.2779821

[64] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 93–100. [Online]. Available: https://doi.org/10.1109/ICRA.2017.7989016

[65] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 2641–2648. [Online]. Available: https://doi.org/10.1109/ROBOT.2006.1642100

[66] R. Featherstone, *Rigid Body Dynamics Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007. [Online]. Available: https://doi.org/10.1007/978-1-4899-7560-7

[67] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5

[68] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, no. 2, pp. 161–176, October 2013. [Online]. Available: https://doi.org/10.1007/s10514-013-9341-4

[69] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5776–5782. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8460537

[70] S. Nobili, M. Camurri, V. Barasuol, M. Focchi, D. Caldwell, C. Semini, and M. Fallon, "Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots," in *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017. [Online]. Available: https://doi.org/10.15607/RSS.2017.XIII.007

[71] L. Di Gaspero and E. Moyer, "Quadprog++," 1998. [Online]. Available: http://quadprog.sourceforge.net

[72] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical Programming*, vol. 27, no. 1, pp. 1–33, September 1983. [Online]. Available: https://doi.org/10.1007/BF02591962

[73] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, March 2016. [Online]. Available: https://doi.org/10.1007/s10514-015-9479-3

[74] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the darpa robotics challenge," *Journal of Field Robotics*, vol. 32, no. 2, pp. 293–312, January 2015. [Online]. Available: https://doi.org/10.1002/rob.21559

[75] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. McCrory, J. van Egmond, M. Griffioen, M. Floyd, S. Kobus, N. Manor, S. Alsheikh, D. Duran, L. Bunch, E. Morphis, L. Colasanto, K. H. Hoang, B. Layton, P. Neuhaus, M. Johnson, and J. Pratt, "Summary of Team IHMC's virtual robotics challenge entry," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, October 2013, pp. 307–314. [Online]. Available: https://doi.org/10.1109/HUMANOIDS.2013.7029992

[76] S.-H. Lee and A. Goswami, "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground," *Autonomous Robots*, vol. 33, no. 4, pp. 399–414, November 2012. [Online]. Available: https://doi.org/10.1007/s10514-012-9294-z

[77] B. Henze, M. A. Roa, and C. Ott, "Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios," *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1522–1543, July 2016. [Online]. Available: https://doi.org/10.1177/0278364916653815

[78] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, and A. Albu-Schaffer, "Overview of the torque-controlled humanoid robot toro," in *IEEE-RAS International Conference on Humanoid Robots*, November 2014, pp. 916–923. [Online]. Available: https://doi.org/10.1109/HUMANOIDS.2014.7041473

[79] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini, "Passive whole-body control for quadruped robots: Experimental validation over challenging terrain," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2553–2560, July 2019. [Online]. Available: https://doi.org/10.1109/LRA.2019.2908502

[80] S. Fahmi, M. Focchi, A. Radulescu, G. Fink, V. Barasuol, and C. Semini, "STANCE: Locomotion adaptation over soft terrain," *IEEE Transactions in Robotics*, pp. 1–15, October 2019. [Online]. Available: https://doi.org/10.1109/TRO.2019.2954670

[81] D. Kim, J. Lee, J. Ahn, O. Campbell, H. Hwang, and L. Sentis, "Computationally-robust and efficient prioritized whole-body controller with contact constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IROS.2018.8593767

[82] S. Caron, Q.-C. Pham, and Y. Nakamura, "Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench for rectangular support areas," in *Proceedings of the 2015 IEEE-RAS International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 5107–5112. [Online]. Available: https://doi.org/10.1109/ICRA.2015.7139910

[83] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, March 2006. [Online]. Available: https://doi.org/10.1007/s10107-004-0559-y

[84] A. Sideris and J. E. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems," in *Proceedings of the 2005, American Control Conference*, June 2005, pp. 2275–2280 vol. 4. [Online]. Available: https://doi.org/10.1109/ACC.2005.1470308

[85] 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, July 2018. [Online]. Available: https://doi.org/10.1109/LRA.2018.2794620

[86] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006. [Online]. Available: https://doi.org/10.1007/978-0-387-40065-5

[87] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, November 2014, pp. 295–302. [Online]. Available: https://doi.org/10.1109/HUMANOIDS.2014.7041375

[88] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, November 2016, pp. 842–849. [Online]. Available: https://doi.org/10.1109/HUMANOIDS.2016.7803371

[89] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, December 2018. [Online]. Available: https://doi.org/10.1109/TRO.2018.2862902

[90] T. van den Boom, *Lecture Notes for the Course SC4060: Model Predictive Control*. TU Delft, 2013.

[91] E. F. Camacho and C. Bordons, *Model Predictive Control*. London, UK: Springer, London, 2007. [Online]. Available: https://doi.org/10.1007/978-1-4471-3398-8

[92] 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, December 1999. [Online]. Available: http://jeb.biologists.org/content/202/23/3325

[93] R. Full and M. Tu, "Mechanics of six-legged runners," *Journal of Experimental Biology*, vol. 148, no. 1, pp. 129–146, January 1990. [Online]. Available: https://jeb.biologists.org/content/148/1/129

[94] ——, "Mechanics of a rapid running insect: two-, four- and six-legged locomotion," *Journal of Experimental Biology*, vol. 156, no. 1, pp. 215–231, March 1991. [Online]. Available: https://jeb.biologists.org/content/156/1/215

[95] P. Holmes, R. J. Full, D. Koditschek, and J. Guckenheimer, "The dynamics of legged locomotion: Models, analyses, and challenges," *Society for Industrial and Applied Mathematics Review*, vol. 48, no. 2, pp. 207–304, August 2006. [Online]. Available: https://doi.org/10.1137/S0036144504445133

[96] 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)*, October 2001, pp. 239–246. [Online]. Available: https://doi.org/10.1109/IROS.2001.973365

[97] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," pp. 1620–1626, September 2003. [Online]. Available: https://doi.org/10.1109/ROBOT.2003.1241826

[98] A. Sherikov, D. Dimitrov, and P. Wieber, "Whole body motion controller with long-term balance constraints," in *2014 IEEE-RAS International Conference on Humanoid Robots*, November 2014, pp. 444–450. [Online]. Available: https://doi.org/10.1109/HUMANOIDS.2014.7041399

[99] M. Krause, J. Englsberger, P.-B. Wieber, and C. Ott, "Stabilization of the capture point dynamics for bipedal walking based on model predictive control," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 165 – 171, 2012, 10th IFAC Symposium on Robot Control. [Online]. Available: https://doi.org/10.3182/20120905-3-HR-2030.00165

[100] S. Faraji, S. Pouya, C. G. Atkeson, and A. J. Ijspeert, "Versatile and robust 3D walking with a simulated humanoid robot (Atlas): A model predictive control approach," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1943–1950. [Online]. Available: https://doi.org/10.1109/ICRA.2014.6907116

[101] M. Shafiee-Ashtiani, A. Yousefi-Koma, and M. Shariat-Panahi, "Robust bipedal locomotion control based on model predictive control and divergent component of motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3505–3510. [Online]. Available: https://doi.org/10.1109/ICRA.2017.7989401

[102] A. Zamparelli, N. Scianca, L. Lanari, and G. Oriolo, "Humanoid Gait Generation on Uneven Ground using Intrinsically Stable MPC," *12th IFAC Symposium on Robot Control SYROCO 2018*, vol. 51, no. 22, pp. 393–398, December 2018. [Online]. Available: https://doi.org/10.1016/j.ifacol.2018.11.574

[103] S. Xin, R. Orsolino, and N. Tsagarakis, "Online Relative Footstep Optimization for Legged Robots Dynamic Walking Usinge Discrete-Time Model Predictive Control," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2019, pp. 1–9, in press.

[104] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, October 2013, pp. 292–299. [Online]. Available: https://doi.org/10.1109/HUMANOIDS.2013.7029990

[105] 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, July 2018. [Online]. Available: https://doi.org/10.1109/LRA.2018.2800124

[106] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, November 2017, pp. 577–584. [Online]. Available: https://doi.org/10.1109/HUMANOIDS.2017.8246930

[107] "Google DeepMind," 2019, accessed on 10.12.2019. [Online]. Available: https://deepmind.com/

[108] N. Heess, T. Dhruva, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, ZiyuWang, S. M. A. Eslami, M. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," *ArXiv*, July 2017. [Online]. Available: https://arxiv.org/abs/1707.02286

[109] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 41:1–41:13, July 2017. [Online]. Available: http://doi.acm.org/10.1145/3072959.3073602

[110] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 143:1–143:14, July 2018. [Online]. Available: http://doi.acm.org/10.1145/3197517.3201311

[111] "Agility Robotics," 2019, accessed on 10.12.2019. [Online]. Available: https://www.agilityrobotics.com/

[112] 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)*, October 2018, pp. 1241–1246. [Online]. Available: https://doi.org/10.1109/IROS.2018.8593722

[113] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative Gait Design for Cassie using DeepRL," in *Dynamic Walking 2019*, June 2019. [Online]. Available: https://zhaomingxie.github.io/publications/DynamicWalking2019.pdf

[114] 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, December 2019. [Online]. Available: https://doi.org/10.1126/scirobotics.aau5872

[115] "ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)," 2012, accessed on 10.01.2020. [Online]. Available: http://www.image-net.org/challenges/LSVRC/2012/

[116] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *International Conference on Neural Information Processing Systems (NIPS)*, December 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257

[117] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, November 1998. [Online]. Available: https://doi.org/10.1109/5.726791

[118] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICMLâĂŹ10. Madison, WI, USA: Omnipress, June 2010, p. 807âĂŞ814. [Online]. Available: https://doi.org/10.5555/3104322.3104425

[119] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. DudÃŋk, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, April 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html

[120] E. HjelmÃěs and B. K. Low, "Face detection: A survey," *Computer Vision and Image Understanding*, vol. 83, no. 3, pp. 236 – 274, September 2001. [Online]. Available: https://doi.org/10.1006/cviu.2001.0921

[121] W. AbdAlmageed, Y. Wu, S. Rawls, S. Harel, T. Hassner, I. Masi, J. Choi, J. Lekust, J. Kim, P. Natarajan, R. Nevatia, and G. Medioni, "Face recognition using deep multi-pose representations," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2016, pp. 1–9.

[122] "Deep convolutional neural networks for pedestrian detection," *Signal Processing: Image Communication*, vol. 47, pp. 482–489, September 2016.

[123] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with large-scale data collection," in *2016 International Symposium on Experimental Robotics*, D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, Eds. Cham: Springer International Publishing, March 2017, pp. 173–184. [Online]. Available: https://doi.org/10.1007/978-3-319-50115-4_16

[124] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring representation learning with cnns for frame-to-frame ego-motion estimation," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 18–25, January 2016. [Online]. Available: https://doi.org/10.1109/LRA.2015.2505717

[125] D. Belter, J. Bednarek, H. Lin, G. Xin, and M. Mistry, "Single-shot foothold selection and constraint evaluation for quadruped locomotion," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 7441–7447.

[126] X. Li, H. Gao, F. Zha, J. Li, Y. Wang, Y. Guo, and X. Wang, "Learning the cost function for foothold selection in a quadruped robot," *Sensors*, vol. 19, no. 6, March 2019. [Online]. Available: https://doi.org/10.3390/s19061292

[127] P. Ferrari, N. Scianca, L. Lanari, and G. Oriolo, "An Integrated Motion Planner/Controller for Humanoid Robots on Uneven Ground," in *2019 18th*

European Control Conference (ECC), June 2019, pp. 1598–1603. [Online]. Available: https://doi.org/10.23919/ECC.2019.8796196

[128] C. Mastalli, I. Havoutis, A. W. Winkler, D. G. Caldwell, and C. Semini, "On-line and on-board planning and perception for quadrupedal locomotion," in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, May 2015, pp. 1–7. [Online]. Available: https://doi.org/10.1109/TePRA.2015.7219685

[129] D. Wahrmann, A. C. Hildebrandt, R. Wittmann, F. Sygulla, D. Rixen, and T. Buschmann, "Fast object approximation for real-time 3D obstacle avoidance with biped robots," in *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, July 2016. [Online]. Available: https://doi.org/10.1109/AIM.2016.7576740

[130] C. Semini, "HyQ – Design and Development of a Hydraulically Actuated Quadruped Robot," Ph.D. dissertation, Istituto Italiano di Tecnologia (IIT) and University of Genova, 2010. [Online]. Available: http://www.semini.ch/downloads/Semini2010PhD_HyQ_Robot.pdf

[131] M. Camurri, "Multisensory State Estimation and Mapping on Dynamic Legged Robots," Ph.D. dissertation, Istituto Italiano di Tecnologia (IIT) and University of Genova, 2017. [Online]. Available: https://iit-dlslab.github.io/papers/camurri17phd.pdf

[132] C. Semini, V. Barasuol, M. Focchi, C. Boelens, M. Emara, S. Casella, O. Villarreal, R. Orsolino, G. Fink, S. Fahmi, G. Medrano-Cerda, and D. G. Caldwell, "Brief introduction to the quadruped robot HyQReal," in *Istituto di Robotica e Macchine Intelligenti (I-RIM)*, September 2019, in press.

[133] "Moog," 2019, accessed on 10.12.2019. [Online]. Available: https://www.moog.com/

[134] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini, *Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality*. Cham: Springer International Publishing, September 2020, pp. 165–209. [Online]. Available: https://doi.org/10.1007/978-3-030-22327-4_9

[135] R. Orsolino, M. Focchi, D. G. Caldwell, and C. Semini, "A combined limit cycle - zero moment point based approach for omni-directional quadrupedal bounding," in *International Conference on Climbing and Walking Robots (CLAWAR)*, 2017.

[136] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[137] 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)*, September 2004, pp. 2149–2154. [Online]. Available: https://doi.org/10.1109/IROS.2004.1389727

[138] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini, "Probabilistic Contact Estimation and Impact Detection for State Estimation of Quadruped Robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1023–1030, April 2017. [Online]. Available: https://doi.org/10.1109/LRA.2017.2652491

[139] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) - The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, February 2016, ch. 5. [Online]. Available: https://doi.org/10.1007/978-3-319-26054-9_5

[140] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," 2010. [Online]. Available: http://eigen.tuxfamily.org

[141] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "RViz: A Toolkit for Real Domain Data Visualization," *Telecommunication Systems*, vol. 60, no. 2, pp. 337–345, October 2015. [Online]. Available: http://dx.doi.org/10.1007/s11235-015-0034-5