DISS. ETH NO. 25726

Optimization-Based Planning and Control for Multi-Limbed Walking Robots

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

C. Dario Bellicoso

MSc in Automation Engineering, Università degli Studi di Napoli, Federico II

born on 15 November 1983

citizen of Italy

accepted on the recommendation of

Prof. Dr. M. Hutter, ETH Zurich
Prof. Dr. B. Siciliano, Università degli Studi di Napoli, Federico II
Prof. Dr. R. Siegwart, ETH Zurich

2019

Robotic Systems Lab
Institute for Robotics and Intelligent Systems
ETH Zurich
Switzerland

To my sister and my parents.

ABSTRACT

Walking robots have unique skills that can be exploited when they are deployed in real-world scenarios. They can locomote over rough terrain, negotiate obstacles, deliver payloads and manipulate the environment, which makes them the ideal solution for search and rescue missions, inspection, and exploration. The implementation of the these capabilities, however, presents theoretical and practical challenges that are yet to be solved: Walking robots have high power requirements and must be able to safely navigate in their designated environment while balancing at all times.

This thesis presents motion planning and control algorithms for robotic legged locomotion and manipulation. The design of these algorithms has the goal of pushing what the current state of the art is capable of in terms of dynamic legged locomotion, as well as coordination with manipulation, aiming at executing these algorithms on torque-controlled robots in real-world environments.

Committing to the implementation of modular software and generic algorithms, our work has focused on the development of motion planning and control algorithms for a torque-controlled quadrupedal robot equipped with a robotic arm. Parts of these have been successfully deployed on other robots, including a wheeled quadrupedal robot and an autonomous excavator.

We present a motion planning that is focused on the generation a wide variety of gaits (i.e., contact sequences), including walking, trotting, pacing, jumping, and galloping. Reduced-complexity models have been used in the planning to ensure dynamic stability, to increase the computational efficiency, and to allow the execution in an online fashion. This allows the planner to react to disturbances and unexpected changes the surrounding environment.

To execute these motions, a whole-body controller has been developed which exploits the full system dynamics to track operational-space motion references. The controller computes torque for each actuator by solving a cascade of tasks, each of which is formulated as a constrained Quadratic Programming problem.

Experimental tests have been carried out on ANYmal, a torque-controlled quadrupedal robot developed by the Robotic Systems Lab at ETH Zurich and ANYbotics. Based on the work presented in this thesis,

ANYmal can execute a wide variety of dynamic gaits and maneuvers, both indoors and on challenging terrain. The success of our approach lies in the use of reduced models for fast and reactive online planning, the use of the full rigid-body dynamics for control, and the integration of lessons learned from field testing.

SOMMARIO

Robot che camminano hanno abilità uniche che possono essere sfruttate quando questi sono utilizzati in scenari reali. Possono camminare su terreno complesso, evitare ostacoli, consegnare un carico e manipolare l'ambiente, rendendo questo tipo di robot la soluzione ideale per missioni di ricerca e recupero, ispezione, ed esplorazione. L'implementazione di queste capacità, tuttavia, pone delle sfide sia teoriche che pratiche che devono ancora essere risolte: robot che camminano non solo richiedono elevata potenza, ma devono anche essere in grado di navigare in sicurezza nell'ambiente designato mantenendo l'equilibrio in ogni momento.

Questa tesi presenta algoritmi di pianificazione del moto e di controllo per locomozione e manipolazione robotica. Lo scopo di questi algoritmi è quello di spingere lo stato dell'arte della locomozione robotica, così come la coordinazione di quest'ultima con la manipolazione, con l'obiettivo di eseguirli su robot controllati in coppia in ambienti reali.

Il nostro lavoro è focalizzato sullo sviluppo di algoritmi di pianificazione e controllo per un robot quadrupede equipaggiato con un manipolatore robotico. Parti del nostro software sono state utilizzate con successo su altri robot, incluso un robot quadrupede con ruote ed uno scavatore autonomo.

Presentiamo un software di pianificazione che è in grado di generare una vasta gamma di modalità di locomozione, tra cui cammino statico, trotto, passo, salto, e galoppo. Modelli a complessità ridotta sono stati usati nella pianificazione del moto per assicurare la stabilità del sistema, per aumentare l'efficienza computazionale, e per permettere la rapida esecuzione dell'architettura di pianificazione. In questo modo, il robot puó reagire a disturbi esterni e a cambiamenti inaspettati nell'ambiente circostante.

Per eseguire questo tipo di movimenti, è stato sviluppato un controllore che sfrutta l'intera dinamica del sistema per tracciare riferimenti nello spazio operativo. Il controllore calcola riferimenti di coppia per ciascun attuatore risolvendo una sequenza di task, ognuno dei quali è formulato come un problema di minimizzazione quadratico.

Risultati sperimentali sono stati eseguiti su ANYmal, un robot quadrupede controllato in coppia e sviluppato al Robotic Systems Lab dell'ETH Zurich e da ANYbotics. Grazie al lavoro presentato in questa tesi, ANYmal può eseguire una vasta gamma di movimenti dinamici, sia in ambienti interni che su terreno complesso. Il successo del nostro approccio

risiede nell'uso di modelli a complessità ridotta per pianificazione veloce e reattiva, nell'uso dell'intera dinamica del sistema per il controllo, e nell'integrazione delle esperienze acquisite durante test sul campo.

ACKNOWLEDGEMENTS

Modern man thinks he loses something - time - when he does not do things quickly. Yet he does not know what to do with the time he gains, except kill it.
— Erich Fromm, "The Art of Loving"

Foremost, I wish to thank my advisers Prof. Marco Hutter, Prof. Bruno Siciliano and Prof. Roland Siegwart, for their constant support and crucial guidance during this research work.

The legged robotics group was initially a small group of people driven by the passion to show the world what we would be capable of. Among these, a close collaboration and friendship started the very first day I arrived in Switzerland: Christian Gehring and Péter Fankhauser. They have been there for me as guides, teachers, and, most importantly, friends. To this handful of people, more joined along the way. Marko Bjelonic has shown me what true talent, commitment and friendship mean. Over the years, this group was completed by Gabriel, Vassilis, Jemin and Klajd. All of them have never failed to be there for me and I am grateful to them for their support.

I wish to thank all the very talented people at the Robotic Systems Lab, Autonomous Systems Lab, and ANYbotics for their great contribution to this research. A big thank you to Maria for her irreplaceable support, and to Selen for providing some of the graphics used in this work.

I have supervised many students at ETH. Of these, I would like to thank Fabian Jenelten and Koen Krämer, who later joined our team. Without them, most of what has been demonstrated in this dissertation would have not been possible. I hope they have learned from me at least as much as I have learned from them.

In Zürich I have managed to create my own patch of friends who whave supported me: Danai, Georgia, Spyros, Eleni, Efi, Konstantinos, Ioanna, Blerina, Herald. Although at a distance, the friends of a lifetime have been there from Napoli: Elia, Peppe, Dimitri, Claudio, Geky, Gianfranco, Francesca, Isaura, Valentina, Floriana, Carlo, Marco, Diego, Fabio, Stefania, Elena, Francesco, Vincenzo, Jonathan, Luca.

Finally, a special thank you goes to my family: Gaja, Antonio, Antonella.

Zürich, December 2018.
Dario

Financial Support
Grateful acknowledgments go to the Swiss National Science Foundation (SNF) and the National Centre of Competence in Research Robotics.

This work has been conducted as part of ANYmal Research, a community to advance legged robotics.

CONTENTS

1

INTRODUCTION

The aggregate of our joy and suffering, thousands of confident religions, ideologies, and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilization, every king and peasant, every young couple in love, every mother and father, hopeful child, inventor and explorer, every teacher of morals, every corrupt politician, every "superstar," every "supreme leader," every saint and sinner in the history of our species lived there–on a mote of dust suspended in a sunbeam.

— Carl Sagan, "A Pale Blue Dot"

1.1 MOTIVATION AND OBJECTIVES

Since the first industrial revolution, automation has changed the way society evolves and adapts. Through technological advances, mankind's new companion slowly started to become a practical reality: robots. First coined by the science-fiction author Karel Čapek in 1921 in his book *Rossum's Universal Robots* [1], the word "robot" (*robota* in Czech) refers to forced labour. The concept was later developed by the author Isaac Asimov, who famously introduced the three laws of robotics in his novel *Runaround* appearing in the collection *Robot Visions* [2] (see the book cover in Figure 1.1). In his books, robots are often compared to humans. Although constrained by the programming of their "positronic brains", they can think like humans and act like them.

One day, robots like the ones dreamed of by Asimov will be our daily companions. They will aid us performing dangerous tasks, explore remote areas both on Earth and on other planets, and save lives of humans. Robots will improve our life in ways that are unimaginable today. Of the many kind of robots that have imagined by story tellers, some of them have one trait that connects us deeply with them: they are shaped like humans. They have legs, arms, hands, can walk, jump, run, and manipulate the environment. They can make decisions and use their limbs to intuitively interact with their surroundings, just like humans do. Among the different

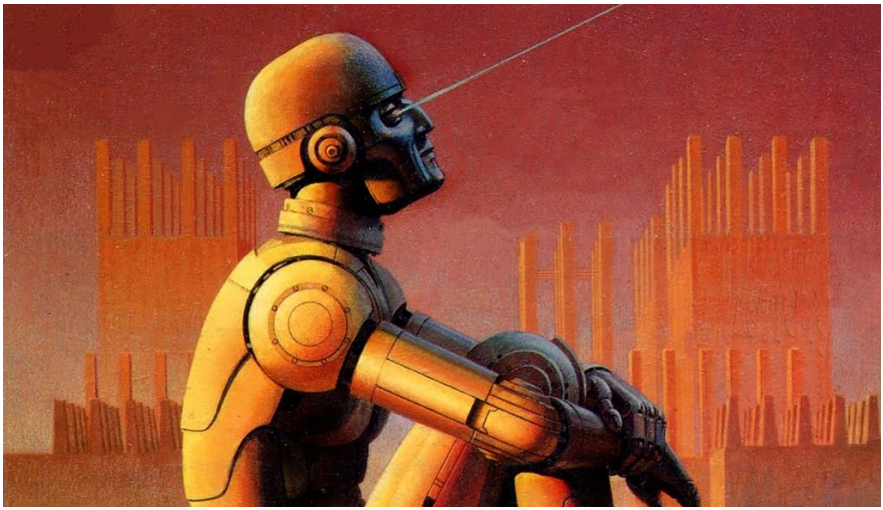


FIGURE 1.1: Isaac Asimov, in his novel "Runaround" appearing in the collection "Robot Visions", introduced for the first time the famous three laws of robotics.

kinds of robots, legged ones are the ideal machines to be deployed in a world where most of the environment is rough and hard to traverse. In such an environment, legged machines have a clear advantage in terms of mobility and obstacle negotiation skills that traditional robots, such as wheeled and tracked ones, do not have.

Legged and in general articulated systems pose, however, greater challenges with respect to other robotic systems. They interact with the environment through several contact points to manipulate the environment or to move in it, have high power requirements, and must maintain balance at all times, while withstanding pushes, slippage and unexpected changes in the terrain. Walking robots that must navigate through a real-world environment and interact with it require powerful computational capabilities.

The day we will see this kind of systems becomes reality is still faraway. Nevertheless, the recent advances in computational power, manufacturing capabilities, and understanding of locomotion dynamics have enabled us to produce impressive results which takes us one step closer to the reality described in science fiction. Today, we see walking robots taking their first steps outside the laboratories and into the real world. Researchers have designed systems which have shifted away from the position control paradigm to compliant force control. The state of the art in motion planning and control techniques, together with the greater computational capa-

bility which is now available, has been pushing the boundaries of what was considered impractical until recent years. Techniques such as Trajectory Optimization (TO), Model Predictive Control (MPC), and Reinforcement Learning based algorithms, have demonstrated impressive results on real systems. All of these approaches, however, require proper understanding of locomotion principles and interaction with the environment when implementing them on a real system.

The goal of this thesis is to design motion planning and control algorithms, aiming at bringing articulated robots out of a lab and operate on rough terrain, and to aid us in search and rescue missions and in inspection tasks. Our approach, tested on a torque-controlled quadrupedal robot which has been recently equipped with a six-degrees of freedom (DOF) robotic arm, has been designed with modularity and generality in mind. The developed planning and control tools are very generic and have been used to control wheel-based systems and an autonomous excavator. We build on a TO methods to design a fast and efficient receding horizon motion planner which is capable of quickly reacting to external disturbances and adapting to unseen changes in the terrain. The generated motions are tracked by a whole-body controller (WBC) which solves a set of prioritized optimization problems to generate torque for the whole body. Thanks to the tight integration of these methods with a modular software framework, we have successfully demonstrated dynamic locomotion behaviours such as walking, running, jumping, galloping, as well as manipulation skills such as object grasping and door opening.

1.2 STATE OF THE ART

In the following, we provide an overview of the history of the development of walking robots, as well a description of the planning and control approaches that are relevant to the work presented in this thesis.

1.2.1 *Legged robots: an overview of history and research*

Mankind has been trying to develop mechanical devices which could provide aid to execute tasks for centuries. These systems were mainly based on wheels due to their simpler design. One of the first mechanical developments was a four-legged walking truck from General Electric [3]. This machine was 3.3 m high, 3 m long, weighed 1400 kg, and had four legs with three hydraulically driven DOF each. All joints were directly controlled by

a human operator which was sitting inside the machine. This kind of interface was very complex to use, and emphasized the need of an automated solution.

It took until the early 1980s to see some of the first mechatronic legged solutions. The MIT Leg Laboratory, led by Marc Raibert, pioneered the design of dynamically balancing legged systems. In 1986, the group developed a planar one-legged hopping robot [4] which demonstrated a modular control design for hopping motions. This was later extended to a 3D one-legged hopper [5] and to a 3D biped [6] capable of executing a somersault maneuver. During this time, the group also developed a quadrupedal robot [7] capable of trotting, pacing, bounding, and switching between these gaits.

In Japan, the first bipedal robot to demonstrate dynamic walking was *WL-10RD* [8]. Balance was obtained by appropriately planning the Zero-Moment Point (ZMP) [9]. This twelve DOF robot was 1.43 m high, 84.5 kg heavy, and driven by hydraulic actuators. In 2000, Honda Motor Co. presented *ASIMO* [10], a 26 DOF humanoid robot. ASIMO demonstrated walking and running by controlling the ZMP. An upgraded version was unveiled in 2011.

In more recent years, thanks to the advances in computation capabilities and mechanical design, several research groups and companies started to propose new solutions for walking robots. The Dynamic Legged Systems group at the Istituto Italiano di Techonologia (IIT), led by Claudio Semini, developed *HyQ* [11], a hydraulically actuated quadrupedal robot. HyQ has three torque-controlled DOF per leg and weighs 75 kg. In [12], the group has demonstrated a control framework for a crawling gait which is capable of walking on slopes up to 50°.

The Legged Robotics group at the Autonomous Systems Lab (ASL), ETH Zurich, led by Prof. Roland Siegwart, began development of *StarlETH* [13], an electrically actuated quadrupedal robot driven by series-elastic actuators (SEA). Weighing around 23 kg, StarlETH is capable of walking and trotting on slopes and rough terrain [14]. The group integrated perception for foothold selection by creating a robocentric elevation mapping framework ([15], [16]). StarlETH also demonstrated a wide variety of gaits [17, 18]. The group later evolved into the Robotic Systems Lab (RSL), led by Prof. Marco Hutter, and presented *ANYmal* [19] (see Figure 1.2), an electrically actuated quadrupedal robot. It was used to compete at the Autonomous Robot for Gas & Oil Sites (ARGOS) and European Robotics League (ERL) challenges. ANYmal, a 30 kg torque-controlled robot, was de-

signed for industrial inspection, search and rescue missions, and payload delivery. ANYmal has been shown walking [20], trotting [21, 22], and planning safe footholds using perception [16, 23]. Thanks to an intuitive software interface [24], ANYmal can perform hand-crafted maneuvers while keeping balance, allowing it to climb stairs and negotiate obstacle.

Impressive results have been demonstrated by Boston Dynamics in the last few years. The company has developed walking robots (see Figure 1.3) that are capable of performing highly dynamic gaits. Unfortunately, very little is known about the methods used to execute such remarkable maneuvers.

1.2.2 *Trajectory Optimization*

The motion planning problem for a walking robot is a challenging one, due to the high dimensionality of the state-input space and the combinatorial characteristic of the planning problem for hybrid systems. One of the prominent methods which has gained attention by the research community is TO, which deals with the problem of finding a system trajectory which minimizes a given cost function subject to some constraints. The state of the art in TO has demonstrated impressive results in the kind of motions that can be produced [28, 29] by generating motions for a wide variety of maneuvers (e.g., standing up or climbing a wall for a humanoid robot, or generating a wide variety of gaits for a quadrupedal robot).

Several formulations to solve TO exist. In this dissertation we focus on direct collocation, an approach which approximates the trajectory as a sequence of polynomial splines and finds optimal values for the spline coefficients. The problem is transcribed to a Nonlinear Programming Problem (NLP), formulated as

$$\min_{\boldsymbol{\xi}} f(\boldsymbol{\xi}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{\xi}) = 0 \quad \boldsymbol{h}(\boldsymbol{\xi}) \leq \mathbf{0}, \tag{1.1}$$

where $\boldsymbol{\xi}$ is the vector of optimization variables, $f(\boldsymbol{\xi})$ is the cost function, and $\boldsymbol{g}(\boldsymbol{\xi}), \boldsymbol{h}(\boldsymbol{\xi})$ are equality and inequality constraints respectively. The NLP can be numerically solved by Interior Point [30] or by sequential quadratic programming (SQP) [31] off-the-shelf software, such as IPOPT [32] or SNOPT [33].

FIGURE 1.2: The robot *ANYmal* (top), developed by RSL and ANYbotics, is a torque-controlled quadrupedal robot for search and rescue missions and outdoor inspection. RSL has recently developed two variants of ANYmal: a wheeled version (bottom left) for efficient driving [25] and one equipped with a manipulator (bottom right) for manipulation tasks [26] (presented in this thesis).

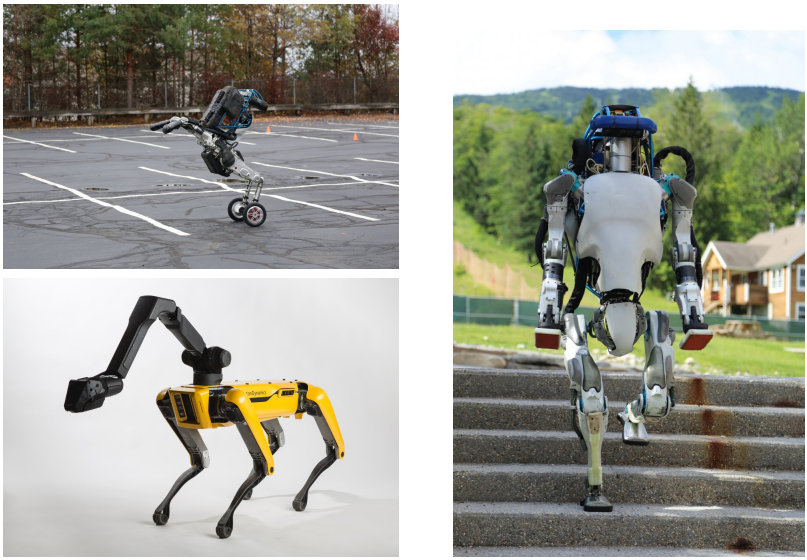


FIGURE 1.3: Some of the most recent robots developed at Boston Dynamics (pictures accessed from [27]), the humanoid *Atlas* (right) and the quadruped *SpotMini* (bottom left). Both have demonstrated impressive capabilities in terms of locomotion and manipulation. Recently, the company has unveiled *Handle* (top left), a robot which is capable of driving, jumping, and executing manipulation tasks.

1.2.3 *Whole-Body Control and Hierarchical Optimization*

An articulated robot (Figure 1.4) that interacts with the environment through multiple contacts is more challenging to control than a freely moving robot. The complexity of locomotion and manipulation can, however, be broken down into smaller (and simpler to define) tasks, which should not interfere with each other. For example, motion tracking should be performed as best as possible while not interfering with constraints imposed by the equations of motion. Contact forces should be planned such that they compensate for the robot's weight and propel it in the environment while the friction model is respected..

For these reasons, researchers has focused on *whole-body control*, a control framework which represents each sub-task as a set of constraints which must be fulfilled, typically in a prioritized manner. A WBC typically solves the inverse dynamics problem by exploiting the full system dynamics, and uses the hierarchical optimization (HO) framework to solve a set of prioritized tasks.

The first approaches to hierarchical optimization in robotics are described in [34] and [35]. The approach was later generalized to an arbitrary number of tasks by [36]. These methods all presented a sequence of tasks which defined equality constraints only, in which the prioritization is solved by computing the projection into the nullspace of the constraints of higher priority tasks. In [37], a framework which also includes inequality constraints is presented. In that approach, each level of the hierarchy is formulated as a quadratic programming (QP) problem with soft equality constraints. The inequality constraints for each QP are stacked together from the top level priority task. The disadvantage of this formulation is the increase in number of constraints when solving lower priority tasks. An alternative approach was presented in [38], although it required the first task to not have any inequality constraints. This was solved in [39], where balancing and momentum control experiments were performed on a two legged system. In [40], a novel algorithm dedicated to solve a hierarchy of tasks is presented which is ten times faster than previous methods.

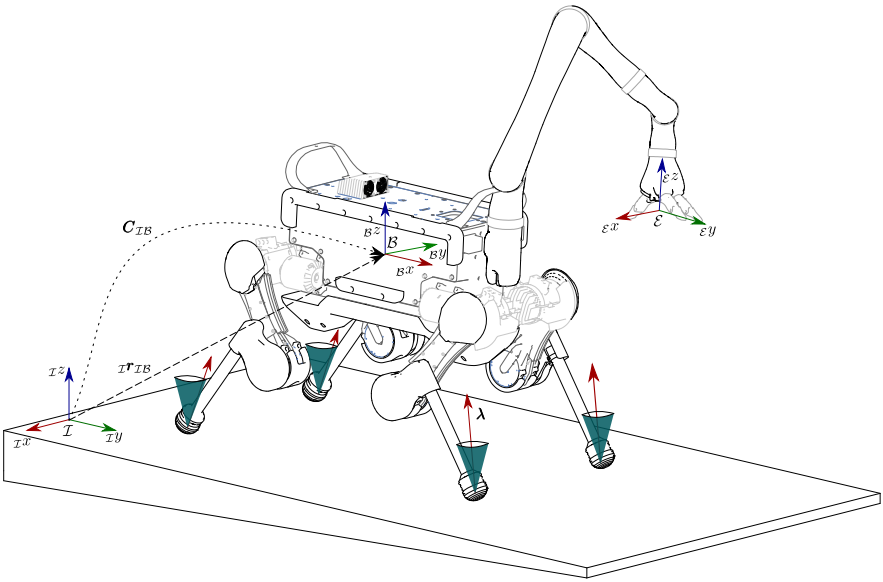


FIGURE 1.4: A sketch of a multi-limbed floating base system. The position ${}_I r_{IE}$ and orientation C_{IE} of the floating base $\mathcal{B}$ are estimated by a state estimator. The reference ground reaction forces λ, constrained to lie within the friction cones, drive the motion of the floating base and are computed by a motion tracking algorithm based on the desired motion. The motion of the end-effector frame $\mathcal{E}$ is generated based on the task to be performed.

1.2.4 *Stability criteria for walking systems*

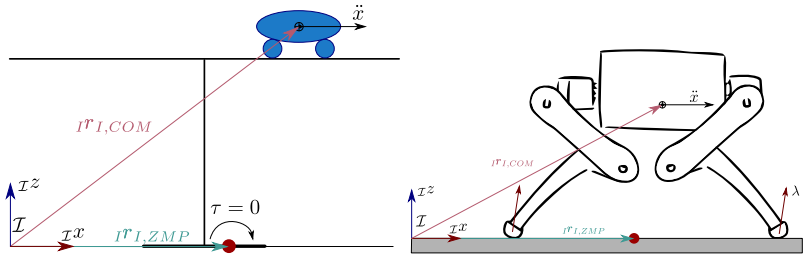
When designing motions for a walking system, a stability criterion is typically employed to ensure balance[1]. In abscence of motion, a legged system is stable when its center of mass (COM), projected along the direction of gravity, falls into the support polygon, defined as the convex hull of the contact points. This is a static stability criterion that has been used for slow walking gaits [41]. For fast walking gaits, or for ones where the support polygon degenerates to a line or to a point, a dynamic stability criterion can be employed that takes into account not only the location of the COM, but also its instantaneous motion. This led to the analysis of the ZMP (see Figure 1.5), a concept first presented in [42]. In [43], an analysis of the external forces acting on a bipedal robot is given. There, the ZMP is defined as "*the point on the ground where the tipping moment acting on the biped, due to gravity and inertia forces, equals zero, the tipping moment being defined as the component of the moment that is tangential to the supporting surface*" [43]. This concept has proven to be very important in bipedal [8, 10] and quadrupedal [44] locomotion and has proven its success over the years [9]. The ZMP represents a reduced model of the dynamics of a walking system, allowing the implementation of faster trajectory optimization methods compared to other models, e.g., centroidal dynamics [45], which produce motions that can be tracked by motion controllers which exploit the full system dynamics.

1.3 CONTRIBUTIONS

In this thesis, we demonstrate the research that has been done at RSL on quadrupedal locomotion and mobile manipulation. A modular motion planner has been developed, which splits the problem of planning whole-body motion references into smaller sub-modules: a floating base planner, a foothold generator, and a swing trajectory planner. These are executed online in a receding horizon fashion, meaning that a motion plan is executed as soon as it is available while a new one is computed in parallel for each of these modules.

The motion plans are tracked by a whole-body controller based on hierarchical optimization. The controller tracks operational space references

1 The term *stability* here is not used in a Lyapunov sense (as is typically done in classical control system design), but refers to the generation of motions that, when tracked, allow a walking system to maintain balance.



(a) Cart-Table model (b) ZMP model for floating base systems

FIGURE 1.5: The ZMP is the point where the resulting ground reaction forces produce zero moment. When the ZMP is located in the convex hull of the contact points, the system is said to be dynamically stable (a). The same concept can be applied to walking systems (b) as a constraint for the generation of motion plans.

while guaranteeing dynamic feasibility of the solution, and constraining the reaction forces and the actuation torques.

The framework has been extensively tested on ANYmal in rough and challenging environments, allowing the robot to react to disturbances and to recover from challenging situations, proving the success of our approach.

1.4 THESIS ORGANIZATION

This introduction is followed by a statement of the contributions presented in this dissertation in Chapter 2. A brief summary of the motivations and contributions of the articles that are related to this dissertation in Chapter 3. This is followed by the the full articles, cumulatively presented in Chapters 5 through 9.

2

CONTRIBUTIONS

This chapter introduces the publications included in this thesis, presenting a brief overview on the motivation on each on them, as well as their contribution.

2.1 RELEVANT PUBLICATIONS

2.1.1 *Paper I*

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

2.1.1.1 *Motivation*

Controlling a walking robot is a very challenging task. The robot must track a reference motion while trying to keep balance, constraint the ground reaction forces to avoid slippage, and constrain the motion of its end-effectors to avoid kinematic limits. The whole-body control approach coupled with hierarchical optimization, allows to break down this complex behaviour into smaller tasks which can be prioritized one with respect to another, such that the solution of a higher priority task does not get affected by the lower priority ones. This approach can be used to blindly overcome rough terrain. By appropriately designing the hierarchy of tasks, we are able to walk down a 15cm step with no additional motion planning requirements.

2.1.1.2 *Contribution*

This paper constitutes the foundation for the control part of this entire thesis. We present how a whole-body control and hierarchical optimiza-

tion approach can be used to improve blind locomotion on a quadrupedal robot. Moreover, we design a whole-body control and hierarchical optimization library which has been successfully been employed both indoors and outdoors on challenging terrain.

2.1.2 *Paper II*

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

2.1.2.1 *Motivation*

Trajectory optimization approaches allow to generate a wide variety of motion plans for floating base systems. These methods, however, typically require high computation capabilities and long optimization times, which can hinder fast reaction and motion regeneration in case of disturbances on the real robot. In this paper, we design a ZMP-based approach which demonstrates how to generate motion plans for walking, dynamic walking and pacing on a quadrupedal robot. The motion plans are tracked by a whole-body controller. We demonstrate how to execute these gaits, as well as how to smoothly transit through them.

2.1.2.2 *Contribution*

We demonstrate how to generate dynamic gaits using a reduced dynamics model which allows to generate motion plans online which can enable the robot to quickly react to disturbances and unexpected variations in the environment.

2.1.3 *Paper III*

C. D. Bellicoso, F. Jenelten, C. Gehring and M. Hutter. *Dynamic Locomotion Through Online Nonlinear Motion Optimization for Quadrupedal Robots*. In IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2261-2268, July 2018.

2.1.3.1 *Motivation*

In this paper we extend the previous results of our online motion planner by generating motions for the full translational components of the COM of the whole system. This allows to execute a new set of gaits, including running trot and pronking. The planner implements a receding-horizon problem that is efficiently solved online.

2.1.3.2 *Contribution*

We demonstrate how to execute a wider variety of walking gaits with respect to our previous work and execute transitions through them.

2.1.4 *Paper IV*

C. Dario Bellicoso, Koen Krämer, Markus Stäuble, Dhionis Sako, Fabian Jenelten, Marko Bjelonic, Marco Hutter. *ALMA - Articulated Locomotion and Manipulation for a Torque-Controllable Robot*. In 2019 IEEE International Conference on Robotics and Automation (ICRA), Montreal, Canada, 2019.

2.1.4.1 *Motivation*

A walking robot interacts with the environment by making contact with it. When equipped with a robotic arm, the system can interact with its surroundings in very different ways: it can open doors, it can grasp and manipulate objects.

2.1.4.2 *Contribution*

We demonstrate how to implement a whole-body controller for a quadrupedal robot equipped with a six DOF robotic arm and how to handle the high redundancy of the whole system. We demonstrate tasks which include opening doors, grasping objects, and reacting to external disturbances to keep balance.

2.1.5 *Paper V*

C. Dario Bellicoso, Fabian Jenelten, Koen Krämer, Thomas Lew, Marko Bjelonic, Marco Hutter. *Motion Generation for Dynamic Locomotion and Manipulation with Whole-Body Control for Multi-Limb Robots*. Submitted to 2019 International Journal on Robotics Research (IJRR).

2.1.5.1 *Motivation*

In our previous works, we have shown the success of our methods on planning and control for multi-limb systems by enabling ANYmal, equipped with a robotic arm, to react to external disturbances, perform a wide variety of gaits, and execute manipulation tasks. To robustly execute highly dynamic gaits however, such as galloping, more knowledge of the system dynamics in the motion plan is required.

2.1.5.2 *Contribution*

We present a unified approach to planning and control for a walking robot equipped with a robotic arm. Thanks to our recent work, the software framework enable ANYmal to execute a wide variety of gaits, including highly dynamic ones like galloping, open doors when equipped with a robotic manipulator, and robustly trot in a real-world terrain. Our previous works are extended by including planning of the orientation of the torso, as well as a detailed description of the unified framework.

2.1.6 *Paper VI*

C. Dario Bellicoso, Marko Bjelonic, Lorenz Wellhausen, Kai Holtmann, Fabian Günther, Marco Tranzatto, Péter Fankhauser, Marco Hutter. *Advances in Real-World Applications for Legged Robots*. In 2018 Journal of Field Robotics (JFR), vol. 35, issue 8, pp. 1311–1326, November 2018.

2.1.6.1 *Motivation*

Legged robots represent an ideal solution as machines to be deployed in real-world environments for search and rescue missions or inspection tasks. The complexity of these machines, however, requires careful design of the system in terms of sensor choices to execute a mission, and locomotion strategies to navigate in outdoor terrain.

2.1.6.2 *Contribution*

We present the system design choices and the planning and control framework which has enable ANYmal, a torque-controlled quadrupedal robot, to execute search and rescue as well as inspection tasks. The article shows the lessons learned from the ARGOS and ERL challenges.

3

CONCLUSION AND OUTLOOK

After changes upon changes, we are more or less the same.

— Simon and Garfunkel, "The Boxer"

This chapter summarizes the main results and outlines prospective research directions.

3.1 MOTION PLANNING

We have proposed a novel design of an optimization-based motion planning algorithm based on a reduced dynamics model for fast and efficient online generation of whole-body motion plans for a quadrupedal robot, later adapted to the handle the presence of an additional limb for manipulation. The framework generates plans in a receding horizon fashion, producing a whole-body motion reference which quickly adapts to unexpected external disturbances and unseen variations in the terrain. Motion planning is separated into modules, each of which runs in its own computational unit without interrupting the main control loop. Our effort to reduce computation times by allowing the user to specify which DOF to include in the COM motion planning has led to a framework that allowed ANYmal, a torque-controlled quadrupedal robot, to execute a wide variety of gaits, including walking, trotting, running, jumping, and galloping, while being pushed and in real-world environments.

3.2 CONTROL

To track the operational space motion references, a whole-body controller algorithm has been implemented which uses hierarchical optimization to solve a cascade of tasks. These include dynamic feasibility (i.e., constraining the optimized generalized accelerations and ground reaction forces to lie in the manifold defined by the system dynamics), torque limits, and friction cone constraints. The framework has been initially evaluated by designing an appropriate hierarchy of tasks which enabled ANYmal to blindly execute a static walk while negotiating a downward step of 15 cm.

The controller has been developed into a software library which has successfully been generalized to allow it to be implemented on a wheeled version of ANYmal and on an autonomous excavator. The software has been thoroughly tested and evaluated of ANYmal through challenges, events, simulated search and rescue missions, and inspection tasks.

3.3 FUTURE DIRECTIONS

The work presented in this dissertation is focused on robustifying blind locomotion and coordination of the latter with manipulation. By adding perception to the motion planner and to the controller, safe footholds could be generated that avoid irregularities in the terrain; vision-based control can be used to enable autonomous manipulation of objects, e.g., grasping a door handle to open doors.

So far, pre-scpecified gait timings and contact sequences have been provided to the planner, with the goal of reducing the overall complexity and the computation time. This limitation should be addressed in future work by including these quantities in the optimization problem. The increase in complexity, however, might lead to performance issues on the real machine. This problem can be mitigated by considering models at decreasing level of detail for different time intervals of the optimization horizon.

The work done so far on the control of multi-limbed robots has focused on a single machine. Future work should solve the problem of the coordination of multiple walking agents that interact to accomplish a goal, such as collaboratively carrying an object.

Reinforcement Learning techniques can be integrated into the optimization-based and model-based algorithms presented here. Thanks to the modular design of the software framework discussed in this research, the substitution of one of these, e.g., the foothold planner, with a learning-based method would allow the evaluation and comparison of the different approaches.

4

PAPER 1: PERCEPTION-LESS TERRAIN ADAPTATION THROUGH WHOLE BODY CONTROL AND HIERARCHICAL OPTIMIZATION

4.1 ABSTRACT

This paper presents a control approach based on a whole body control framework combined with hierarchical optimization. Locomotion is formulated as multiple tasks (e.g. maintaining balance or tracking a desired motion of one of the limbs) which are solved in a prioritized way using QP solvers. It is shown how complex locomotion behaviors can purely emerge from robot-specific inequality tasks (i.e. torque or reaching limits) together with the optimization of balance and system manipulability. Without any specific motion planning, this prioritized task optimization leads to a natural adaption of the robot to the terrain while walking and hence enables blind locomotion over rough grounds. The presented framework is implemented and successfully tested on ANYmal, a torque controllable quadrupedal robot. It enables the machine to walk while accounting for slippage and torque limitation constraints, and even step down from an unperceived 14 cm obstacle. Thereby, ANYmal exploits the maximum reach of the limbs and automatically adapts the body posture and height.

4.2 INTRODUCTION

Motion planning and control of legged robots is coupled with several challenges. First, legged robots are typically high-dimensional systems with partially redundant degrees of freedom. Second, interaction with the environment is achieved through multiple contact points which impose changing contact constraints and interaction forces. Finally, the environment itself is generally unknown and terrain perception can be limited or absent altogether. To lower the design complexity of motion and control algorithms, locomotion is often described as a set of simpler tasks such as body posture control to keep balance, limb motion to move, or contact force constraints to prevent slippage. A theoretical framework which accommodates this kind of decomposition is the Whole Body Control (WBC) framework

FIGURE 4.1: ANYmal is a fully torque-controllable quadrupedal robot, actuated by custom developed series elastic actuators which provide very precise torque control.

coupled with Hierarchical Optimization (HO). Generally speaking, any implementation that generates control signals for all the actuated joints of a robotic system may be called a WBC; however, literature has focused on WBC which implement, at least, floating base inverse dynamics [46]. Tasks can be either solved at the same priority through weighted average or can be prioritized one w.r.t. the other [13].

The first implementations of HO were dealing exclusively with equality constraint tasks (e.g. motion tracking objectives) [36]. Inequality constraints, such as torque or joint limits and friction cone bounds, were implemented by means of potential fields [47], [48]. The first proper attempt at explicitly taking into account inequality constraints has been shown only in recent years [49]. The set of tasks were solved as a cascade of Quadratic Programming (QP) problems constrained by both equalities and inequalities. The solution of each QP, together with its constraints, would shape the space in which to search for the next solution. However, computational requirements were very high, and could become unfeasible to implement on a real time system. Successive work has shown how this issue can be tackled (e.g. [50]), by combining explicit inequality constraint han-

dling with the search of solutions in the null space of higher priority tasks. This method yields QP problems which get smaller as equality constraints are added to the cascade of QPs.

In the present paper, we show how such a control approach can be exploited to create adaptive behaviors for locomotion over challenging terrain. Instead of integrating perception and extending motion planning as a function of the environment, we present clever control tasks of different priorities to exploit the large degree of mobility of legged systems and automatically adapt to (uneven and unknown) ground.

Hence, our contribution is twofold: first, we show a concrete implementation of WBC using HO coupled with the full system dynamics and motion planning for a walking behavior on a real system; second, we also show how inequality constraint tasks in the operational space can be used to achieve reactive and perception-less adaptation to the terrain. The proposed control framework was successfully tested on the torque controllable quadrupedal robot ANYmal (Fig. 4.1).

4.3 MODEL FORMULATION

In general, a walking robot can be modeled as a system with a free-floating base B to which legs and arms are attached. The motion of the entire system can be described w.r.t. a fixed inertial frame I. We write the position vector of the base frame w.r.t. the inertial frame as ${}_I\boldsymbol{r}_{IB} \in \mathbb{R}^3$ and use unit quaternions to parametrize the orientation of the main body. The limb joint generalized positions are stacked in the vector $\boldsymbol{q}_j \in \mathbb{R}^{n_j}$. We write the generalized positions vector $\boldsymbol{q}$ and the generalized velocities vector $\boldsymbol{u}$ as

$$\boldsymbol{q} = \begin{bmatrix} {}_I\boldsymbol{r}_{IB} \\ \boldsymbol{q}_{IB} \\ \boldsymbol{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \quad \boldsymbol{u} = \begin{bmatrix} {}_I\boldsymbol{v}_{IB} \\ {}_B\boldsymbol{\omega}_{IB} \\ \dot{\boldsymbol{q}}_j \end{bmatrix} \in \mathbb{R}^{n_u}, \tag{4.1}$$

where $n_u = 6 + n_j$, ${}_B\boldsymbol{\omega}_{IB}$ is the angular velocity of the base w.r.t. the inertial frame expressed in B frame, and $\boldsymbol{q}_{IB} \in SO(3)$ is the unit quaternion that projects the components of a vector expressed in B frame to those of the same vector expressed in I frame. The equations of motion of legged systems can be written as [51]

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}_s^T\boldsymbol{\lambda}, \tag{4.2}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n_u \times n_u}$ is the mass matrix and $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) \in \mathbb{R}^{n_u}$ is the vector of Coriolis, centrifugal and gravity terms. The selection matrix $\boldsymbol{S} =$

$\begin{bmatrix}\mathbf{0}_{n_\tau \times (n_u - n_\tau)} & \mathbb{I}_{n_\tau \times n_\tau}\end{bmatrix}$ selects which DoFs are actuated. If all limb joints are actuated, then $n_\tau = n_j$. The vector of constraint forces $\boldsymbol{\lambda}$ is mapped to the joint space torques through the support Jacobian $\boldsymbol{J}_s \in \mathbb{R}^{3n_c \times n_u}$, which is obtained by stacking the constraint Jacobians as $\boldsymbol{J}_s = \begin{bmatrix}\boldsymbol{J}_{C_1}^T & \cdots & \boldsymbol{J}_{C_{n_c}}^T\end{bmatrix}^T$, with n_c the number of limbs in contact. Motion at the support contact points must be constrained. If the feet are modeled as point contacts, then each contact constraint introduces three equations ${}_I\boldsymbol{r}_{IC}(t) = const$, which can be differentiated twice to yield

$$\begin{aligned} {}_I\dot{\boldsymbol{r}}_{IC} &= \boldsymbol{J}_C \boldsymbol{u} = \mathbf{0} \\ {}_I\ddot{\boldsymbol{r}}_{IC} &= \boldsymbol{J}_C \dot{\boldsymbol{u}} + \dot{\boldsymbol{J}}_C \boldsymbol{u} = \mathbf{0}. \end{aligned} \tag{4.3}$$

4.3.1 *Support-consistent dynamics*

As shown in [46], if the motion of the whole body remains in the null space of the contact constraints described in (4.3), then the equations of motion (4.2) can be projected to a reduced space which still fully describes the system dynamics. Assuming that $\boldsymbol{J}_s^T$ has full column rank $\rho(\boldsymbol{J}_s^T) = 3n_c$, we can employ the *QR decomposition* to decompose the support Jacobian

$$\boldsymbol{J}_s^T = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{R} \\ \mathbf{0}_{(n_u - 3n_c) \times 3n_c} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_c & \boldsymbol{Q}_u \end{bmatrix} \begin{bmatrix} \boldsymbol{R} \\ \mathbf{0}_{(n_u - 3n_c) \times 3n_c} \end{bmatrix}, \tag{4.4}$$

where $\boldsymbol{R} \in \mathbb{R}^{3n_c \times 3n_c}$, $\boldsymbol{Q}_c = \boldsymbol{Q}\boldsymbol{S}_c \in \mathbb{R}^{n_u \times 3n_c}$ and $\boldsymbol{Q}_u = \boldsymbol{Q}\boldsymbol{S}_u \in \mathbb{R}^{n_u \times (n_u - 3n_c)}$, with

$$\boldsymbol{S}_c = \begin{bmatrix} \mathbb{I}_{3n_c \times 3n_c} \\ \mathbf{0}_{(n_u - 3n_c) \times 3n_c} \end{bmatrix}, \quad \boldsymbol{S}_u = \begin{bmatrix} \mathbf{0}_{3n_c \times (n_u - 3n_c)} \\ \mathbb{I}_{(n_u - 3n_c) \times (n_u - 3n_c)} \end{bmatrix}. \tag{4.5}$$

By left multiplying eq.(4.2) by $\boldsymbol{Q}^T$, we can project the equations of motion to the contact constrained and unconstrained spaces

$$\boldsymbol{Q}_c^T [\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u})] = \boldsymbol{Q}_c^T \boldsymbol{S}^T \boldsymbol{\tau} + \boldsymbol{R}\boldsymbol{\lambda} \tag{4.6a}$$

$$\boldsymbol{Q}_u^T [\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u})] = \boldsymbol{Q}_u^T \boldsymbol{S}^T \boldsymbol{\tau}, \tag{4.6b}$$

As long as the motion of the system is in the null space of the contact constraints (4.3), eq.(4.6b) describes the dynamics of the entire system. Note that the constraint forces $\boldsymbol{\lambda}$ do not appear in eq.(4.6b). These can be reconstructed from eq.(4.6a) by writing

$$\boldsymbol{\lambda} = \boldsymbol{R}^{-1} \boldsymbol{Q}_C^T \left[\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) - \boldsymbol{S}^T \boldsymbol{\tau} \right]. \tag{4.7}$$

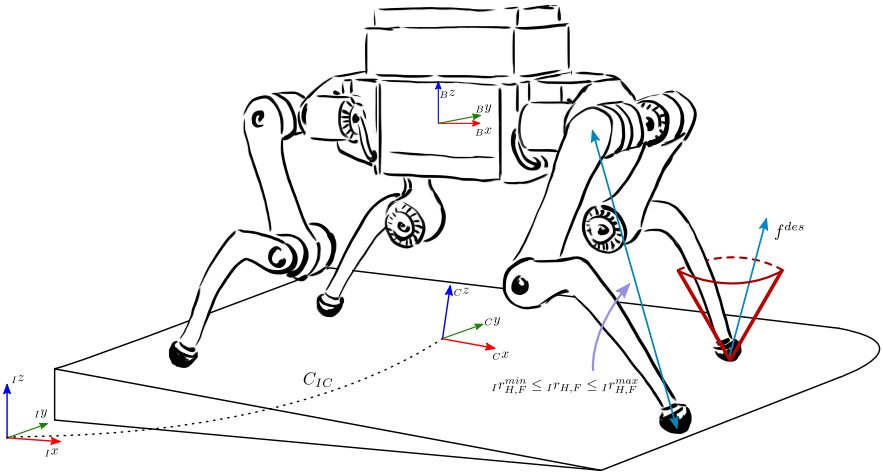


FIGURE 4.2: Solving the locomotion problem requires to solve several sub-tasks at the same time. These typically include solving for the equations of motion, limiting the resulting contact forces to lie within the friction cone, track a desired motion.

4.4 HIERARCHICAL OPTIMIZATION

A task $\boldsymbol{T}$ can be defined as a set of linear equality and/or inequality constraints on the solution vector $\boldsymbol{x} \in \mathbb{R}^n$:

$$\boldsymbol{T} : \begin{cases} \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{w} \\ \boldsymbol{D}\boldsymbol{x} - \boldsymbol{f} \leq \boldsymbol{v} \end{cases}, \tag{4.8}$$

where $\boldsymbol{w}$ and $\boldsymbol{v}$ are slack variables to be minimized. A set of tasks $\boldsymbol{T}_1, \ldots, \boldsymbol{T}_p$ can be either solved at the same time, optionally weighted against each other, or in a strict prioritized order. Solving p tasks yields an optimal solution $\boldsymbol{x}^*$. As shown in [50], to ensure strict prioritization the next solution $\boldsymbol{x}_{p+1}$ can be found in the null space of all higher priority equality constraints $\boldsymbol{Z}_p = \mathcal{N}\left(\underline{\boldsymbol{A}}_p\right)$, with $\underline{\boldsymbol{A}}_p = \begin{bmatrix} \boldsymbol{A}_1^T & \ldots & \boldsymbol{A}_p^T \end{bmatrix}^T$, of the higher priority constraints, yielding $\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{Z}_p \boldsymbol{z}_{p+1}$, where $\boldsymbol{z}_{p+1}$ is a vector which lives

in the row space of $\boldsymbol{Z}_p$. Solving for a new task $\boldsymbol{T}_{p+1}$ means computing $\boldsymbol{z}^*_{p+1}$ and $\boldsymbol{v}^*_{p+1}$ from the following QP problem:

$$\begin{aligned}
\min_{\boldsymbol{z}_{p+1}, \boldsymbol{v}_{p+1}} \quad & \frac{1}{2}\|\boldsymbol{A}_{p+1}(\boldsymbol{x}^* + \boldsymbol{Z}_p \boldsymbol{z}_{p+1}) - \boldsymbol{b}_{p+1}\|^2 + \frac{1}{2}\|\boldsymbol{v}_{p+1}\|^2 \\
\text{s. t.} \quad & \boldsymbol{D}_{p+1}(\boldsymbol{x}^* + \boldsymbol{Z}_p \boldsymbol{z}_{p+1}) - \boldsymbol{f}_{p+1} \leq \boldsymbol{v}_{p+1} \\
& \boldsymbol{D}_{p}(\boldsymbol{x}^* + \boldsymbol{Z}_p \boldsymbol{z}_{p+1}) - \boldsymbol{f}_{p} \leq \boldsymbol{v}^*_{p} \\
& \qquad \vdots \\
& \boldsymbol{D}_{1}(\boldsymbol{x}^* + \boldsymbol{Z}_p \boldsymbol{z}_{p+1}) - \boldsymbol{f}_{1} \leq \boldsymbol{v}^*_{1} \\
& \boldsymbol{v}_{p+1} \geq \boldsymbol{0}.
\end{aligned} \tag{4.9}$$

Writing $\boldsymbol{\xi}_p^T = \begin{bmatrix} \boldsymbol{z}_p^T & \boldsymbol{v}_p^T \end{bmatrix}$, problem eq.(4.9) can be rewritten as

$$\begin{aligned}
\min_{\boldsymbol{\xi}_{p+1}} \quad & \frac{1}{2}\boldsymbol{\xi}_{p+1}^T \boldsymbol{H}_{p+1} \boldsymbol{\xi}_{p+1} + \boldsymbol{c}_{p+1}^T \boldsymbol{\xi}_{p+1} \\
\text{s. t.} \quad & \tilde{\boldsymbol{D}}_{p+1} \boldsymbol{\xi}_{p+1} \leq \tilde{\boldsymbol{f}}_{p+1},
\end{aligned} \tag{4.10}$$

where

$$\begin{aligned}
\boldsymbol{H}_{p+1} &= \begin{bmatrix} \boldsymbol{Z}_p^T \boldsymbol{A}_{p+1}^T \boldsymbol{A}_{p+1} \boldsymbol{Z}_p & \boldsymbol{0} \\ \boldsymbol{0} & \mathbb{I} \end{bmatrix} \\
\boldsymbol{c}_{p+1} &= \begin{bmatrix} \boldsymbol{Z}_p^T \boldsymbol{A}_{p+1}^T (\boldsymbol{A}_{p+1} \boldsymbol{x}^* - \boldsymbol{b}_{p+1}) \\ \boldsymbol{0} \end{bmatrix} \\
\tilde{\boldsymbol{D}}_{p+1} &= \begin{bmatrix} \boldsymbol{D}_{p+1}\boldsymbol{Z}_p & -\mathbb{I} \\ \boldsymbol{D}_p \boldsymbol{Z}_p & \boldsymbol{0} \\ \vdots & \\ \boldsymbol{D}_1 \boldsymbol{Z}_p & \boldsymbol{0} \\ \boldsymbol{0} & -\mathbb{I} \end{bmatrix}, \tilde{\boldsymbol{f}}_{p+1} = \begin{bmatrix} \boldsymbol{f}_{p+1} - \boldsymbol{D}_{p+1}\boldsymbol{x}^* \\ \boldsymbol{f}_p - \boldsymbol{D}_p \boldsymbol{x}^* + \boldsymbol{v}^*_p \\ \vdots \\ \boldsymbol{f}_1 - \boldsymbol{D}_1 \boldsymbol{x}^* + \boldsymbol{v}^*_1 \\ \boldsymbol{0} \end{bmatrix}.
\end{aligned} \tag{4.11}$$

The computation of the null space basis $\boldsymbol{Z}_p$ of the stack of equality constraints $\underline{\boldsymbol{A}}$ can be implemented employing the *QR* or the *SVD* decomposition. To speed up the computation time, we use the former decomposition, which has a complexity that grows with the cube of the rows of its argument. For this reason, the computation of the null space of $\underline{\boldsymbol{A}}$ becomes slower as more equalities are added to the stack of tasks. A more efficient

approach is then to use an iterative approach. Given two full row rank matrices A_1 and A_2, the null space basis of $\underline{A} = \begin{bmatrix} A_1^T & A_2^T \end{bmatrix}^T$, Z_2 can be computed as

$$Z_2 = \mathcal{N}(A_1)\,\mathcal{N}(A_2\mathcal{N}(A_1)) = Z_1\mathcal{N}(A_2Z_1) \tag{4.12}$$

It can be easily shown that eq.(4.12) is a null space basis for both A_1 and A_2:

$$A_1Z_2 = A_1Z_1\mathcal{N}(A_2Z_1) = \mathbf{0}\cdot\mathcal{N}(A_2Z_1) = \mathbf{0} \tag{4.13a}$$
$$A_2Z_2 = A_2Z_1\mathcal{N}(A_2Z_1) = \mathbf{0}. \tag{4.13b}$$

4.5 WHOLE BODY CONTROL AS TASK FORMULATION

We formulate the WBC problem as a quadratic optimization problem composed of linear equality and inequality tasks. To this end, all (dynamic) constraints and objectives need to be brought into the form of (4.8). This optimization problem is then solved using the hierarchical optimization algorithm to determine the desired actuator torques $\boldsymbol{\tau}_d$.

4.5.1 *Equality tasks*

Dynamic consistency

The equations of motion of a mechanical system set a relationship between the generalized accelerations $\dot{\boldsymbol{u}}$, the constraint forces $\boldsymbol{\lambda}$ and the actuation torques $\boldsymbol{\tau}$. As shown in section 4.3.1, as long as the motion is in the null space of the contact constraints, the contact forces can be expressed as a linear combination of torques and resulting motion. As a consequence, we search for a solution $\boldsymbol{x}$ in the space of motion and torques, resulting in

$$\boldsymbol{x}_d = \begin{bmatrix} \dot{\boldsymbol{u}}_d^T & \boldsymbol{\tau}_d^T \end{bmatrix}^T, \tag{4.14}$$

with $\dot{\boldsymbol{u}}_d \in \mathbb{R}^n$ and $\boldsymbol{\tau}_d \in \mathbb{R}^r$. Using this optimization variable, the equations of motion (4.6b) can be written as

$$\boldsymbol{Q}_u^T \begin{bmatrix} -\boldsymbol{M}(\boldsymbol{q}) & \boldsymbol{S}^T \end{bmatrix} \boldsymbol{x}_d = \boldsymbol{Q}_u^T \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}). \tag{4.15}$$

Contact motion constraints

Since legs in contact with the ground are not allowed to move, the following task is required to ensure contact consistency:

$$\begin{bmatrix} {}_I\boldsymbol{J}_{F_i} & \boldsymbol{0}_{3\times n_\tau} \end{bmatrix} \boldsymbol{x}_d = -{}_I\dot{\boldsymbol{J}}_{F_i}\boldsymbol{u}. \tag{4.16}$$

Task space motion tracking

Similar to the motion constraints formulated for the feet in contact, we can also define motion tasks for other points of interest such as e.g. the system center of gravity. However, in contrast to the contacts, we want to define these tasks often not with respect to the inertial frame. In our previous works ([52], [22]) we have shown how to estimate the terrain orientation by fitting a plane through the most recent stance location of each foot. The roll and pitch of this plane, together with the yaw of the main body, define the orientation of the so-called *control frame* C, in which we define our desired body and feet motions. The mapping to operational space velocities in the inertial frame is defined by ${}_C\boldsymbol{v} = \boldsymbol{C}_{CI} \cdot {}_I\boldsymbol{v} = \boldsymbol{C}_{CI} \cdot {}_I\boldsymbol{J}\boldsymbol{u}$, where $\boldsymbol{C}_{CI}$ is the rotation matrix that projects the components of a vector from I frame to C frame. The control frame is time-varying, hence using the well known relationship $\dot{\boldsymbol{C}}_{CI} = \mathcal{S}({}_C\boldsymbol{\omega}_{CI})\boldsymbol{C}_{CI}$, where $\mathcal{S}(\cdot)$ is the skew-matrix operator defined as $\mathcal{S}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$, we can compute the mapping between accelerations in I and C frame as ${}_C\dot{\boldsymbol{v}} = {}_C\boldsymbol{J}\dot{\boldsymbol{u}} + {}_C\dot{\boldsymbol{J}}\boldsymbol{u}$. The tasks for different motion objectives can now be formulated. Given a motion plan ${}_C\boldsymbol{r}^{des}_{x,y}$, ${}_C\dot{\boldsymbol{r}}^{des}_{x,y}$ and ${}_C\ddot{\boldsymbol{r}}^{des}_{x,y}$ for the x and y components of the center of mass, tracking in the Operational Space can be achieved by writing

$$\begin{aligned} \begin{bmatrix} {}_C\boldsymbol{J}_{B_{x,y}} & \boldsymbol{0}_{2\times n_\tau} \end{bmatrix} \boldsymbol{x}_d = {}_C\ddot{\boldsymbol{r}}^{des}_{x,y} + \boldsymbol{K}_D \cdot {}_C\dot{\tilde{\boldsymbol{r}}}_{x,y} \\ + \boldsymbol{K}_P \cdot {}_C\tilde{\boldsymbol{r}}_{x,y} - {}_C\dot{\boldsymbol{J}}_{B_{x,y}}\boldsymbol{u}, \end{aligned} \tag{4.17}$$

where $\boldsymbol{K}_P, \boldsymbol{K}_D \in \mathbb{R}^{2\times 2}$ are diagonal positive definite matrices which define proportional and derivative gains, ${}_C\dot{\tilde{\boldsymbol{r}}}_{x,y}$ is the velocity error ${}_C\dot{\boldsymbol{r}}^{des}_{x,y} - {}_C\dot{\boldsymbol{r}}_{x,y}$, and ${}_C\tilde{\boldsymbol{r}}_{x,y}$ is the position error ${}_C\boldsymbol{r}^{des}_{x,y} - {}_C\boldsymbol{r}_{x,y}$.

The same approach is taken for the main body orientation given the desired motion plan $\boldsymbol{q}^{des}_{CB}$ and ${}_C\boldsymbol{\omega}_{CB}{}^{des}$. The orientation error is defined by the *box minus* operator [53] which yields $\tilde{\boldsymbol{\varphi}} = \boldsymbol{q}^{des}_{CB} \boxminus \boldsymbol{q}_{CB} \in \mathbb{R}^3$, while the velocity error is defined as ${}_C\tilde{\boldsymbol{\omega}}_{CB} = {}_C\boldsymbol{\omega}^{des}_{CB} - {}_C\boldsymbol{\omega}_{CB}$. Hence, the tracking task can be written as

$$\begin{bmatrix} {}_C\boldsymbol{J}_{B_{x,y}} & \boldsymbol{0}_{3\times n_\tau} \end{bmatrix} \boldsymbol{x}_d = \boldsymbol{K}_D \cdot {}_C\tilde{\boldsymbol{\omega}}_{CB} + \boldsymbol{K}_P \cdot \tilde{\boldsymbol{\varphi}} - {}_C\dot{\boldsymbol{J}}_B\boldsymbol{u}. \tag{4.18}$$

Finally, for foot motions we define fifth order Hermite splines which provide desired position, velocity and acceleration in the control frame C. We define the $i-th$ foot velocity error as ${}_C\dot{\tilde{\boldsymbol{r}}}_{F_i} = {}_C\dot{\boldsymbol{r}}^{des}_{F_i} - {}_C\dot{\boldsymbol{r}}_{F_i}$ and the position error as ${}_C\tilde{\boldsymbol{r}}_{F_i} = {}_C\boldsymbol{r}^{des}_{F_i} - {}_C\boldsymbol{r}_{F_i}$.

$$\begin{aligned}\begin{bmatrix} {}_C\boldsymbol{J}_{F_i} & \boldsymbol{0}_{3\times n_\tau} \end{bmatrix} \boldsymbol{x}_d = {}_C\ddot{\boldsymbol{r}}^{des}_{F_i} + \boldsymbol{K}_D \cdot {}_C\dot{\tilde{\boldsymbol{r}}}_{F_i} \\ + \boldsymbol{K}_P \cdot {}_C\tilde{\boldsymbol{r}}_{F_i} - {}_C\dot{\boldsymbol{J}}_{F_i}\boldsymbol{u}.\end{aligned} \tag{4.19}$$

Energy optimization

Similar to the motion optimization objectives, we can also introduce tasks for contact forces or torques. These tasks are mostly taken at lowest priority. Optimization of joint torque is typically used in order to improve energetic efficiency [54]:

$$\begin{bmatrix} \boldsymbol{0}_{n_\tau \times n_u} & \mathbb{I}_{n_\tau \times n_\tau} \end{bmatrix} \boldsymbol{x}_d = 0. \tag{4.20}$$

Slippage reduction

In order to minimize contact forces, which typically reduces the risk of slippage in case of model uncertainty, we can formulate the following:

$$\boldsymbol{R}^{-1}\boldsymbol{Q}_c^T \begin{bmatrix} -\boldsymbol{M}(\boldsymbol{q}) & \boldsymbol{S}^T \end{bmatrix} \boldsymbol{x}_d = \boldsymbol{R}^{-1}\boldsymbol{Q}_c^T \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}). \tag{4.21}$$

Manipulability optimization

Another very interesting optimization criterion is manipulability of the system in order to stay away from non-singular postures. Rather than setting a constant desired main body height w.r.t. the estimated terrain, we can use this DOF to avoid the knee singularities of all legs. To achieve this, we use the definition of manipulability measure as described in [51]:

$$m(\boldsymbol{q}) = \sqrt{\det\left({}_I\boldsymbol{J}_{BF}(q) \cdot {}_I\boldsymbol{J}^T_{BF}(q)\right)}, \tag{4.22}$$

where ${}_I\boldsymbol{J}_{B,F}(q)$ is the 3×3 translational Jacobian that maps leg joint velocities to the operational space linear velocities of a foot. We set the desired height motion as a weighted sum of the gradients of the manipulability

measures of all legs w.r.t. to the knee flexion-extension joints $q_{k_{KFE}}$ and damp it as a function of the main body height velocity:

$$\begin{aligned} {}_I\ddot{z}_B^{des} &= k_0 \sum_{k=0}^{n_{legs}} w_k \frac{\partial m_k(\boldsymbol{q})}{\partial q_{k_{KFE}}} - k_d {}_I\dot{z}_B \\ &= k_0 \sum_{k=0}^{n_{legs}} w_k m_k(\boldsymbol{q}) \operatorname{Tr}\{\frac{\partial {}_I\boldsymbol{J}_{k_{BF}}}{\partial q_{k_{KFE}}} \cdot {}_I\boldsymbol{J}_{k_{BF}}^{\dagger}\} - k_d {}_I\dot{z}_B, \end{aligned} \tag{4.23}$$

where ${}_I\boldsymbol{J}_{k_{BF}}^{\dagger}$ is the pseudo-inverse of ${}_I\boldsymbol{J}_{k_{BF}}$, $w_k \geq 0, w_k \in \mathbb{R}$ is the weighting relative to leg k, and k_0 is a positive real weighting factor. We compute the Hessian matrix $\partial {}_I\boldsymbol{J}_{k_{BF}} / \partial q_{k_{KFE}}$ using the algorithms described in [55]. The task can then be written as

$$\begin{bmatrix} \boldsymbol{S}_z & \boldsymbol{0}_{1 \times n_\tau} \end{bmatrix} \boldsymbol{x}_d = {}_I\ddot{z}_B^{des}, \tag{4.24}$$

where $\boldsymbol{S}_z \in \mathbb{R}^{1 \times n_u}$ is the row vector which selects the entry in $\boldsymbol{x}_d$ associated to the main body height motion.

4.5.2 *Inequality Tasks*

In order to set motion constraints $\boldsymbol{r}^{min}(t) \leq \boldsymbol{r}(t) \leq \boldsymbol{r}^{max}(t)$ in the configuration space, we can approximate $\boldsymbol{r}(t)$ with its Taylor expansion around the current time instant $\bar{t}$ truncated at the second derivative:

$$\boldsymbol{r}(t) \simeq \boldsymbol{r}(\bar{t}) + \dot{\boldsymbol{r}}(\bar{t})\delta t + \frac{1}{2}\ddot{\boldsymbol{r}}(\bar{t})\delta t^2, \tag{4.25}$$

with $\delta t = t - \bar{t}$. The inequality constraints

$$\boldsymbol{r}^{min} \leq \boldsymbol{r}(\bar{t}) + \dot{\boldsymbol{r}}(\bar{t})\delta t + \frac{1}{2}\ddot{\boldsymbol{r}}(\bar{t})\delta t^2 \leq \boldsymbol{r}^{max} \tag{4.26}$$

can be then rewritten as

$$\begin{aligned} \frac{\delta t^2}{2}\boldsymbol{J}\dot{\boldsymbol{u}} &\leq (\boldsymbol{r}^{max} - \boldsymbol{r}(\bar{t})) - \delta t \left(\boldsymbol{J} + \frac{\delta t}{2}\dot{\boldsymbol{J}}\right)\boldsymbol{u} \\ -\frac{\delta t^2}{2}\boldsymbol{J}\dot{\boldsymbol{u}} &\leq (\boldsymbol{r}(\bar{t}) - \boldsymbol{r}^{min}) + \delta t \left(\boldsymbol{J} + \frac{\delta t}{2}\dot{\boldsymbol{J}}\right)\boldsymbol{u}. \end{aligned} \tag{4.27}$$

The time interval δt can be interpreted as a degree of freedom in the design of the control system. By setting it as $\delta t = k_t t_s$ with t_s the control period, the positive parameter $k_t \in \mathbb{R}$ can be tuned to set the number of control loops necessary for the inequality constraints in eq.(4.27) to be deactivated, which determines how soft the constraint will be perceived at the configuration level.

Torque limits

Using this approach, we can formulate the following joint torque limitation task to ensure that the solution complies with the hardware limitations defined by minimum τ^{min} and maximum τ^{max} admissible torques.

$$\begin{bmatrix} \mathbf{0}_{n_\tau \times n_u} & \mathbb{I}_{n_\tau \times n_\tau} \\ \mathbf{0}_{n_\tau \times n_u} & -\mathbb{I}_{n_\tau \times n_\tau} \end{bmatrix} \boldsymbol{x}_d \leq \begin{pmatrix} \tau^{max} \mathbb{1}_{n_\tau \times 1} \\ -\tau^{min} \mathbb{1}_{n_\tau \times 1} \end{pmatrix}, \tag{4.28}$$

where $\mathbb{1}_{n_\tau \times 1} = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \in \mathbb{R}^{n_\tau}$.

Contact force limits

To avoid slipping, the resulting contact forces $\boldsymbol{\lambda} \in \mathbb{R}^{3n_c}$ must be constrained to lie within the friction cones. To obtain linear constraints, we approximate the friction cone with a pyramid. Since we approximate the local terrain with a plane whose orientation defines the control frame, all the friction pyramids will be aligned with it. Hence, given the friction coefficient μ, the heading ${}_I\hat{\boldsymbol{h}}$, lateral ${}_I\hat{\boldsymbol{l}}$ and normal ${}_I\hat{\boldsymbol{n}}$ directions of the control frame expressed in the inertial frame, the friction constraints on the contact force ${}_I\boldsymbol{f}_i$ for the $i-th$ support leg can be written as

$$\begin{aligned} ({}_I\hat{\boldsymbol{h}} - {}_I\hat{\boldsymbol{n}}\mu)^T {}_I\boldsymbol{f}_i &\leq \mathbf{0} \\ -({}_I\hat{\boldsymbol{h}} + {}_I\hat{\boldsymbol{n}}\mu)^T {}_I\boldsymbol{f}_i &\leq \mathbf{0} \\ ({}_I\hat{\boldsymbol{l}} - {}_I\hat{\boldsymbol{n}}\mu)^T {}_I\boldsymbol{f}_i &\leq \mathbf{0} \\ -({}_I\hat{\boldsymbol{l}} + {}_I\hat{\boldsymbol{n}}\mu)^T {}_I\boldsymbol{f}_i &\leq \mathbf{0}. \end{aligned} \tag{4.29}$$

To avoid jumps in the actuation signals when a leg is switching between support and swing mode, we modulate the normal component of $\boldsymbol{f}_i$ according to the motion plan by gradually limiting it to zero before lift-off, and gradually ramping up from zero on touch-down. To achieve this, we add two additional constraints:

$$f_i^{min} \leq {}_I\hat{\boldsymbol{n}}^T {}_I\boldsymbol{f}_i \leq f_i^{max}, \tag{4.30}$$

where f_i^{max} is computed as a function of the stance and swing phases, and $f_i^{min} \geq 0$ is set to a fixed value. Equations (4.29) and (4.30) can be rewritten in compact form as $\boldsymbol{B}_i \boldsymbol{f}_i \leq \boldsymbol{\beta}_i$. Using eq.(4.7), the task constraints in terms of $\boldsymbol{x}_d$ will then be written as

$$\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{Q}_c^T \begin{bmatrix} \boldsymbol{M}(\boldsymbol{q}) & \boldsymbol{S}^T \end{bmatrix} \boldsymbol{x}_d \leq \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{Q}_c^T \boldsymbol{\beta}, \tag{4.31}$$

where $\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_1^T & \dots & \boldsymbol{B}_{n_c}^T \end{bmatrix}^T$ and $\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T & \dots & \boldsymbol{\beta}_{n_c}^T \end{bmatrix}^T$.

Task space motion limits

To limit task space positions, we make use of formulation (4.27). The main body height ${}_I\boldsymbol{r}_{IB_z}$ limits are set w.r.t. the footprint height h_z, which is computed as

$$h_z = \frac{1}{n_c} \sum_{k=0}^{n_{legs}} w_k {}_I\boldsymbol{r}_{IF_k}, \tag{4.32}$$

where n_c is the number of legs in contact. The weight w_k is 1 when leg k is in contact, 0 otherwise. The inequality motion task can be written as

$$h^{min} \leq {}_I\boldsymbol{r}_{IB_z} - h_z \leq h^{max}. \tag{4.33}$$

As previously discussed, the task can be rewritten in joint space as

$$\begin{aligned} &\begin{bmatrix} \frac{\delta t^2}{2} {}_I\boldsymbol{J}_{B_z} & \boldsymbol{0}_{1\times n_\tau} \\ -\frac{\delta t^2}{2} {}_I\boldsymbol{J}_{B_z} & \boldsymbol{0}_{1\times n_\tau} \end{bmatrix} \boldsymbol{x}_d \leq \\ &\begin{pmatrix} h^{max} - {}_I\boldsymbol{r}_{IB_z} + h_z \\ {}_I\boldsymbol{r}_{IB_z} - h_z - h^{min} \end{pmatrix} - \delta t \begin{pmatrix} ({}_I\boldsymbol{J}_{B_z} + \frac{\delta t}{2} {}_I\dot{\boldsymbol{J}}_{B_z}) \\ -({}_I\boldsymbol{J}_{B_z} + \frac{\delta t}{2} {}_I\dot{\boldsymbol{J}}_{B_z}) \end{pmatrix} \boldsymbol{u}. \end{aligned} \tag{4.34}$$

Limb extensions limits can be seen as limits on the relative motion between feet and hips. To achieve terrain adaptation, we write these limits such that the hips will follow the z component of the feet position in I frame. Hence, for each leg k we write

$$\begin{aligned} &\begin{bmatrix} \frac{\delta t^2}{2} {}_I\boldsymbol{J}_{H_z^k} & \boldsymbol{0}_{1\times n_\tau} \\ -\frac{\delta t^2}{2} {}_I\boldsymbol{J}_{H_z^k} & \boldsymbol{0}_{1\times n_\tau} \end{bmatrix} \boldsymbol{x}_d \leq \\ &\begin{pmatrix} h^{max} - {}_I\boldsymbol{r}_{IH_z^k} + h_z^k \\ {}_I\boldsymbol{r}_{IH_z^k} - h_z^k - h^{min} \end{pmatrix} - \delta t \begin{pmatrix} ({}_I\boldsymbol{J}_{H_z^k} + \frac{\delta t}{2} {}_I\dot{\boldsymbol{J}}_{H_z^k}) \\ -({}_I\boldsymbol{J}_{H_z^k} + \frac{\delta t}{2} {}_I\dot{\boldsymbol{J}}_{H_z^k}) \end{pmatrix} \boldsymbol{u}, \end{aligned} \tag{4.35}$$

where ${}_I\boldsymbol{J}_{H_z^k}$ is the third row of the Jacobian which projects the generalized velocities to $k-th$ hip linear velocities in operational space, ${}_I\boldsymbol{r}_{IH_z}$ is the

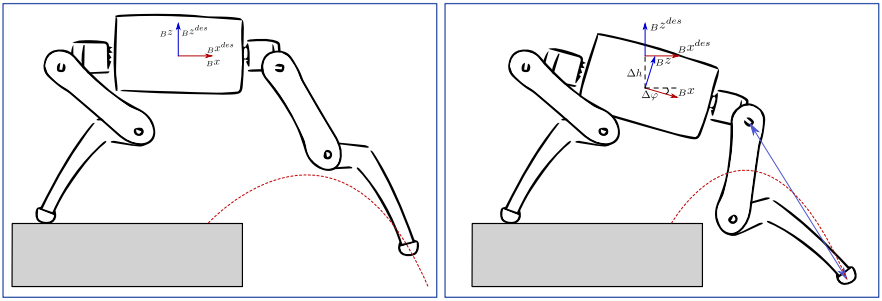


FIGURE 4.3: When a regular set of tasks is employed, walking down a high step requires prior knowledge of the terrain and appropriate planning (left). With our method, a robot can reactively adapt to unperceived steps (right).

z component of the position of the $k-th$ hip in world frame, h_z^k is the height of the foot of the $k-th$ leg, and h^{min} and h^{max} are the minimum and maximum user-defined relative distances between a hip and a foot.

4.6 TERRAIN ADAPTATION

In a previous work [52] we have successfully shown locomotion over unperceived tilted terrain. However, the method described there fails when stepping down from a high step as shown in Fig.4.3. The robot's limb tries to blindly extend until kinematic limits are encountered. In such a situation, it is necessary to plan an appropriate whole-body motion which should take into account several factors such as balancing and kinematic limits. Instead, we can now exploit two advantages of the HO framework: each task can set both equality and inequality constraints in the Operational Space, and tasks are solved in a strict prioritized order. We defer the problem of designing a motion in the planning phase to the problem of reacting to task constraints in the control phase. To achieve this, we set an inequality constraint task which imposes limb extensions limits at a higher priority than torso orientation and height motion tracking, but at a lower priority than swing foot motion tracking (see Tab.4.1). As soon as the kinematic limits are violated, the main body will tilt towards the feet, achieving a reactive and perception-less terrain adaptation with no requirements on motion design (Fig.4.3). Balancing and contact force limit tasks are solved with the highest priority, which guarantees balance at all times.

TABLE 4.1: The tasks used in the experiments described in section 4.7.2. Each task is associated with a priority (1 is the highest).

Priority	Task
1	Equations of Motion
2	No contact motion
3	Torque limits
4	Friction cone and λ modulation
5	Desired torso x, y position
6	Swing foot motion tracking
7	Limb kinematic limits
8	Main body yaw
9	Main body height
10	Main body roll and pitch
11	Contact force minimization

4.7 EXPERIMENTS

4.7.1 *Setup*

Our experiments[1] were conducted on ANYmal [56], an accurately torque-controllable quadrupedal robot. Control signals are generated in a $400Hz$ control loop which runs on a dedicated on-board computer together with state estimation [57] and communication with the drives. For modeling of kinematics and dynamics, we use the open-source Rigid Body Dynamics Library [58] (RBDL), which is a C++ implementation of the algorithms described in [59]. Table 4.1 describes which sub-tasks were implemented in each experiment that we describe in the following subsections. To numerically solve each optimization problem, we use a custom version of the QuadProg++ [60] library, a C++ open-source QP solver which implements the Active Set algorithm described in [61].

4.7.2 *Full terrain adaptation*

To test our framework, we are using a simple motion planner that implements a walking behavior based on the Zero Moment Point (ZMP) [9]. We

1 https://youtu.be/AjiLCbJUYKI

plan the motion of the center of mass by solving a constrained Quadratic Programming problem, as shown in [62] and [63]. By minimizing the desired accelerations of the motion plan, the optimization problem provides the coefficients of a fifth order Hermite spline which result in a smooth continuous motion. The results presented in Fig.4.4 show the tracking performance obtained while walking on flat terrain. To execute locomotion over uneven terrain, we implement the method discussed in section 4.6, i.e. we constrain the motion of the main body to avoid over-extension of the limbs. We achieve this by setting an inequality task which has a higher priority than main body height and orientation tracking tasks described in section 4.5. The step-down sequence depicted in Fig.4.5 shows how the control framework allows to negotiate a 14cm step (23% of the maximum leg length). Automatic terrain adaptation is achieved when the orientation tracking task is subject to the higher priority limb extension kinematic limits. The 10-14 s time interval from Fig.4.6 shows how the orientation error is kept low as long as the relative distance on the ${}_{I}z$ direction is below the kinematic limits, which can be set by the user. When the foot motion is such that these limits are violated, the control framework tries to deactivate them by tilting the main body in the direction of the foot motion. Fig.4.6 additionally shows that the main body height tracking task performance also degrades during the adaptation phase. The entire adaptation behavior can be summarized in the following steps, clarified in Fig.4.5: 1) the desired height of the right front (RF) leg is gradually lowered until a touch-down event is registered; 2) when the limb extensions limits are violated for the RF leg, the main body torso orientation adapts by tilting forward; 3) the left hind leg (LH) kinematic limits are violated because of the main body tilting, so the height of the main body gradually decreases until the limits are no longer active.

4.8 CONCLUSIONS AND FUTURE WORK

A whole body control framework was implemented which uses hierarchical optimization to solve a set of prioritized tasks. We present a clever combination of several inequality tasks in order to optimally exploit the system redundancy and hence adaptively shape the motion of the robot as a function of the terrain. The framework has allowed ANYmal, a fully torque-controllable quadrupedal robot, to overcome unperceived steps, with no requirement on motion design from the user. The framework has been thoroughly tested in a series of experiments, which include walking, ma-

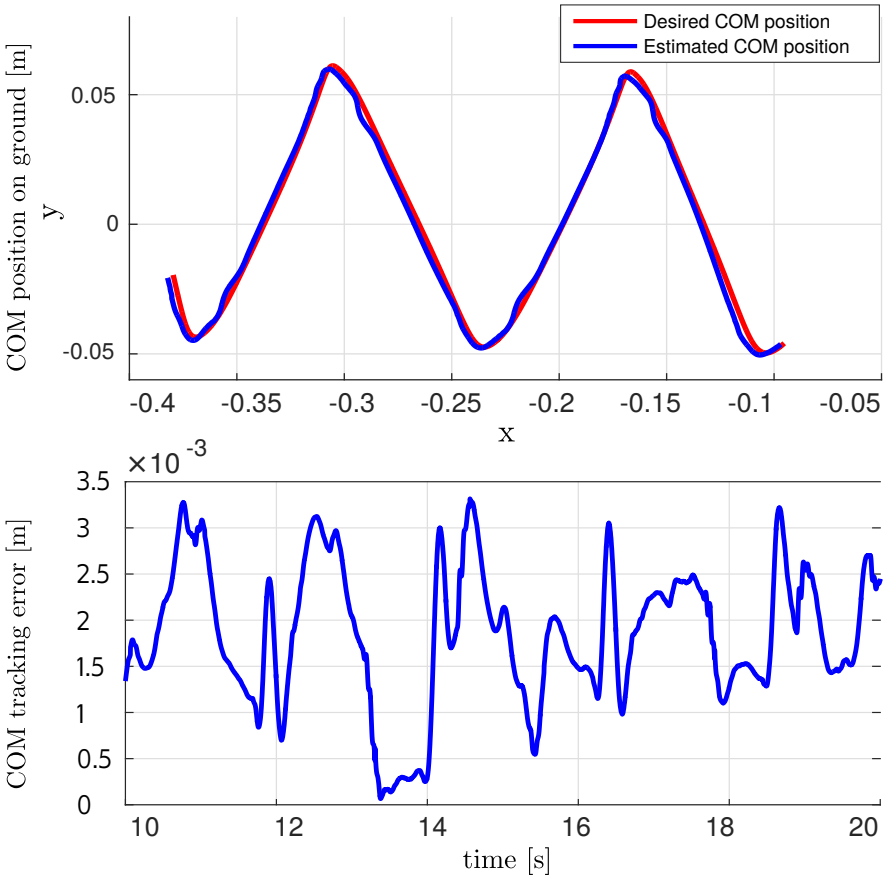


FIGURE 4.4: The tracking behavior of the center of mass (COM) of the whole body during a ZMP based walking gait. During this experiment, the position error had an upper bound of 3.4 mm.

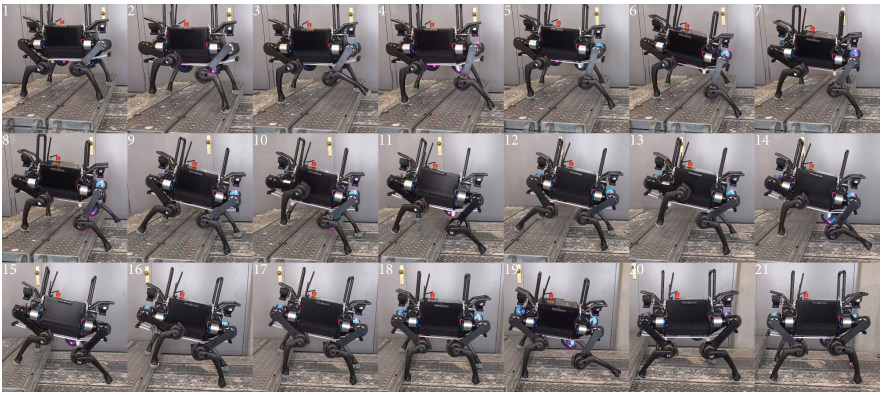


FIGURE 4.5: A sequence showing ANYmal reacting to an unperceived 14*cm* step. The reactive behavior is obtained by imposing motion constraints between the hips and the feet. The terrain is initially assumed to be flat. The figure shows how the main body adapts to avoid over-extension of the right front (sequence 5-7) and right hind (sequence 16-18) legs. Main body motion tracking is sacrificed in favor of terrain adaptation.

nipulability optimization and terrain adaptation. Next steps involve using this framework to implement features such as self-collision avoidance and the use of additional limbs such as arms to allow for manipulation.

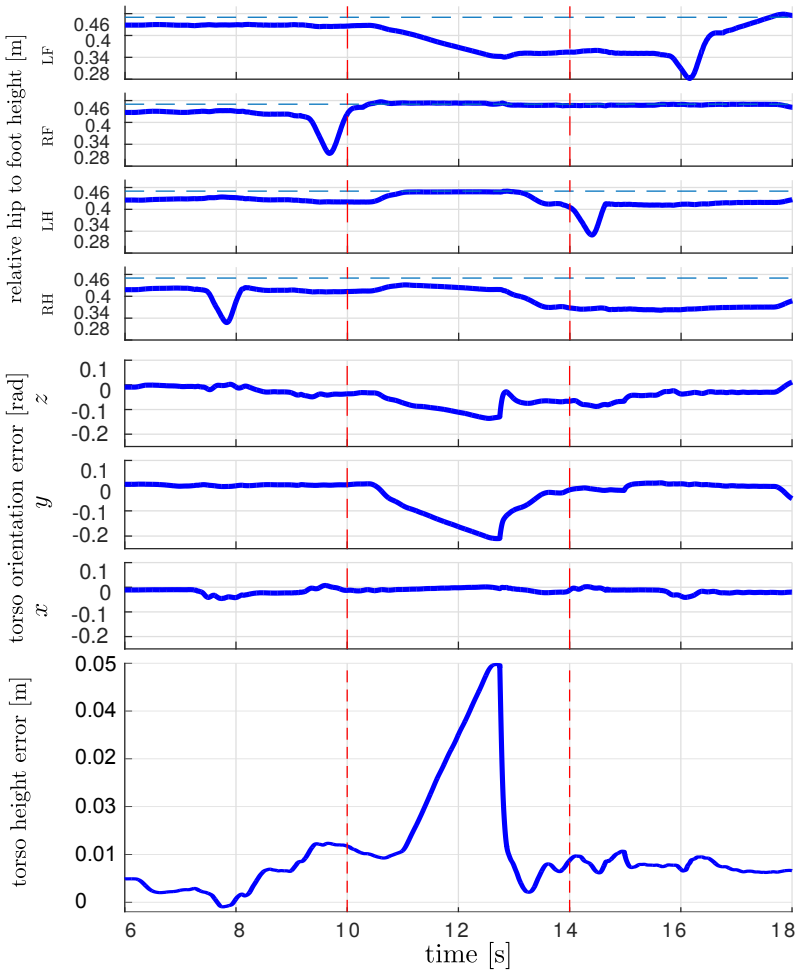


FIGURE 4.6: The results obtained from the terrain adaptation experiment. With reference to the sequence of frames numbered 3-8 in Fig.4.5, it can be seen how the kinematic limits are reached for the right fore (RF) leg shortly after 10 s. At that point, the pitch and roll tracking tasks are taken over by the limb extension limits task which also overcome height tracking when the left hind (LH) leg violates the limits. The limits are soften by choosing $k_t t_s = 0.1$. When the swing foot makes contact, a new estimation of the terrain is available, which yields a new pose reference for the torso.

5

PAPER 2: DYNAMIC LOCOMOTION AND WHOLE-BODY CONTROL FOR QUADRUPEDAL ROBOTS

5.1 ABSTRACT

This paper presents a framework which allows a quadrupedal robot to execute dynamic gaits including trot, pace and dynamic lateral walk, as well as a smooth transition between them. Our method relies on an online ZMP based motion planner which continuously updates the reference motion trajectory as a function of the contact schedule and the state of the robot. The planner is coupled with a hierarchical whole-body controller which optimizes the whole-body motion and contact forces by solving a cascade of prioritized tasks. We tested our framework on ANYmal, a fully torque controllable quadrupedal robot which is actuated by series-elastic actuators.

5.2 INTRODUCTION

Research focused on robotic locomotion has made great efforts in trying to reach the natural capabilities of animals, which are capable of executing highly dynamic gaits and naturally perform quick and smooth transitions from one gait to another. The recent advances in both the motion planning and control methods, as well as the increase in performance of both hardware and software, are narrowing the gap between legged robots and their biological counterparts. Although we are still far from a fully autonomous system capable of locomotion in all environments, the recent years have seen several important results.

Dynamic locomotion is a complex problem which involves a multi-body system which interacts with the environment through several contact points. The execution of dynamic gaits for such a system over unknown and challenging terrain requires careful motion planning and accurate motion and force control. Research groups and companies have demonstrated the ability to walk, run and jump over challenging terrain. *Cheetah 2* [64] from MIT has been shown able to run and jump over obstacles which are 80% of the robot's leg length using an MPC controller. Recent results

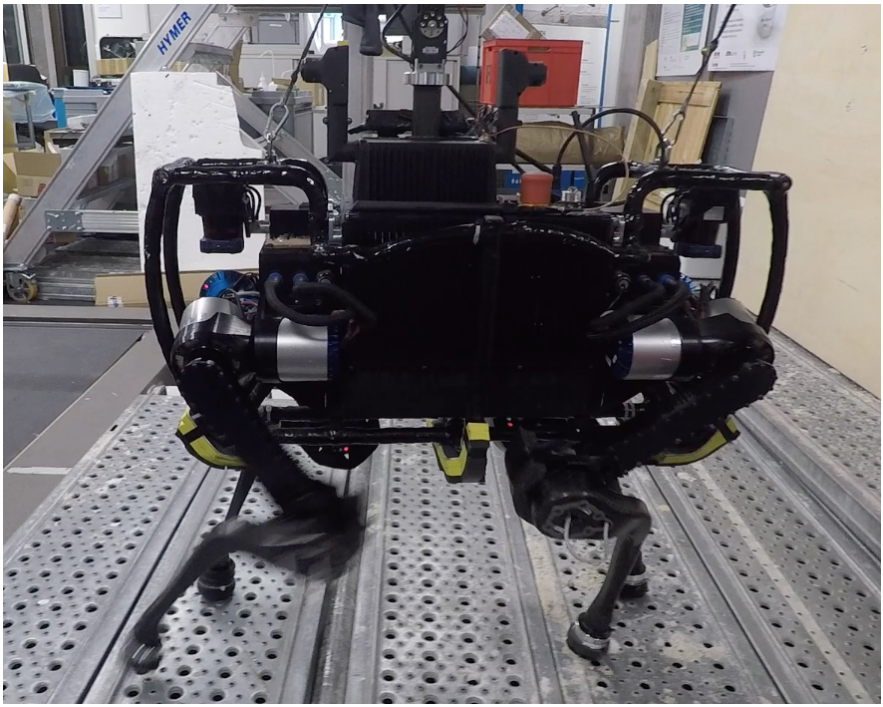


FIGURE 5.1: Our motion plan and control framework, tested on the fully torque-controllable quadrupedal robot ANYmal, can execute a dynamic walk which includes overlapping swing phases for lateral legs (e.g. left-front and left-hind legs).

have shown *ANYmal* [56] walking [65], trotting [22], and climbing over stairs [66] by employing whole-body control and motion optimization algorithms. The IIT *HyQ* [11] quadruped has been shown walking using a creeping gait [67] over stepping stones [63]. Motion planning for the torso was obtained by solving an optimization problem which constrained the *Zero-Moment Point* (ZMP) [9] to lie in the support polygons.

A notable example of dynamic locomotion is given by the legged robots developed by Boston Dynamics. One of their recent legged robots, *Spot*, has demonstrated a wide variety of dynamic gaits, including a dynamic walk, trotting with full flight phases, pacing, as well as demonstrating smooth transitions between these. *Spot mini*, a more recent machine, has shown similar skills as well as manipulation capability with a robotic arm. While impressive, the methods to achieve such results are not available. In this

paper, we contribute a novel framework and methods for motion planning and control algorithms needed to execute complex and dynamic gaits.

Based on the results shown in [62] and [63], we developed a ZMP based motion planner which can generate a motion through a sequence of arbitrary support polygons, including triangles and quadrilaterals, but also lines, which is the case for a dynamic walk with overlapping swing phases or for a trotting and pacing gait. Rather than computing the optimal motion on discrete events (e.g. a foot touchdown), we recompute the optimization as soon as the previous one ends successfully, providing us with a motion plan which continuously adapts to the new state of the robot as well as to changes in the gait pattern. We also reformulate the cost function to include costs on the deviation from previous solutions, as well as costs which penalize the deviation from a path regularizer. Motion tracking is obtained by means of a whole-body controller which solves a sequence of motion and contact force prioritized tasks using the hieararchical optimization formulation. We demonstrate our method by executing dynamic gaits such as a dynamic lateral walk with overlapping swing phases, a trot and a pace on the quadrupedal robot ANYmal. We conclude our experiments by showing a transition between a dynamic lateral walk and a pace gait. To the best of our knowledge, this is the first contribution with results and methodologies on the execution of such dynamic gaits, specifically a dynamic lateral walk, a pacing gait and transitions between them.

5.3 MODEL FORMULATION

In general, a walking robot can be modeled as a system with a free-floating base B to which legs are attached (see Fig.5.2). The motion of the entire system can be described w.r.t. a fixed inertial frame I. We write the position vector from the inertial to the base frame expressed in the inertial frame as ${}_I\boldsymbol{r}_{IB} \in \mathbb{R}^3$ and use Hamiltonian unit quaternions to parametrize the orientation of the main body. The limb joint angles are stacked in the vector $\boldsymbol{q}_j \in \mathbb{R}^{n_j}$. We write the generalized coordinates vector $\boldsymbol{q}$ and the generalized velocities vector $\boldsymbol{u}$ as

$$\boldsymbol{q} = \begin{bmatrix} {}_I\boldsymbol{r}_{IB} \\ \boldsymbol{q}_{IB} \\ \boldsymbol{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \quad \boldsymbol{u} = \begin{bmatrix} {}_I\boldsymbol{v}_B \\ {}_B\boldsymbol{\omega}_{IB} \\ \dot{\boldsymbol{q}}_j \end{bmatrix} \in \mathbb{R}^{n_u}, \tag{5.1}$$

where $n_u = 6 + n_j$, ${}_I\boldsymbol{v}_B$ and ${}_B\boldsymbol{\omega}_{IB}$ are the linear and angular velocity of the base w.r.t. the inertial frame expressed respectively in the I and B frame,

and $\boldsymbol{q}_{IB} \in SO(3)$ is the unit quaternion that projects the components of a vector expressed in B frame to those of the same vector expressed in I frame. The equations of motion of legged systems can be written as $\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} + \boldsymbol{h}(\boldsymbol{q},\boldsymbol{u}) = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}_s^T\boldsymbol{\lambda}$, where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n_u \times n_u}$ is the mass matrix and $\boldsymbol{h}(\boldsymbol{q},\boldsymbol{u}) \in \mathbb{R}^{n_u}$ is the vector of Coriolis, centrifugal and gravity terms. The selection matrix $\boldsymbol{S} = \begin{bmatrix} \boldsymbol{0}_{n_\tau \times (n_u - n_\tau)} & \mathbb{I}_{n_\tau \times n_\tau} \end{bmatrix}$ selects which DoFs are actuated. If all limb joints are actuated, then $n_\tau = n_j$. The vector of constraint forces $\boldsymbol{\lambda}$ is mapped to the joint space torques through the support Jacobian $\boldsymbol{J}_s \in \mathbb{R}^{3n_c \times n_u}$, which is obtained by stacking the constraint Jacobians as $\boldsymbol{J}_s = \begin{bmatrix} \boldsymbol{J}_{C_1}^T & \cdots & \boldsymbol{J}_{C_{n_c}}^T \end{bmatrix}^T$, with n_c the number of limbs in contact. Motion at the supporting contact points must be constrained. If the feet are modeled as point contacts, then each contact introduces three constraint equations ${}_I\boldsymbol{r}_{IC}(t) = const$, which can be differentiated twice to yield

$$ {}_I\dot{\boldsymbol{r}}_{IC} = \boldsymbol{J}_C\boldsymbol{u} = \boldsymbol{0}, \quad {}_I\ddot{\boldsymbol{r}}_{IC} = \boldsymbol{J}_C\dot{\boldsymbol{u}} + \dot{\boldsymbol{J}}_C\boldsymbol{u} = \boldsymbol{0}. \tag{5.2} $$

5.4 HIERARCHICAL OPTIMIZATION

We have recently shown [65] the implementation of a whole body controller which finds the optimal joint torques and joint accelerations by solving a cascade of prioritized QP problems. In this work we use a different implementation which solves for contact forces rather than joint torques. Hence, we search for a vector $\boldsymbol{\xi}_d = \begin{bmatrix} \dot{\boldsymbol{u}}_d^T & \boldsymbol{\lambda}_d^T \end{bmatrix}^T \in \mathbb{R}^{n_u + n_c}$ of desired joint accelerations and contact forces. The dimension of ξ_d is a function of the number of contact points n_c. When the feet are modeled as point contacts, the dimensions of ξ_d will be smaller or equal to the case of optimizing for joint torques. As noted in [68], this is beneficial in terms of computation speed when solving the numerical optimization of tasks. It also results in an more natural way of expressing constraints of contact forces.

Equations of motion

By taking advantage of the decomposition induced by the selection matrix $\boldsymbol{S}^T$, we can constrain the motion and the contact forces to live on the manifold described by the floating base system dynamics by writing

$$ \begin{bmatrix} \boldsymbol{M}_{fb} & -\boldsymbol{J}_{s_{fb}}^T \end{bmatrix} \boldsymbol{\xi}_d = -\boldsymbol{h}_{fb}, \tag{5.3} $$

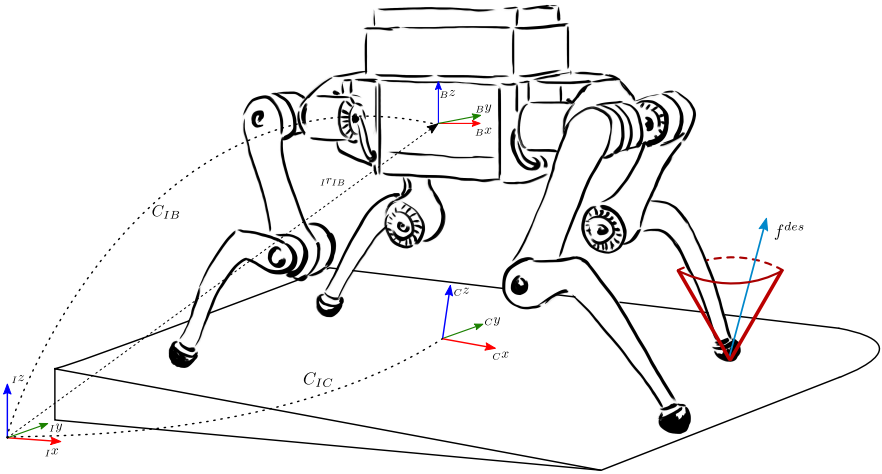


FIGURE 5.2: A sketch of the quadrupedal robot ANYmal. The floating base pose is defined by the position of the base frame wr.t. the inertial frame expressed in the inertial frame ${}_I\boldsymbol{r}_{IB}$, and by the orientation of the base w.r.t. the inertial frame $\boldsymbol{C}_{IB}$. As discussed in earlier works [22], we define a control frame $\boldsymbol{C}_{IC}$ which is aligned with the local estimation of the terrain and with the robot's heading direction.

where $\boldsymbol{M}_{fb}$, $\boldsymbol{J}_{s_{fb}}^T$ and $\boldsymbol{h}_{fb}$ are the first six rows of the composite inertia matrix, the support Jacobian and the nonlinear terms respectively, which are the equations related to the dynamics of the floating base.

Contact motion constraint

The solution found by the controller should not violate the contact constraints defined in (5.2). Hence, we impose null accelerations at the contact points by setting

$$\begin{bmatrix} \boldsymbol{J}_s & \boldsymbol{0}_{3n_c \times 3n_c} \end{bmatrix} \boldsymbol{\xi}_d = -\dot{\boldsymbol{J}}_s \boldsymbol{u}. \tag{5.4}$$

Contact force and torque limits

To avoid slipping, contact forces should be constrained to lie in the friction cone which is aligned with the normal to the contact surface ${}_I\boldsymbol{n}$ expressed in world frame and is a function of the friction coefficient μ. To write these constraints as linear ones, we approximate the friction cones with pyramids. These are aligned with the unit perpendicular vectors ${}_I\boldsymbol{l}$ and ${}_I\boldsymbol{h}$

which lie on the contact surface. We achieve this by writing constraints on the k-th contact force $\boldsymbol{\lambda}_k$ as

$$\begin{aligned} ({}_I\boldsymbol{h} - {}_I\boldsymbol{n}\mu)^T {}_I\boldsymbol{\lambda}_k &\leq \mathbf{0} \\ -({}_I\boldsymbol{h} + {}_I\boldsymbol{n}\mu)^T {}_I\boldsymbol{\lambda}_k &\leq \mathbf{0} \\ ({}_I\boldsymbol{l} - {}_I\boldsymbol{n}\mu)^T {}_I\boldsymbol{\lambda}_k &\leq \mathbf{0} \\ -({}_I\boldsymbol{l} + {}_I\boldsymbol{n}\mu)^T {}_I\boldsymbol{\lambda}_k &\leq \mathbf{0}. \end{aligned} \tag{5.5}$$

Joint actuation torques should also be limited to avoid mechanical limits given by the actuators. We can write

$$\boldsymbol{\tau}_{min} - \boldsymbol{h}_j \leq \begin{bmatrix} \boldsymbol{M}_j & -\boldsymbol{J}_{s_j}^T \end{bmatrix} \boldsymbol{\xi}_d \leq \boldsymbol{\tau}_{max} - \boldsymbol{h}_j \quad , \tag{5.6}$$

where the j subscript is relative to the rows of the equations of motion describing the joint dynamics. The quantities $\boldsymbol{\tau}_{min}$ and $\boldsymbol{\tau}_{max}$ are n_j dimensional vectors of minimum and maximum actuator torques.

Motion tracking

To track the desired motion of the floating base and the swing legs, we constrain the joint accelerations by implementing operational space controllers with feed-forward reference acceleration and a motion dependent state feedback state. For the main body we write

$$\begin{aligned} \begin{bmatrix} {}_C\boldsymbol{J}_P & \mathbf{0} \end{bmatrix} \boldsymbol{\xi}_d = {}_C\ddot{\boldsymbol{r}}_{IB}^d + \boldsymbol{k}_D^{pos}({}_C\dot{\boldsymbol{r}}_{IB}^d - {}_C\boldsymbol{v}_B) \\ + \boldsymbol{k}_P^{pos}({}_C\boldsymbol{r}_{IB}^d - {}_C\boldsymbol{r}_{IB}) \end{aligned} \tag{5.7}$$

for the linear motion and

$$\begin{bmatrix} {}_C\boldsymbol{J}_R & \mathbf{0} \end{bmatrix} \boldsymbol{\xi}_d = -\boldsymbol{k}_D^{ang}{}_C\boldsymbol{\omega}_B + \boldsymbol{k}_P^{ang}(\boldsymbol{q}_{CB}^d \boxminus \boldsymbol{q}_{CB}) \tag{5.8}$$

for the angular motion. The Jacobian matrices ${}_C\boldsymbol{J}_P$ and ${}_C\boldsymbol{J}_R$ are the translational and rotational Jacobian associated to the base expressed in the *Control frame C*, which is a frame aligned with the local estimation of the terrain and with the heading direction of the robot (for more details see [65] and [22]). The $\boxminus$ operator yields the Euler vector which represents the relative orientation between the desired attitude $\boldsymbol{q}_{CB}^d$ and the estimated one $\boldsymbol{q}_{CB}$. The gains $\boldsymbol{k}_P^{pos}$, $\boldsymbol{k}_D^{pos}$, $\boldsymbol{k}_P^{ang}$ and $\boldsymbol{k}_D^{ang}$ are diagonal positive definite matrices of control gains. The reference motion ${}_C\boldsymbol{r}_{IB}$ and its derivatives are the output of the motion plan which is described in section 5.5.

TABLE 5.1: The list of prioritized tasks used in our experiments. Each task is associated with a priority (1 is the highest).

Priority	Task
1	Floating base equations of motion
2	Torque limits
	Friction cone and λ modulation
3	No contact motion
4	Center of mass linear motion
	Center of mass angular motion
	Swing leg motion tracking
5	Contact force minimization

Contact force minimization

We minimize the contact forces by setting

$$\begin{bmatrix} \mathbf{0}_{3n_c \times n_u} & \mathbb{I}_{3n_c \times 3n_c} \end{bmatrix} \boldsymbol{\xi}_d = \mathbf{0}. \tag{5.9}$$

Table 5.1 summarizes the tasks and their priority (a lower value indicates a higher priority) used throughout the experiments.

Given a computed set of joint motions and contact forces $\boldsymbol{\xi}_d = \begin{bmatrix} \dot{\boldsymbol{u}}_d^T & \boldsymbol{\lambda}_d^T \end{bmatrix}^T$, we compute the reference joint torques as $\boldsymbol{\tau}_d = \begin{bmatrix} \boldsymbol{M}_j & -\boldsymbol{J}_{s_j}^T \end{bmatrix} \boldsymbol{\xi}_d + \boldsymbol{h}_j$, where $\boldsymbol{M}_j$, $\boldsymbol{J}_{s_j}^T$ and $\boldsymbol{h}_j$ are defined as in (5.6).

5.5 MOTION OPTIMIZATION

We describe how to obtain a motion plan for the x and y coordinates of the whole-body center of mass (com) such that balance is ensured during locomotion at all times. Fig.5.3 shows the resulting motion plan obtained during the execution of a dynamic lateral walk.

The locomotion framework can receive high-level velocity commands from an external source, i.e. an operator device or a high-level navigation planner. To drive locomotion to the reference speed, footholds are generated for all legs (section 5.5.1) to obtain the desired average velocity of the torso. This information, together with the footfall pattern (e.g. fig.5.4), is used to generate a sequence of support polygons (section 5.5.2) which are sent to the motion plan optimizer (section 5.5.3). This in turn will produce

position, velocity and acceleration profiles for the x and y coordinates of the whole-body center of mass.

The entire plan is computed in the *Plan frame P* which is located at the origin of the inertial frame and follows the yaw rotation of the torso of the robot. Hence, it is always aligned to the heading direction of the robot. This allows us to interpret the problem formulation contributions to the x and y coordinates of the plan as constraints on heading and lateral directions of the motion.

5.5.1 *Foothold generation*

External high-level velocity commands are used to drive locomotion in a specific direction and speed by adjusting the reference footholds, which are computed for each leg at each new control loop. Depending on the contact state of a leg, these are computed in two different ways: when a leg is in contact, the commanded velocities will be projected to foothold locations computed such that the average torso speed matches the required locomotion speed; if a leg is swinging, the reference foothold is computed in the same way but added to a velocity feedback term which stabilizes the robot when it receives a push which produces a velocity control error. For more details, see [22].

5.5.2 *Support polygon sequence*

We describe a gait by defining lift-off ϕ_{lo} and touch-down ϕ_{td} events in the phase domain for each leg. This also defines a contact schedule for all legs. Fig.5.4 shows the phase events for a dynamic lateral walk, which exhibits overlapping swing phases for fore/hind left and fore/hind right legs. Given the touch-down and lift-off events for all legs, as well as the current feet locations and the set of desired footholds, we can compute a sequence of support polygons (defined as the tuple of vertices and time duration in seconds) to be used in the motion planner. We do this whenever a new motion plan is available, such that the new solution adapts to changes in the contact schedule, changes in the reference footholds and changes in the high-level operator velocity. To compute each polygon, we start from the current phase φ and store the $x - y$ coordinates of the feet which are in contact. We search for the next phase event φ_k with $k = 1$ to get the duration of the first polygon $t_0 = t_s(\varphi_k - \varphi)$, with t_s the stride duration in seconds. Thus, we have completely defined a support polygon by

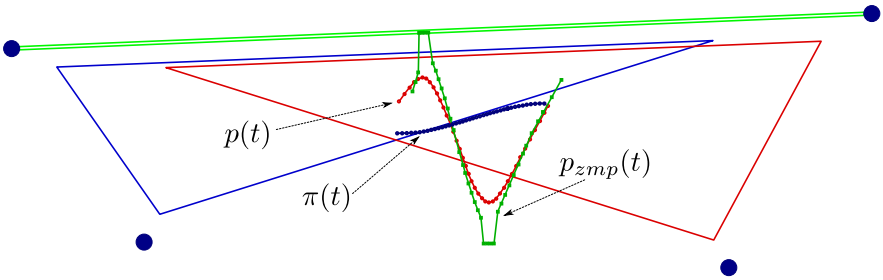


FIGURE 5.3: The solution obtained by the motion planner when a dynamic lateral walk gait is active. The support polygons are computed as the convex hull of the points which are in contact (large blue dots) according to the contact schedule. To increase robustness of the solution, a safety margin is applied to the polygons such that their area is reduced. Only the first three support polygons of the sequence (respectively in red, green and blue) are shown. The resulting center of mass motion $\boldsymbol{p}(t)$ (red path) is optimized to deviate the least possible from a path regularizer $\boldsymbol{\pi}(t)$ (blue path), while the ZMP $\boldsymbol{p}_{zmp}(t)$ (green path) is constrained to always lie in the active support polygon. When two lateral legs are simultaneously swinging, the support polygon collapses to a line (shown in green). To avoid numerical issues and excessive accelerations due to a required increase in the order of the approximating spline, we expand the line to a rectangle.

its vertices and its duration in seconds. We iterate by updating φ to φ_k and searching for the next phase event. Since the contact schedule is periodic, we repeat these steps until the phase event $\varphi_0 + 1$, which corresponds to the starting phase φ_0.

5.5.3 *Problem formulation*

As done in [63], we choose to represent the motion plan by using two sequences of fifth order polynomial splines for the x and y coordinates of the center of mass position. Each spline has a time duration t_{fk}. Hence, the position $\boldsymbol{p} \in \mathbb{R}^2$ at time t described by the k-th pair of splines in the sequence can be compactly written as

$$\boldsymbol{p}(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_{kx}^T \\ \boldsymbol{\alpha}_{ky}^T \end{bmatrix} \boldsymbol{\eta}(t) \tag{5.10}$$

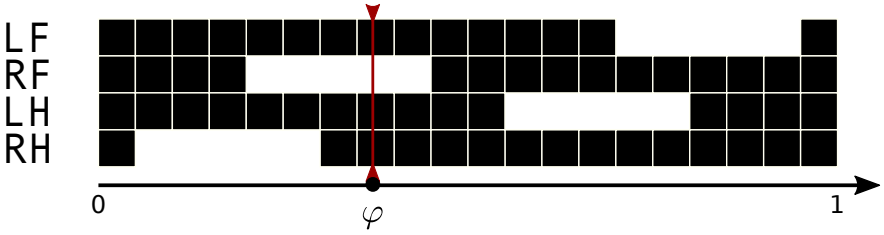


FIGURE 5.4: The contact schedule associated to a dynamic lateral walk. The dark areas represent a contact phase, while the light areas represent a swing phase. The abbreviations stand for left fore (LF), right fore (RF), left hind (LH) and right hind (RH). This gait exhibits an overlap in the swing phases of the left fore/hind and the right fore/hind legs. The red line represents the current phase $\varphi \in [0, 1]$

where

$$\begin{aligned} \boldsymbol{\alpha}_{kx}^T &= \begin{bmatrix} a_{k5}^x & a_{k4}^x & a_{k3}^x & a_{k2}^x & a_{k1}^x & a_{k0}^x \end{bmatrix} \\ \boldsymbol{\alpha}_{ky}^T &= \begin{bmatrix} a_{k5}^y & a_{k4}^y & a_{k3}^y & a_{k2}^y & a_{k1}^y & a_{k0}^y \end{bmatrix} \end{aligned} \tag{5.11}$$

and

$$\boldsymbol{\eta}(t)^T = \begin{bmatrix} t^5 & t^4 & t^3 & t^2 & t & 1 \end{bmatrix}. \tag{5.12}$$

Alternatively we can write

$$\boldsymbol{p}(t) = \begin{bmatrix} \boldsymbol{\eta}(t)^T & \mathbf{0}_{6\times 1} \\ \mathbf{0}_{6\times 1} & \boldsymbol{\eta}(t)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{kx} \\ \boldsymbol{\alpha}_{ky} \end{bmatrix} = \boldsymbol{T}(t)\boldsymbol{\alpha}_k, \tag{5.13}$$

with $\boldsymbol{T}(t) \in \mathbb{R}^{2\times 12}$. We hence represent the k-th pair of splines as $\boldsymbol{s}_k^T = \begin{bmatrix} \boldsymbol{\alpha}_{kx}^T & \boldsymbol{\alpha}_{ky}^T \end{bmatrix}$. Using this parametrization, we setup an optimization problem that will solve for all the spline coefficients for $k = 0, \ldots, n-1$, with n the number of splines used to generate the motion. This also allows to easily represent velocities and accelerations as

$$\dot{\boldsymbol{p}}(t) = \begin{bmatrix} \boldsymbol{\alpha}_{kx}^T \\ \boldsymbol{\alpha}_{ky}^T \end{bmatrix} \dot{\boldsymbol{\eta}}(t) \qquad \ddot{\boldsymbol{p}}(t) = \begin{bmatrix} \boldsymbol{\alpha}_{kx}^T \\ \boldsymbol{\alpha}_{ky}^T \end{bmatrix} \ddot{\boldsymbol{\eta}}(t) \tag{5.14}$$

or by writing $\dot{\boldsymbol{p}}(t) = \dot{\boldsymbol{T}}(t)\boldsymbol{\alpha}_k$ and $\ddot{\boldsymbol{p}}(t) = \ddot{\boldsymbol{T}}(t)\boldsymbol{\alpha}_k$. The problem we solve is then to search for the vector optimal coefficients

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\alpha}_{0x}^T & \boldsymbol{\alpha}_{0y}^T & \boldsymbol{\alpha}_{1x}^T & \boldsymbol{\alpha}_{1y}^T & \ldots & \boldsymbol{\alpha}_{(n-1)x}^T & \boldsymbol{\alpha}_{(n-1)y}^T \end{bmatrix}^T \tag{5.15}$$

by solving a QP problem. The total number of splines used to approximate the motion is $2n$. The following will describe the setup of an optimization problem formulated as

$$\begin{aligned} \min_{\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{\xi}^T \boldsymbol{Q}\boldsymbol{\xi} + \boldsymbol{c}^T\boldsymbol{\xi} \\ \text{s. t.} \quad & \boldsymbol{A}\boldsymbol{\xi} = \boldsymbol{b}, \quad \boldsymbol{D}\boldsymbol{\xi} \leq \boldsymbol{f}. \end{aligned} \tag{5.16}$$

In the following, if not explicitly noted otherwise, we will refer only to the x coordinate of each pair of splines $\boldsymbol{s}_k$. The same arguments hold for the y coordinate.

5.5.4 *Plan initialization*

The motion plan is initialized with an extended state (position, velocity and acceleration) which is used as a hard constraint for the initial state of the spline sequence. This is done by fusing together the previous desired position ${}_C\boldsymbol{r}_{IB}^{des}$ with the current measured one ${}_C\boldsymbol{r}_{IB}$ by using an exponentially decaying alpha filter designed as

$${}_C\boldsymbol{r}_{IB}^{ref} = \alpha\, {}_C\boldsymbol{r}_{IB}^{des} + (1-\alpha)\, {}_C\boldsymbol{r}_{IB} \tag{5.17}$$

with $\alpha = 0.5\mathrm{e}^{-\lambda t_c}$, where $\boldsymbol{\lambda} \in \mathbb{R}$ can be set to weigh the desired position as a function of the computation time t_c since the last successful optimization. A similar filter is employed to set the initial velocities, whereas acceleration is initialized with the last available reference one.

5.5.5 *Cost function*

In our formulation, several terms contribute to the Hessian matrix $\boldsymbol{Q} \in \mathbb{R}^{12n\times 12n}$ and the linear term $\boldsymbol{c} \in \mathbb{R}^{12n}$ appearing in (5.16). We penalize the deviation of the position $\boldsymbol{p}(t) = \boldsymbol{T}(t)\boldsymbol{\alpha}$ from a desired position reference $\boldsymbol{p}_r$ by writing

$$\begin{aligned} &\frac{1}{2}\|\boldsymbol{T}\boldsymbol{\alpha} - \boldsymbol{p}_r\|_W^2 \\ &\quad = \frac{1}{2}(\boldsymbol{T}\boldsymbol{\alpha} - \boldsymbol{p}_r)^T \boldsymbol{W}(\boldsymbol{T}\boldsymbol{\alpha} - \boldsymbol{p}_r) \\ &\quad = \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{T}^T\boldsymbol{W}\boldsymbol{T}\boldsymbol{\alpha} - \boldsymbol{p}_r^T\boldsymbol{W}\boldsymbol{T}\boldsymbol{\alpha} + \frac{1}{2}\boldsymbol{p}_r^T\boldsymbol{W}\boldsymbol{p}_r. \end{aligned} \tag{5.18}$$

Minimizing the squared norm in (5.18) is equivalent to solving a QP in the form 5.16 with

$$\boldsymbol{Q} = \boldsymbol{T}^T \boldsymbol{W} \boldsymbol{T} \qquad \boldsymbol{c} = -\boldsymbol{T}^T \boldsymbol{W}^T \boldsymbol{p}_r. \tag{5.19}$$

To penalize deviation for velocity references, it is simply required to use $\dot{\boldsymbol{T}}$ and $\dot{\boldsymbol{p}}_r$ in (5.18). A similar reasoning can be done for acceleration deviations.

5.5.5.1 *Acceleration minimization*

As shown in [62], we can minimize the acceleration by writing

$$\boldsymbol{Q}_k^{acc} = \begin{bmatrix} (400/7)t_f^7 & 40t_f^6 & 24t_f^5 & 10t_f^4 & \\ 40t_f^6 & 28.8t_f^5 & 18t_f^4 & 8t_f^3 & \vdots \\ 24t_f^5 & 18t_f^4 & 12t_f^3 & 6t_f^2 & \boldsymbol{0}_{4\times 2} \\ 10t_f^4 & 8t_f^3 & 6t_f^2 & 4t_f & \vdots \\ \cdots & \boldsymbol{0}_{2\times 4} & \cdots & & \boldsymbol{0}_{2\times 2} \end{bmatrix} \tag{5.20}$$

with a null linear term $\boldsymbol{c}_k^{acc} = \boldsymbol{0}$. Note that if this were the only term added to the cost function, there would be no cost associated to the α_{1k} and α_{2k} coefficients of each spline, leading to a positive semi-definite Hessian. This is problematic when using a method such as the Active Set one [61] which requires the Hessian to be positive definite. In that case, a regularizer term can be added as

$$\boldsymbol{Q}_k^{acc_\rho} = \begin{bmatrix} \boldsymbol{0}_{4\times 4} & \boldsymbol{0}_{4\times 2} \\ \boldsymbol{0}_{2\times 4} & \rho \mathbb{I}_{2\times 2} \end{bmatrix} \tag{5.21}$$

with $\rho = 10\mathrm{e}^{-8}$ and a null linear term.

5.5.5.2 *Soft final constraints*

We set the final position $\boldsymbol{p}_f \in \mathbb{R}^2$ to be at the center $\boldsymbol{p}_f^{ref} \in \mathbb{R}^2$ of the polygon defined by the planned footholds, which is the polygon that would support the robot if it would stop to stand at the end of the support polygon sequence. To minimize the norm $\|\boldsymbol{p}_f - \boldsymbol{p}_f^{ref}\|_{\boldsymbol{W}_f}^2 = \|\boldsymbol{A}_f \boldsymbol{s}_f - \boldsymbol{p}_f^{ref}\|_{\boldsymbol{W}_f}^2$ we write

$$\boldsymbol{Q}_f = \boldsymbol{A}_f^T \boldsymbol{W}_f \boldsymbol{A} \qquad \boldsymbol{c}_f = -\boldsymbol{A}^T \boldsymbol{W}_f \boldsymbol{p}_f^{ref}, \tag{5.22}$$

with $\boldsymbol{W}_f \in \mathbb{R}^{2\times 2}$ a symmetric diagonal matrix of positive weights. To avoid solutions in which the optimizer places the final state far away from the

reference position, we add inequality constraints on the position such that it is constrained in a box centered at the reference position.

5.5.5.3 *Deviation from previous solution*

Since we are computing a new optimization as soon as the previous one was successful, to avoid large deviations between successive motion plans we penalize deviations on position, velocity and acceleration obtained by the current solution $\boldsymbol{\xi}$ from the ones resulting from the previous one $\boldsymbol{\xi}_{i-1}$. Given t_{pre} the time elapsed since the last successful optimization, we penalize the deviations $\|\boldsymbol{p}(\bar{t}) - \boldsymbol{p}_{i-1}(t_{pre} + \bar{t})\|^2_{\boldsymbol{W}_{pre}}, \quad \forall \bar{t} \in \left[0, t_f\right]$, where $t_f = \sum_{k=0}^{n-1} t_{fk}$ is the sum of the durations of all of the n splines that compose the motion. We discretize the optimization horizon $\left[0, t_f\right]$ using a sample time dt. We employ a similar cost penalizing deviations from velocities and accelerations obtained by the last solution.

5.5.5.4 *Path regularization*

Continuous update of the motion plan can cause drift in the motion of the torso w.r.t. the reference footholds. This can be caused by the accumulation of control errors, which alter the solution such that the motion can become unfeasible. To avoid this issue, we add a cost on the deviation from a reference *regularizer path*. The path itself is approximated as a sequence of splines represented by $\boldsymbol{\pi}(t)$, $\dot{\boldsymbol{\pi}}(t)$ and $\ddot{\boldsymbol{\pi}}(t)$. The spline coefficients of the path regularizer are obtained from a minimization problem setup such that:

- The initial state $\boldsymbol{\pi}(0)$, $\dot{\boldsymbol{\pi}}(0)$ and $\ddot{\boldsymbol{\pi}}(0)$ coincides with the center of the initial support polygon and is equal to the initial velocity and acceleration
- The final state $\boldsymbol{\pi}(t_f)$, $\dot{\boldsymbol{\pi}}(t_f)$ and $\ddot{\boldsymbol{\pi}}(t_f)$ is set as in section 5.5.5.2
- Acceleration is minimized
- Splines are smoothly connected by setting equality constraints on the state at the spline junctions

By sampling the entire motion as done in section 5.5.5.3, we penalize deviations of the resulting motion from the path regularizer.

5.5.6 *Equality constraints*

Since the entire motion is composed of a sequence of splines, we setup the problem to ensure that they are connected. Since the initial state cannot be modified by the motion plan, we set a hard equality constraint such that the initial state of the first spline $\boldsymbol{s}_0$ coincides with the one set in the plan initialization. The initial hard constraint can be written as $\boldsymbol{p}(0) = \boldsymbol{p}^r$, $\dot{\boldsymbol{p}}(0) = \dot{\boldsymbol{p}}^r$ and $\ddot{\boldsymbol{p}}(0) = \ddot{\boldsymbol{p}}^r$. Hence, we write

$$\begin{bmatrix} \boldsymbol{\eta}(0)^T \\ \dot{\boldsymbol{\eta}}(0)^T \\ \ddot{\boldsymbol{\eta}}(0)^T \end{bmatrix} \boldsymbol{\alpha}_0^x = \begin{bmatrix} \boldsymbol{p}_x^r \\ \dot{\boldsymbol{p}}_x^r \\ \ddot{\boldsymbol{p}}_x^r \end{bmatrix}. \tag{5.23}$$

To ensure that two splines $\boldsymbol{s}_k$ and $\boldsymbol{s}_{k+1}$ are connected, we constrain them such that

$$\begin{bmatrix} \boldsymbol{\eta}(t_{fk})^T & -\boldsymbol{\eta}(0)^T \\ \dot{\boldsymbol{\eta}}(t_{fk})^T & -\dot{\boldsymbol{\eta}}(0)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_k^x \\ \boldsymbol{\alpha}_{k+1}^x \end{bmatrix} = \mathbf{0}, \tag{5.24}$$

with t_{fk} representing the duration in seconds of spline $\boldsymbol{s}_k$. When the equality junction connecting two splines lies between two disjoint support polygons, we cannot guarantee the smoothness or the motion anymore, but will have to allow the ZMP to jump, which will cause a jump in the acceleration reference. Although this has a negative effect on the controller, it does allow the optimizer to find a solution when traversing disjoint support polygons. We check whether two polygons intersect by using the *Separating Axis Theorem* (SAT) described in [69]. If such an intersection exists, we also constrain acceleration between two splines by writing

$$\begin{bmatrix} \ddot{\boldsymbol{\eta}}(t_f)^T & -\ddot{\boldsymbol{\eta}}(0)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_k^x \\ \boldsymbol{\alpha}_{k+1}^x \end{bmatrix} = \mathbf{0}. \tag{5.25}$$

5.5.7 *Inequality constraints*

As demonstrated in [70] and [63], balance during locomotion can be ensured by constraining the *Zero-Moment Point* (ZMP) to lie inside the support polygon. Starting from the measured feet configuration at the time the motion plan is called and using the planned feet positions, we compute a sequence of support polygons and their duration by using the contact schedule as defined in [22]. By assuming at least two feet on the ground

at all times, the support polygon for a quadrupedal robot can be a line, a triangle or a quadrilateral.

As discussed in [43] and [63], the location of the ZMP is a function of the motion of the center of mass. The location of the x coordinate of the ZMP is defined by

$$x_{zmp} = x_{com} - \frac{z_{com}\ddot{x}_{com}}{g + \ddot{z}_{com}}, \tag{5.26}$$

where g is the gravitational acceleration. We reorder the vertices of each support polygon counter-clockwise by computing their polar coordinates w.r.t. the center of the support polygon and comparing their phase. We can then constrain the ZMP by adding one constraint for each edge of the support polygon by writing $ax_{zmp} + by_{zmp} + c \geq 0$, where a, b and c are the coefficients of the line that passes through the vertices of an edge of the support polygon. The normal vector to this line $\boldsymbol{n} = \begin{bmatrix} a & b \end{bmatrix}^T$ is directed towards the inside of the polygon.

As mentioned in section 5.5.5, we place additional inequality constraints on the final position $\boldsymbol{p}_f$ of the motion. This is obtained by adding four constraints of the form (5.26) on $\boldsymbol{p}_f$.

5.5.8 *Constraint relaxation*

The optimizer can fail if the constraints are too tight. First, the initial ZMP may not lie in the current support polygon. For this reason, we exclude the ZMP constraint on the first sample of the motion plan. This way the optimizer is still able to find a solution, and our experiments have shown this to work on the real system.

Second, the support polygon safety margin may be too high. To counteract this, whenever an optimization fails, we iteratively decrease the margin by a fixed amount, and we increase it back at a slower rate to ensure the success of the optimization.

Finally, we check for the time duration t_{fk} associated to each polygon. If it is smaller or comparable to the sample time used to discretize the motion and to setup the constraints, we remove the support polygon from the sequence.

5.6 EXPERIMENTS

5.6.1 *Setup*

Our experiments[1] were conducted on ANYmal [56], an accurately torque-controllable quadrupedal robot. Control signals are generated in a 400Hz control loop which runs on a dedicated on-board computer together with state estimation [57] and communication with the drives. For modeling and computation of kinematics and dynamics, we use the open-source Rigid Body Dynamics Library [58] (RBDL), which is a C++ implementation of the algorithms described in [59]. To numerically solve the QP problems described in the previous sections, we use a custom version of the QuadProg++ [60] library, a C++ open-source QP solver which implements the Active Set algorithm described in [61].

5.6.2 *Dynamic locomotion*

We have tested our planning and control framework by executing several gaits, including a dynamic lateral walk with overlapping swing phases; a trotting gait; a pacing gait; a transition between dynamic lateral walk and pacing (see Fig.5.7). The execution of all these gaits was successful both on flat terrain and on slightly uneven terrain, including locomotion over a small but unperceived step. We have also tested an online transition from stand to a dynamic lateral walk to a pace. This was achieved by applying a transition from the walking to the pacing gait by interpolating the two footfall patterns as a function of the commanded heading speed. As shown in the videos, we can smoothly switch from stand, to dynamic walk to pace and safely back to stand.

5.7 CONCLUSIONS AND FUTURE WORK

We have shown how it is possible to execute highly dynamic and agile gaits on a torque-controllable quadruped robot by combining a motion planner based on a simplified dynamic model and a hierarchical whole-body controller. The motion planner based on the ZMP stability criterion is continuously updating the reference trajectory as a function of the current gait pattern and the robot state; the whole-body controller tracks the reference plan by solving a hierarchy of tasks. We have demonstrated the

1 https://youtu.be/iUQE-ZQqdJY

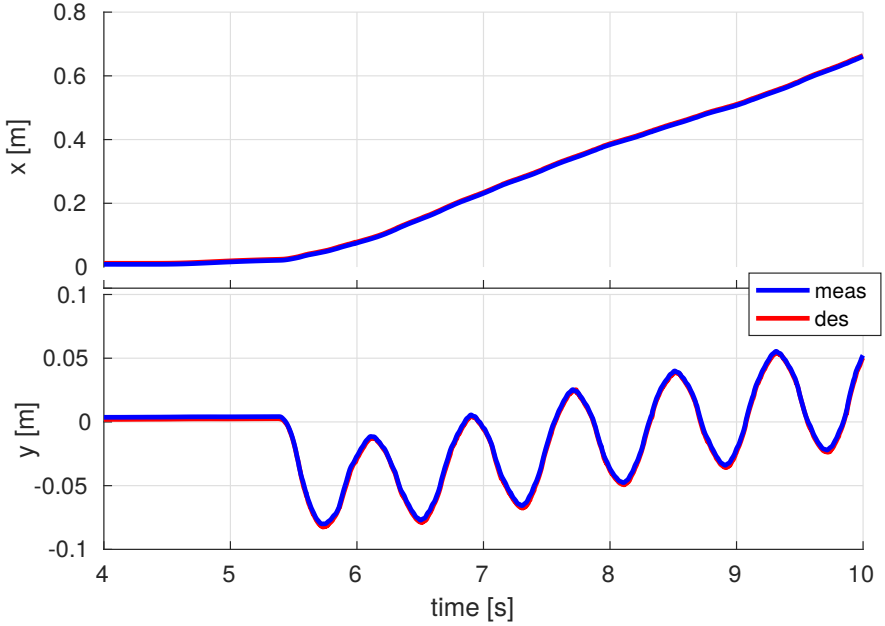


FIGURE 5.5: The time evolution of the measured (blue) and reference (red) whole-body center of mass position on the x and y directions during a pacing gait.

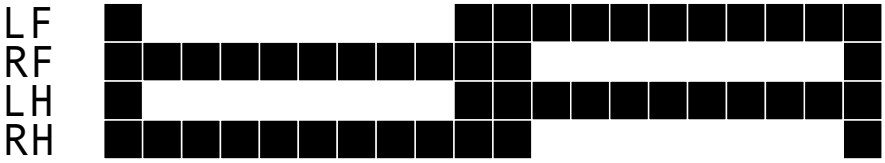


FIGURE 5.6: The contact schedule associated to a pacing gait. This gait exhibits a complete overlap in the swing phase of two legs which are on the same side of the main body, namely fore/hind left or fore/hind right legs.

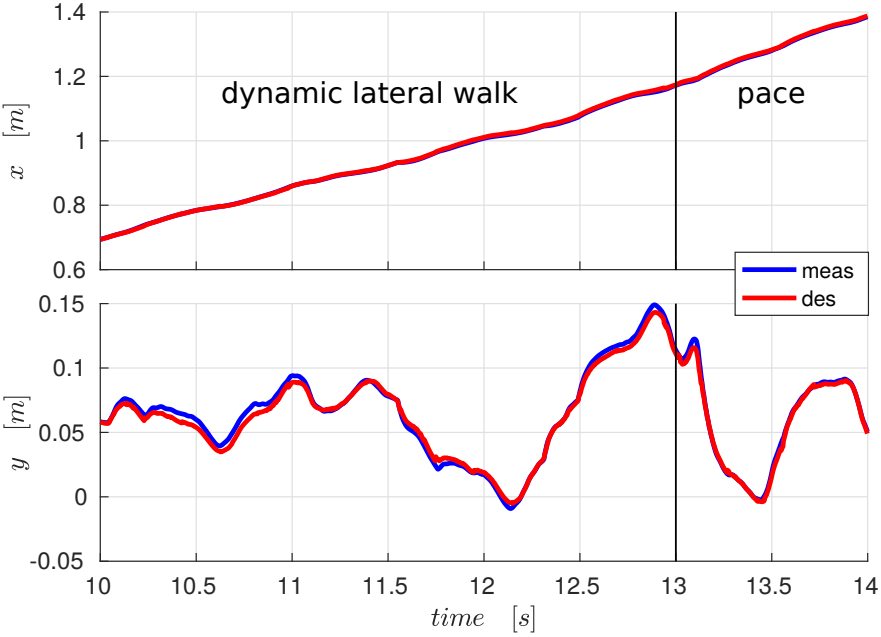


FIGURE 5.7: The time evolution of the measured and reference whole-body center of mass position in the inertial frame during a transition from dynamic lateral walk to pace.

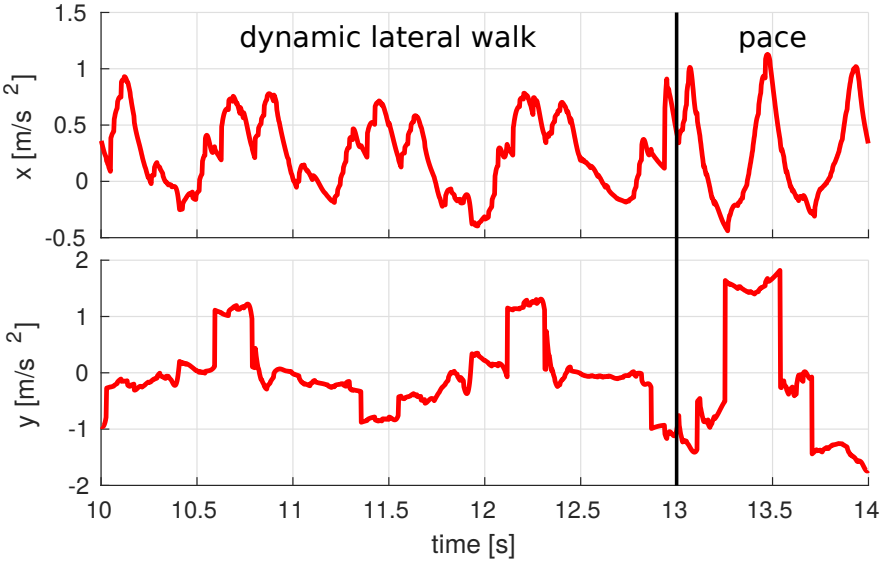


FIGURE 5.8: The time evolution of the reference acceleration for the whole-body center of mass during a transition from dynamic lateral walk to pace.

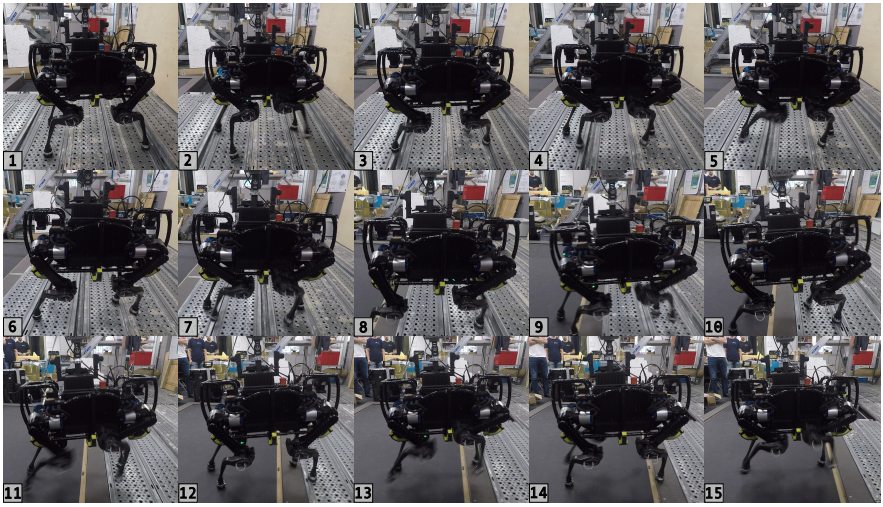


FIGURE 5.9: A sequence showing ANYmal during a transition from a dynamic lateral walk to a pacing gait. Balance is retained even though there is an unperceived step.

execution of dynamic gaits such as a trot, a dynamic lateral walk, a pace and a transition between the last two on ANYmal. To the best of our knowledge, this is the first time that experimental results are published for such gaits.

The next steps will be focused on improving the main features of gait execution and gait transition. This includes foothold evaluation and gait pattern adaptation in order to be able to deal with harsh changes in the state of the robot and sudden changes in the active gait pattern. The framework should also be extended to include full flight phases, which would allow to perform even more agile locomotion.

6

PAPER 3: DYNAMIC LOCOMOTION THROUGH ONLINE NONLINEAR MOTION OPTIMIZATION FOR QUADRUPEDAL ROBOTS

6.1 ABSTRACT

This paper presents a realtime motion planning and control method which enables a quadrupedal robot to execute dynamic gaits including trot, pace and dynamic lateral walk, as well as gaits with full flight phases such as jumping, pronking and running trot. The proposed method also enables smooth transitions between these gaits. Our approach relies on an online ZMP based motion planner which continuously updates the reference motion trajectory as a function of the contact schedule and the state of the robot. The reference footholds for each leg are obtained by solving a separate optimization problem. The resulting optimized motion plans are tracked by a hierarchical whole-body controller. Our framework has been tested in simulation and on ANYmal, a fully torque-controllable quadrupedal robot, both in simulation and on the actual robot.

6.2 INTRODUCTION

Robotic locomotion is a complex problem that involves a multi-body system interacting with the environment through multiple contact points.It becomes even more demanding when locomotion is executed in an unknown environment and under the presence of external disturbances. While animals have shown to be able to skillfully cope with these limitations, their robotic counterparts are still struggling to robustly locomote out of lab environments. Our goal is to have robots which are capable of safely executing dynamic locomotion over unstructured terrain. This would turn out to be extremely helpful in search and rescue scenarios, where tasks are carried out in an unpredictable environment and fast and stable locomotion can be essential. The capability of executing different kinds of gaits makes legged robots even more adaptable to different scenarios. A static walk might be the only option where careful stepping is the most critical requirement; running might be necessary if speed is the decisive

factor for a successful mission; dynamic lateral walk might be the most energy efficient option at relatively low speeds. From these considerations, it emerges that the desired skills of a walking robot include not only the ability to execute dynamic gaits but also the ability to switch between the gaits. Although robots are not yet fully capable of autonomous locomotion in such environments, the past years have shown that both the academia and the industry are taking big steps in the development of tools and methods which would allow this.

The *Cheetah 2* [64], a quadrupedal robot developed at MIT, has demonstrated its agile skills by jumping over unexpected obstacles using a fast online *Model Predictive Controller* (MPC). The IIT *HyQ* [11] quadruped has shown walking using a creeping gait [67] over stepping stones [63]. In [44], the trajectory optimization framework is exploited to execute dynamic gaits such as trotting and pacing. Agile and dynamic skills have been demonstrated by the Boston Dynamics robots, including the quadrupedal robot *Spot* and the more recent *Spot Mini*, which additionally sports a robot arm for manipulation tasks. However, very little is known about the methods which enable such remarkable capabilities. In [71], agile running motions have been demonstrated on the humanoid robot Toro [72]. Although the targeted robot platform is topologically different (i.e. humanoid versus quadruped), the problem formulation introduced there (e.g. parametrization of the motion with quintic splines; dynamic model used when in full flight phase) shares many similarities with ours. While impressive in the results, their approach is focused on the planning and execution of running gaits on flat terrain or on stepping stones. We demonstrate, on the other hand, a single framework which deals with a larger variety of gaits, from a static walk up to a running trot and pace.

Over the past few years, *ANYmal* [56] (see Fig.6.1) has shown its ability to walk [65], trot [22] and climb over stairs [66]. In our previous work [73], we have advanced the capabilities of ANYmal by demonstrating dynamic lateral walking, pacing and gait transitions between these gaits. The method relied on a simplified and linearized *Zero-Moment Point* (ZMP, [9]) model to generate planar Center of Mass (CoM) motion plans, which were executed in an MPC-like fashion, relying on a hierarchical whole-body controller to track the reference motion. That approach, while novel, presented limitations both in the variety of gaits that could be produced and tracked, and in the simple approach which allowed us to switch between those gaits.

In this paper, we discuss our approach to overcome those limitations. First, we discuss the solution to an optimization problem which is solved

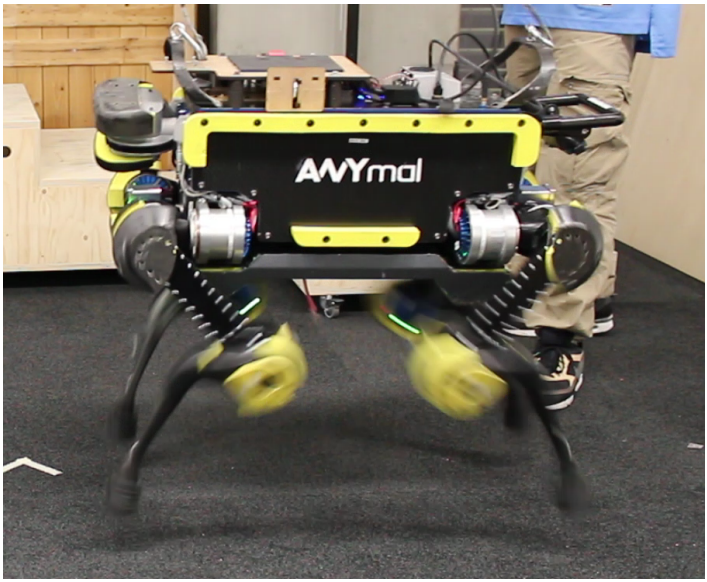


FIGURE 6.1: ANYmal, a torque-controllable quadrupedal robot, executes a squat jump. The motion is computed by solving a nonlinear optimization problem which ensures dynamically stable plans.

online to generate motion plans for dynamic gaits such as trotting, walking, and pacing, as well as for more agile maneuvers such as jumping and running trot or pace; second, we introduce a separate online optimization problem which computes reference footholds; finally, we discuss a gait switching algorithm which allows us to switch between any of the gaits that are discussed in this paper. We show the results of our approach both in a simulated environment and on the quadrupedal robot ANYmal.

The frequency at which a motion plan is computed can vary depending on the type of gait and on the optimization parameters. As shown in our experiments, the optimization can be evaluated at more than 100Hz. The motion plans are tracked by a whole-body controller which runs at an update frequency of 400Hz.

6.3 MODEL FORMULATION

The model of a walking robot can be described as a free-floating base B to which limbs are attached. The motion of the entire system can be described w.r.t. a fixed inertial frame I. The position of the Base w.r.t. the inertial frame is written as ${}_I\boldsymbol{r}_{IB} \in \mathbb{R}^3$. The orientation $\boldsymbol{q}_{IB} \in SO(3)$ of the Base w.r.t. the inertial frame is parametrized using Hamiltonian unit quaternions. The limb joint angles are stacked in the vector $\boldsymbol{q}_j \in \mathbb{R}^{n_j}$. We write the generalized coordinates vector $\boldsymbol{q}$ and the generalized velocities vector $\boldsymbol{u}$ as

$$\boldsymbol{q} = \begin{bmatrix} {}_I\boldsymbol{r}_{IB} \\ \boldsymbol{q}_{IB} \\ \boldsymbol{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \quad \boldsymbol{u} = \begin{bmatrix} {}_I\boldsymbol{v}_B \\ {}_B\boldsymbol{\omega}_{IB} \\ \dot{\boldsymbol{q}}_j \end{bmatrix} \in \mathbb{R}^{n_u}, \tag{6.1}$$

where $n_u = 6 + n_j$, ${}_I\boldsymbol{v}_B$ and ${}_B\boldsymbol{\omega}_{IB}$ are the linear and angular velocity of the Base w.r.t. the inertial frame expressed respectively in the I and B frame. The equations of motion of mechanical systems which are in contact with the environment can be written as $\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}_s^T\boldsymbol{\lambda}$, where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n_u \times n_u}$ is the mass matrix and $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) \in \mathbb{R}^{n_u}$ is the vector of Coriolis, centrifugal and gravity terms. The selection matrix $\boldsymbol{S} = \begin{bmatrix} \boldsymbol{0}_{n_\tau \times (n_u - n_\tau)} & \mathbb{I}_{n_\tau \times n_\tau} \end{bmatrix}$ selects which Degrees of Freedom (DoFs) are actuated. If all limb joints are actuated, then $n_\tau = n_j$. The vector of constraint forces $\boldsymbol{\lambda}$ is mapped to the joint space torques through the support Jacobian $\boldsymbol{J}_s \in \mathbb{R}^{3n_c \times n_u}$, which is obtained by stacking the constraint Jacobians as $\boldsymbol{J}_s = \begin{bmatrix} \boldsymbol{J}_{C_1}^T & \cdots & \boldsymbol{J}_{C_{n_c}}^T \end{bmatrix}^T$, with n_c the number of limbs in contact.

6.4 MOTION OPTIMIZATION

Our motion planning framework consists of two main modules, namely a CoM motion planner and a foothold generator, both of which are separately solving an optimization problem. The computation is carried out in parallel to the the main control loop, such that it does not interrupt the whole-body motion tracking controller. Motion plans for the swing feet are obtained by generating splines through the current position of the foot and its reference optimized foothold.

The motion plans are computed in the *Plan frame P*, which is located at the footprint center projected onto the local terrain along the terrain normal. The footprint is represented by the location of the feet of the robot. The orientation of the Plan frame is such that it is parallel to the local estimation of the terrain from the contact points [52], while its yaw angle is computed s.t. the x axis is aligned with the reference high-level velocity. The latter is generated from an external source, e.g. an operator device or a high-level navigation planner. The Plan frame allows the optimization parameters (e.g. weights relative to a specific coordinate to the CoM) to be independent of the inertial frame and to be properly defined w.r.t. the desired direction of motion.

To drive locomotion to the reference speed, footholds are generated (section 6.4.3) for all legs such that the robot can achieve the desired high-level velocity on average. This information, together with the contact schedule (i.e. the predefined lift-off and touch-down timings for each leg), is used to generate a sequence of support polygons (section 6.4.4) which are sent to the motion plan optimizer (section 6.4.1). This in turn will produce position, velocity and acceleration profiles for the whole-body center of mass. Fig.6.2 depicts an overview of the entire motion planning and tracking framework.

We build on our previous work [73] which described how to produce a motion plan for the x and y coordinates of the Center of Mass (CoM) which can be tracked by a quadrupedal robot. In that case, the reference height of the CoM was set to a user-specified value and tracked by the controller. To optimize for gaits which include full flight phases, the optimizer must know how to produce motion plans which will be projected to contact forces which produce jumping motions. To do this, we include the z coordinate of the CoM in the optimization, such that we optimize for the position, velocity and acceleration of the CoM of the system. This has important implications on the problem formulation. First, the dimension

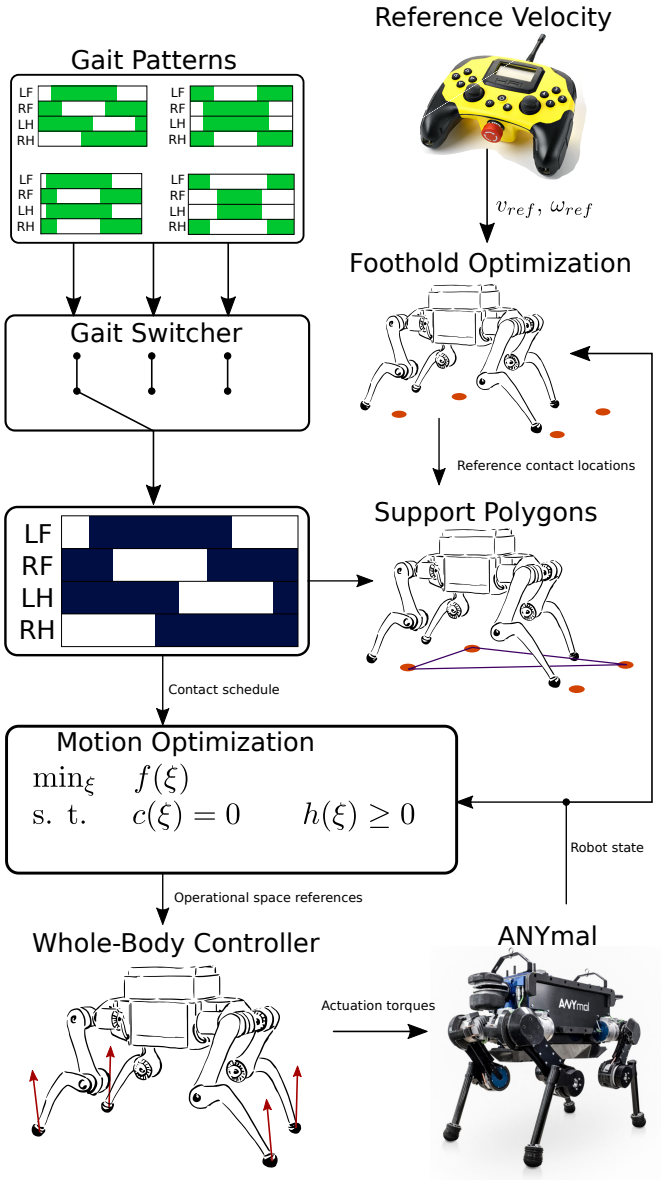


FIGURE 6.2: An overview of the planning and control architecture described in this paper.

of the optimization problem is increased. Then, by dropping the assumption that the z component of the center of mass acceleration $\ddot{\boldsymbol{p}}_{CoM}$ is zero, constraints on the ZMP model become nonlinear. Although the method described in this paper is more complex (thus, more computationally expensive) than the one that we build on, we are still capable of computing the plans online and executing them in a MPC-like fashion, i.e. we continuously run the motion optimizer initialized with the most recent state of the robot to update the motion plan that is tracked by the whole-body motion controller.

6.4.1 *Problem formulation*

We describe each coordinate of the CoM motion plan as a sequence of quintic splines. As it will be explained in detail in the following sections, the motion plan that we are optimizing is described over a sequence of support polygons which depend on the type of gait that is being executed. In our implementation, for each component of the CoM there is at least one spline for each support polygon. Two splines are used for full flight phases. The motion of one of the coordinates of the CoM, for example the x component, can be described by the i-th spline as

$$\begin{aligned} x(t) &= \alpha_{i5}^x t^5 + \alpha_{i4}^x t^4 + \alpha_{i3}^x t^3 + \alpha_{i2}^x t^2 + \alpha_{i1}^x t + \alpha_{i0}^x \\ &= \begin{bmatrix} t^5 & t^4 & t^3 & t^2 & t & 1 \end{bmatrix} \\ &\quad \cdot \begin{bmatrix} \alpha_{i5}^x & \alpha_{i4}^x & \alpha_{i3}^x & \alpha_{i2}^x & \alpha_{i1}^x & \alpha_{i0}^x \end{bmatrix}^T \\ &= \boldsymbol{\eta}^T(t)\boldsymbol{\alpha}_i^x, \end{aligned} \tag{6.2}$$

with $t \in [\bar{t}, \bar{t} + \Delta t_i]$, where $\bar{t}$ is the sum of time durations of all splines up to the $(i-1)$-th one, and Δt_i is the time duration of the i-th spline.

Using the notation introduced, we can compactly write velocity and acceleration as

$$\dot{x}(t) = \dot{\boldsymbol{\eta}}^T(t)\boldsymbol{\alpha}_i^x \qquad \ddot{x}(t) = \ddot{\boldsymbol{\eta}}^T(t)\boldsymbol{\alpha}_i^x, \tag{6.3}$$

where we have used the time vectors

$$\begin{aligned} \dot{\boldsymbol{\eta}}(t) &= \begin{bmatrix} 5t^4 & 4t^3 & 3t^2 & 2t & 1 & 0 \end{bmatrix}^T \\ \ddot{\boldsymbol{\eta}}(t) &= \begin{bmatrix} 20t^3 & 12t^2 & 6t & 2 & 0 & 0 \end{bmatrix}^T. \end{aligned} \tag{6.4}$$

Similar considerations can be carried out for the y and z components of the CoM. The vector of optimization parameters is then defined as

$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_0^T \cdots \boldsymbol{\alpha}_i^T \cdots \boldsymbol{\alpha}_{n_s}^T \end{bmatrix}^T$, with $3n_s$ the total number of splines and $\boldsymbol{\alpha}_i = \begin{bmatrix} \boldsymbol{\alpha}_i^{x\,T} & \boldsymbol{\alpha}_i^{y\,T} & \boldsymbol{\alpha}_i^{z\,T} \end{bmatrix}^T$. By also defining

$$\boldsymbol{T}(t) = \begin{bmatrix} \boldsymbol{\eta}^T(t) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\eta}^T(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\eta}^T(t) \end{bmatrix}, \tag{6.5}$$

we can easily write the CoM position $\boldsymbol{p}_{CoM}(t) = \boldsymbol{T}(t)\boldsymbol{\alpha}_i$, where $\boldsymbol{p}_{CoM} \in \mathbb{R}^3$. Velocity and acceleration vectors can be written as $\dot{\boldsymbol{p}}_{CoM}(t) = \dot{\boldsymbol{T}}(t)\boldsymbol{\alpha}_i$ and $\ddot{\boldsymbol{p}}_{CoM}(t) = \ddot{\boldsymbol{T}}(t)\boldsymbol{\alpha}_i$ respectively.

6.4.2 *Center of mass optimization*

Due to the nature of the constraints that will be described in the following sections, we formulate the motion planning algorithm as a nonlinear optimization problem which minimizes a generic cost function $\boldsymbol{f}(\boldsymbol{\xi})$ subject to equality and inequality constraints $\boldsymbol{c}(\boldsymbol{\xi}) \leq \mathbf{0}$ and $\boldsymbol{h}(\boldsymbol{\xi}) = \mathbf{0}$. To numerically solve the optimization problem, we use the *Sequential Quadratic Programming* (SQP) algorithm [74], which requires the computation of Jacobians and Hessians of the constraints and the cost function. The optimization, continuously re-evaluated as soon as the previous one is completed, provides a motion plan over a time horizon of τ seconds, where τ is the periodicity of the locomotion gait (e.g. walk or trot) that is being optimized for.

In the following, we describe each single contribution to the cost function and the setup of all constraints used in our formulation.

6.4.2.1 *Cost function*

As done in [63] and [73], we minimize the acceleration of the entire motion plan. To do this, we set a quadratic cost computed for spline segment as

$$\boldsymbol{Q}_k^{acc} = \begin{bmatrix} (400/7)\Delta t_k^7 & 40\Delta t_k^6 & 24\Delta t_k^5 & 10\Delta t_k^4 & \\ 40\Delta t_k^6 & 28.8\Delta t_k^5 & 18\Delta t_k^4 & 8\Delta t_k^3 & \\ 24\Delta t_k^5 & 18\Delta t_k^4 & 12\Delta t_k^3 & 6\Delta t_k^2 & \mathbf{0}_{4\times 2} \\ 10\Delta t_k^4 & 8\Delta t_k^3 & 6\Delta t_k^2 & 4\Delta t_k & \\ & \mathbf{0}_{2\times 4} & & & \mathbf{0}_{2\times 2} \end{bmatrix} \tag{6.6}$$

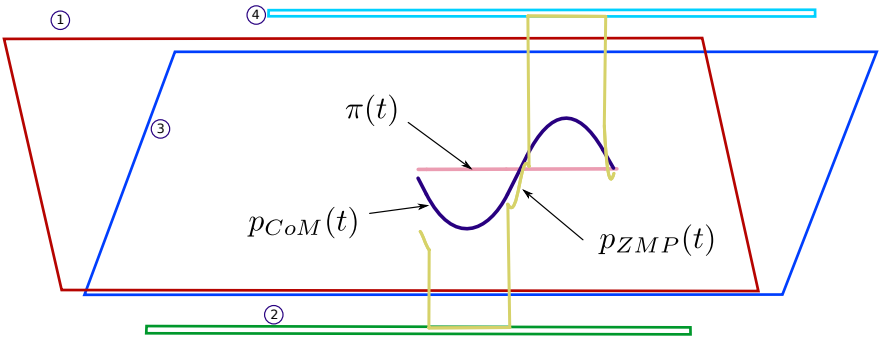


FIGURE 6.3: The top view of a motion plan computed for a pace (both lateral legs simultaneously swinging). The blue curve is the desired CoM position, the yellow curve the resulting ZMP position, and the pink curve is the path regularizer. The support polygon sequence is numbered from 1 to 4.

and $\boldsymbol{c}_k^{acc} = \mathbf{0}$, with Δt_k the time duration of the k-th spline segment in seconds. As explained in [62], the Hessian matrix $\boldsymbol{Q}_k^{acc}$ in eq.(6.6) is obtained by squaring and integrating the acceleration of the CoM over the time duration of the k-th spline. $\boldsymbol{Q}_k^{acc}$ is added as a sub-matrix to the Hessian of the overall cost function.

Soft constraints are set to impose constraints on the initial and final state of the whole motion plan. The initial acceleration is read from that of the previous optimization. The final desired position is computed as a function of the reference high-level linear velocity $\boldsymbol{v}_{ref}$ and the z-component of the angular velocity $\boldsymbol{\omega}_{ref}$ and the optimization horizon τ. We assume that these are constant over the optimization horizon (which is a valid assumption if they change at a much slower rate than the optimization update frequency), hence we can compute the final position as

$$\boldsymbol{p}_{final} = \boldsymbol{p}_{init} + \boldsymbol{R}(\tau\hat{\boldsymbol{\omega}}_z)(\tau\boldsymbol{v}_{ref}), \tag{6.7}$$

where the second term is the rotation of a position vector computed as a velocity dependent offset vector and $\hat{\boldsymbol{\omega}}_z = \begin{bmatrix} 0 & 0 & \boldsymbol{\omega}_{ref_z} \end{bmatrix}^T$.

External disturbances and continuous updates of the motion plan can cause a drift of the floating base of the robot w.r.t. the reference footholds over time. For this reason, we minimize the deviation between the motion plan w.r.t. to a *path regularizer* $\boldsymbol{\pi}$ (see Fig.6.3) which represents a high-level approximation of the motion plan. The initial position of $\boldsymbol{\pi}$ is computed as the footprint center averaged over the optimization horizon τ, where it is

assumed that the feet are either in contact or move with a constant velocity. The final position is computed through the reference high-level velocity as in (6.7). We set the initial velocity equal to the reference velocity v_{ref}, and the final one to be aligned with the predicted torso orientation at the end of the motion plan. The initial and final accelerations of π are set to zero. The initial and final height of π are set to a user specified reference value.

Typically, the motion plan shows large deviations w.r.t. the path regularizer. Overshoots in the vertical direction can cause violation of kinematic limits in the legs. To limit this issue, we bound overshoots along the z axis by adding a cost in the form $\lambda_{lin}\epsilon_z + \lambda_{quad}\epsilon_z^2$ and augmenting our formulation with the constraints

$$\begin{aligned} p^z_{CoM}(t) - \pi_z(t) &\leq \epsilon_z \\ p^z_{CoM}(t) - \pi_z(t) &\geq -\epsilon_z \\ \epsilon_z &\geq 0, \end{aligned} \tag{6.8}$$

where ϵ is a slack variable which is added to the vector of optimization variables ξ. Due to the time dependency, the trajectory needs to be sampled where each sample point introduces two additional inequality constraints.

6.4.2.2 *Equality constraints*

To ensure that each pair of adjacent splines is connected, we set junction constraints. Using the notation introduced in section 6.4.1, we write the junction constraints for the x coordinate between the k-th and $(k+1)$-th spline as

$$\begin{bmatrix} \eta(t_{fk})^T & -\eta(0)^T \\ \dot{\eta}(t_{fk})^T & -\dot{\eta}(0)^T \end{bmatrix} \begin{bmatrix} \alpha^x_k \\ \alpha^x_{k+1} \end{bmatrix} = \mathbf{0}, \tag{6.9}$$

with t_{fk} representing the duration in seconds of spline s_k. Similarly, (6.9) can be written for the y and z coordinates.

Junction conditions are formulated differently if two spline segments are connected through a *full flight phase*. In this case, the dynamics of the CoM are described by $\ddot{p}_{CoM} = g$, where g is the gravity acceleration vector. Integrating the dynamics yields

$$\begin{aligned} \dot{p}(t) &= gt + \dot{p}_{CoM}(0) \\ p(t) &= \frac{1}{2}gt^2 + \dot{p}_{CoM}(0)t + p_{CoM}(0). \end{aligned} \tag{6.10}$$

The constraints at touch down can be written as a function of the spline coefficients as

$$\begin{aligned}\ddot{\boldsymbol{T}}(0)\boldsymbol{\alpha}_{i+1} &= \boldsymbol{g}\\ \dot{\boldsymbol{T}}(0)\boldsymbol{\alpha}_{i+1} &= \boldsymbol{g}t_f + \dot{\boldsymbol{T}}(t_i)\boldsymbol{\alpha}_i\\ \boldsymbol{T}(0)\boldsymbol{\alpha}_{i+1} &= \frac{1}{2}\boldsymbol{g}t_f^2 + \left[\dot{\boldsymbol{T}}(t_i)t_f + \boldsymbol{T}(t_i)\right]\boldsymbol{\alpha}_i\end{aligned} \tag{6.11}$$

with t_i indicating the duration of the i spline. Similar replacements can be found for initial and final conditions if the first or the last support polygon corresponds to a full flight phase.

6.4.2.3 *Inequality constraints*

As shown in [43], the ZMP position $\boldsymbol{p}_{ZMP}$ w.r.t. the origin of the plan frame O is described by

$$\boldsymbol{p}_{ZMP} = \frac{\boldsymbol{n} \times \boldsymbol{M}_O^{gi}}{\boldsymbol{n}^T \boldsymbol{F}^{gi}}, \tag{6.12}$$

where $\boldsymbol{M}_O^{gi}$ and $\boldsymbol{F}^{gi}$ are the components of the so-called *gravito-inertial* wrench [75], which are computed as

$$\begin{aligned}\boldsymbol{M}_O^{gi} &= m \cdot \boldsymbol{p}_{CoM} \times (\boldsymbol{g} - \ddot{\boldsymbol{p}}_{CoM}) - \dot{\boldsymbol{L}}\\ \boldsymbol{F}^{gi} &= m \cdot \boldsymbol{g} - \dot{\boldsymbol{P}},\end{aligned} \tag{6.13}$$

with m being the mass and $\boldsymbol{P}$ and $\boldsymbol{L}$ the linear and angular momentum at the CoM. In the following, we will approximate $\dot{\boldsymbol{L}} = \boldsymbol{0}$ as we do not optimize for rotations and their derivatives.

To ensure dynamic stability of the planned motions, we constrain the ZMP to always lie in the support polygon. To do this, we write constraints in the form

$$\boldsymbol{h}_{ZMP} = \boldsymbol{d}^T \boldsymbol{p}_{ZMP} + r \geq \boldsymbol{0}, \tag{6.14}$$

where $\boldsymbol{d}^T = \begin{bmatrix} p & q & 0 \end{bmatrix}$ and r are the coefficients which describe an edges that belongs to the support polygon. Substituting (6.12) into (6.14) we get

$$\begin{aligned}&\boldsymbol{d}^T \boldsymbol{S}(\boldsymbol{n}) \boldsymbol{M}_O^{gi} + r\boldsymbol{n}^T \boldsymbol{F}^{gi}\\ =&\boldsymbol{d}^T \boldsymbol{S}(\boldsymbol{n}) \boldsymbol{S}(\boldsymbol{p}_{CoM})(\boldsymbol{g} - \ddot{\boldsymbol{p}}_{CoM}) + r\boldsymbol{n}^T (\boldsymbol{g} - \ddot{\boldsymbol{p}}_{CoM}) \geq \boldsymbol{0}\end{aligned} \tag{6.15}$$

where $\boldsymbol{S}(\boldsymbol{a})$ is a skew symmetric matrix computed such that $\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$.

Using (6.15), we compute the gradient $\nabla \boldsymbol{h}_{ZMP}$ of (6.14) w.r.t. the position and acceleration of the com $\boldsymbol{p}_{CoM}, \ddot{\boldsymbol{p}}_{CoM}$ by writing

$$\nabla \boldsymbol{h}_{ZMP} = \begin{bmatrix} \boldsymbol{\Gamma} \cdot (\ddot{\boldsymbol{p}}_{CoM} - \boldsymbol{g}) \\ -\boldsymbol{\Gamma} \cdot \boldsymbol{p}_{CoM} - r\boldsymbol{n} \end{bmatrix} \tag{6.16}$$

with $\boldsymbol{\Gamma} = \boldsymbol{S}\left(\boldsymbol{S}^T(\boldsymbol{n})\boldsymbol{d}\right)$, and the Hessian $\nabla^2 \boldsymbol{h}_{ZMP}$ as

$$\nabla^2 \boldsymbol{h}_{ZMP} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{\Gamma}^T \\ -\boldsymbol{\Gamma}^T & \boldsymbol{0} \end{bmatrix}, \tag{6.17}$$

The Hessian in (6.17) is anti-symmetric, hence it drops out from the optimization.

We soften the inequality constraints of the initial n_{ineq} samples, with n_{ineq} a tuning parameter set by the user. To achieve this, we add slack variables ϵ_{ineq} to the optimization parameters $\boldsymbol{\xi}$. The formulation is modified by adding the cost $\lambda_{lin}\epsilon_{ineq} + \lambda_{quad}\epsilon_{ineq}^2$ and appending the constraints $\boldsymbol{c}_{ineq} \geq -\epsilon_{ineq}, \quad \epsilon_{ineq} \geq 0$, where $\boldsymbol{c}_{ineq}$ are the first n_{ineq} constraints described in (6.15). From a physical point of view the relaxation amounts to a variable sized support polygon that cannot be smaller than the nominal one.

6.4.2.4 *Assigning a new plan*

When a new motion plan is obtained and sent to the controller, it is important to properly append it to the previously executed plan. To do this, we first store the computation time t_c that was necessary for the solver to optimize. We use this as an initial guess to search for the position in the motion plan that is closest to the current measured one, such that the transition to the new plan is a smooth one. To do this, we solve a line search by writing

$$t^* = \arg\min_t \|\boldsymbol{p}(t) - \boldsymbol{p}_{meas}\|^2_{\boldsymbol{W}} \tag{6.18}$$

where $\boldsymbol{W}$ is a diagonal positive definite weighting matrix and $\boldsymbol{p}_{meas}$ is the measured CoM position after completing the optimization. We use a Newton method with a back-tracing line search algorithm that usually converges within 3 iterations and leads to an average correction of 20% of the initial spline time guess. During experiments with ANYmal, it has been found that a line search significantly increases the control performance.

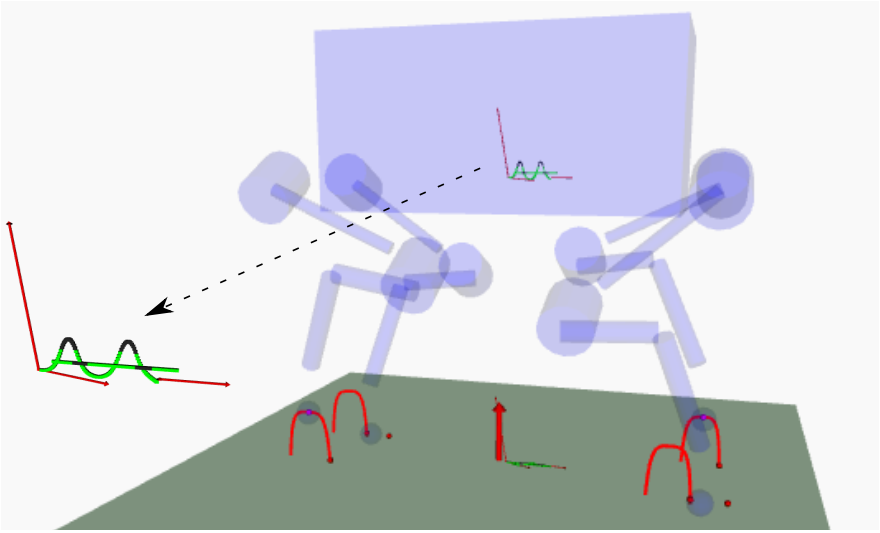


FIGURE 6.4: A motion plan computed for a running trot, which includes a full flight phase. The computed motion plan (shown in close-up) sets vertical accelerations which cause the robot to jump.

6.4.3 *Foothold optimization*

In our previous works ([22], [73]) we have computed reference footholds using a linear inverted pendulum model [22] which predicts the next foothold as a function of the measured and desired velocity of the torso. This method has shown to add robustness to the real system in case of external disturbances or deviation from the reference motion. However, we wish to embed it into a framework which allows us to take into account different contributions to the computation of the footholds, as well as to set constraints on the final result. For this reason, we set up a Quadratic Programming (QP) problem as

$$\min_{\xi} \quad \frac{1}{2}\xi^T Q\xi + c^T\xi \qquad \text{s. t.} \qquad D\xi \leq f. \tag{6.19}$$

where $\xi \in \mathbb{R}^{2n_{feet}}$ is a vector of x and y components of footholds p_{f_i}, with $i = 1, \ldots, n_{feet}$, and $n_{feet} = 4$ the total number of feet. Similarly to what is done for the CoM motion planner, we optimize in parallel to the main control loop. Thus we update the *foothold plan* whenever a new

optimization is ready. In the following we describe each contribution to the cost function in (6.19) as well as the constraints which appear in it.

6.4.3.1 *Cost function*

A user-defined default foot position can be assigned which is relative to a default standing configuration. This can be interpreted as a regularization term in the foothold optimization. To track these default foothold locations $\boldsymbol{p}_{ref_i}$, we add

$$\boldsymbol{Q}_{def_i} = \boldsymbol{W}_{def}, \qquad \boldsymbol{c}_{def_i} = -\boldsymbol{W}_{def}^T \boldsymbol{p}_{ref_i} \tag{6.20}$$

to the cost function described in (6.19). The choice of the default footholds will influence how wide the footprint will be when all legs are in contact with the environment. While this can make a trotting gait more robust to disturbances, it will produce wider lateral motions in slower walking gaits. In practice, the default foothold location that worked reliably during our experiments on ANYmal was computed as the vertical projection of the hips to the terrain.

To track on average high-level velocities which drive the whole locomotion framework, we penalize deviations from foothold locations. These are computed assuming that a constant velocity is achieved over the duration of the optimization horizon. To avoid jumps in the reference footholds, we additionally set a cost to the distance between the current solution and the previously computed one.

Finally, we add to the cost a stabilizing term which is a function of the inverted pendulum model as described in [22], which computes the k-th foothold as $w(\boldsymbol{v}_{ref} - \boldsymbol{v}_{hip_k})\sqrt{h/g}$, with w a positive scalar weight, $\boldsymbol{v}_{ref}$ the high level reference velocity, $\boldsymbol{v}_{hip_k}$ the linear velocity of the k-th hip, h the height of the k-th hip from the ground and g the gravity acceleration constant.

6.4.3.2 *Inequality constraints*

To avoid computing footholds which would violate the kinematic extension of the legs, we exploit the QP setup by adding inequality constraints on the feasible location of each foothold. We do this by considering the maximum leg extension l_{max} and the measured height of each hip h_i. By projecting the hip location on the terrain $\boldsymbol{h}_0$, we set inequalities describing a polygon which has n_p vertices equally distributed around $\boldsymbol{h}_0$. The distance between each vertex and the hip projection to the terrain is computed as $\sqrt{l_{max}^2 - h_i^2}$.

6.4.4 *Support polygon sequence*

The method used to generate a sequence of support polygons is similar to the one presented in [73]. Each gait is described by a contact schedule, which in turn defines lift-off and touch-down events for each leg over a complete gait stride. Between each pair of events, a support polygon is defined by the convex hull of the location of the expected contact points, as well as its time duration in seconds. The edges of each support polygon are used to generate the inequality constraints for the ZMP location, such that the optimized motion is dynamically stable.

6.4.5 *Gait Switching*

To fully exploit the capability of executing different gaits, we have developed a gait switching algorithm which enables the locomotion framework to safely and quickly switch from a currently *active gait* to a *desired gait*. The software allows the latter to be selected by the operator.

As described in section 6.4, the motion optimization horizon is computed as the time duration of the active gait. Hence, by changing the contact schedule at the end of the current stride, we avoid jumps in the solution of the optimization. To properly append a new contact schedule to the current one, we first try to connect the active gait with the desired one by selecting lift-off events from the former and combine them with touch-down events from the latter. This can fail for several reasons: there are no overlapping swing phases across active and desired gait at the transition; a new phase event is detected (a foot touch-down or lift-off) that neither corresponds to the active nor the desired gait; or the swing phases would become numerically too large or small at the transition. In case of a failure, the transition is obtained by adding a full stance phase between the gait transitions. Additionally, each gait has its own stride duration which is updated in the vicinity of a transition.

The transition is seen by the optimizer as a change of the incoming sequence of support polygons. This allows the adaption of the body trajectory to the desired gait before the switch occurs.

6.5 CONTROL

To track the reference motion plan we use a Whole-Body Controller (WBC) which relies on the Hierarchical Optimization [73] (HO) framework to

TABLE 6.1: The list of prioritized tasks used to control the robot and track the motion plans. Priority 1 is the highest.

Priority	Task
1	Floating base equations of motion
2	Torque limits
	Friction cone and $\boldsymbol{\lambda}$ modulation
3	No motion at the contact points
4	Center of mass linear motion
	Center of mass angular motion
	Swing leg motion tracking
5	Contact force minimization

optimize for the generalized accelerations $\dot{\boldsymbol{u}}_d \in \mathbb{R}^{n_u}$ and contact forces $\boldsymbol{\lambda}_d \in \mathbb{R}^{3n_c}$, where n_u is the dimension of the generalized velocity vector $\boldsymbol{u}$, and n_c is the number of feet that are in contact with the environment. Table 6.1 summarizes the list of tasks, together with their priorities in the control hierarchy, which are used in our controller. Each of these tasks, which specifies equality or inequality constraints on the optimal solution, is solved as a QP problem. From the optimal solution $\dot{\boldsymbol{u}}^*$ and $\boldsymbol{\lambda}^*$, we compute the control torques $\boldsymbol{\tau}$ as

$$\boldsymbol{\tau} = \boldsymbol{M}_j \dot{\boldsymbol{u}}^* + \boldsymbol{h}_j - \boldsymbol{J}_{sj}^T \boldsymbol{\lambda}^*, \tag{6.21}$$

where $\boldsymbol{M}_j$, $\boldsymbol{h}_j$ and $\boldsymbol{J}_{sj}$ are, respectively, the rows of the mass matrix, velocity dependent terms and support Jacobian (described in section 6.3) relative to the actuated joints.

6.6 RESULTS

The framework described in the previous sections was tested in simulation and on the real hardware. A video demonstrating the capabilities of our approach, as well as the experiments discussed in this section, can be found online[1].

1 https://youtu.be/wQpLzoEzsx8

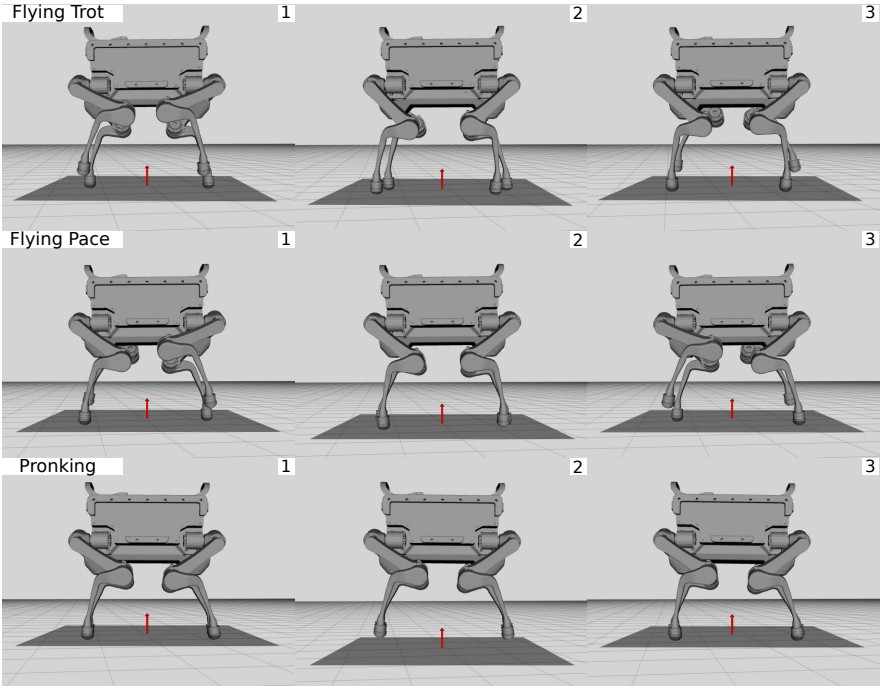


FIGURE 6.5: The motion planner is capable of generating motions for gaits which include full flight phases, such as flying trot, flying pace and pronking.

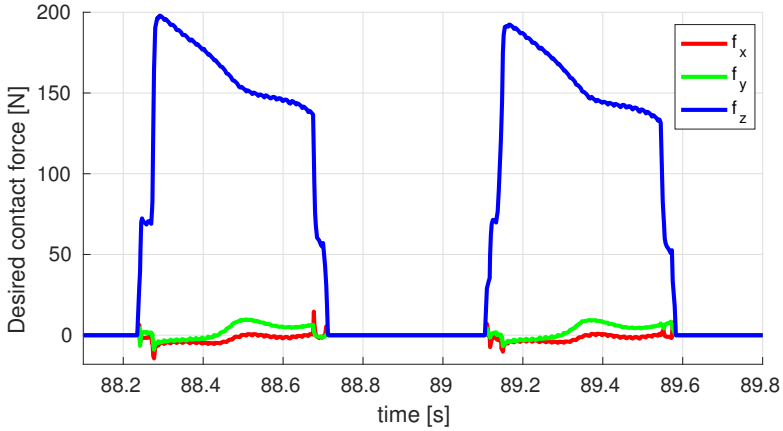


FIGURE 6.6: The desired contact forces for the left hind leg of ANYmal over two strides of a trotting gait computed by the motion controller.

6.6.1 *Simulation*

We have tested our approach by executing the motion plans in the robot simulation toolbox *Gazebo*. Fig.6.5 depicts snapshots of the execution of a flying trot, a flying pace and a pronking gait, while Fig.6.6 show the contact forces computed by the whole-body controller while trotting forwards. The optimized motions are tracked by a hierarchical whole-body controller, as discussed in section 6.5. The user can steer the robot in any desired direction by using an operator device (e.g. a joystick). A graphical user interaction module has been designed to allow to trigger a switch between the available gaits.

6.6.2 *Experiments*

Our experiments were conducted on ANYmal [56], an accurately torque-controllable quadrupedal robot. Control signals are generated in a 400Hz control loop which runs on the robot's on-board computer (Intel i7-7600U, 2.7 - 3.5GHz, dual core 64-bit) together with state estimation [57]. For modeling and computation of kinematics and dynamics, we use the open-source Rigid Body Dynamics Library [58] (RBDL), which is a C++ implementation of the algorithms described in [59]. To solve the nonlinear optimization problem described in section 6.4, we use a custom library which

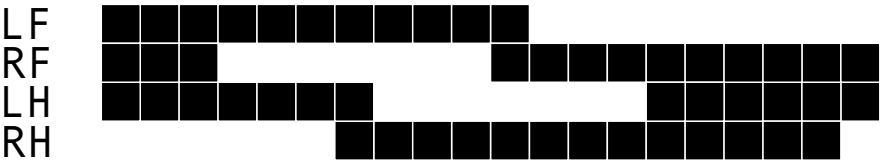


FIGURE 6.7: The gait pattern of a dynamic lateral walk which exhibits swing phase overlap over each pair of successive swing leg. The dark regions represent stance phases.

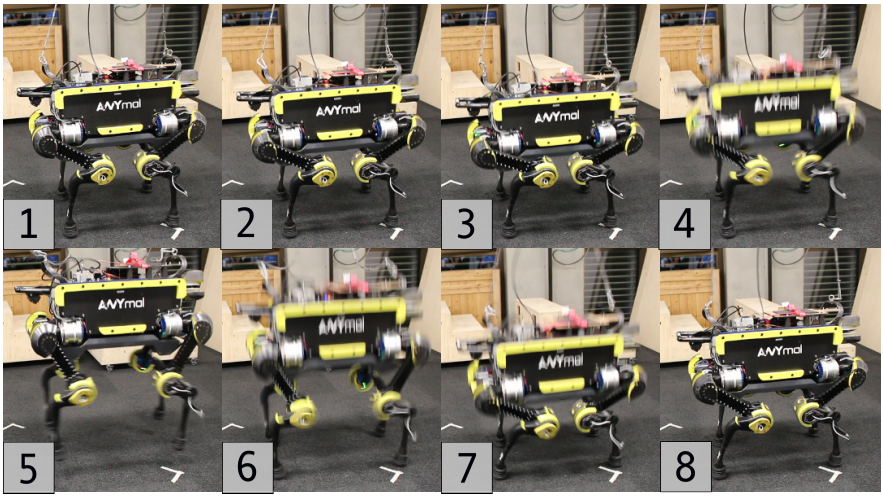


FIGURE 6.8: A sequence demonstrating a squat jump executed on ANYmal.

implements the SQP framework, which solves the nonlinear program by iterating through a sequence of Quadratic Programming (QP) problems. Each QP is numerically solved using the QuadProg++ [60] library, an off-the-shelf open source QP solver which implements the Goldfarb-Idnani active-set method [61]. To test the capability of producing full flight phases, we have tested a squat jump (see Fig.6.8) and a pronking gait (see Fig.6.9) on ANYmal. To execute the squat jump, the optimizer plans a motion that initially lowers the torso to build up speed and finally jump. The gait switching module has been tested by commanding an online switch between a trot and a dynamic lateral walk which exhibits full swing phase (see Fig.6.7) overlap between each pair of successive swing legs.

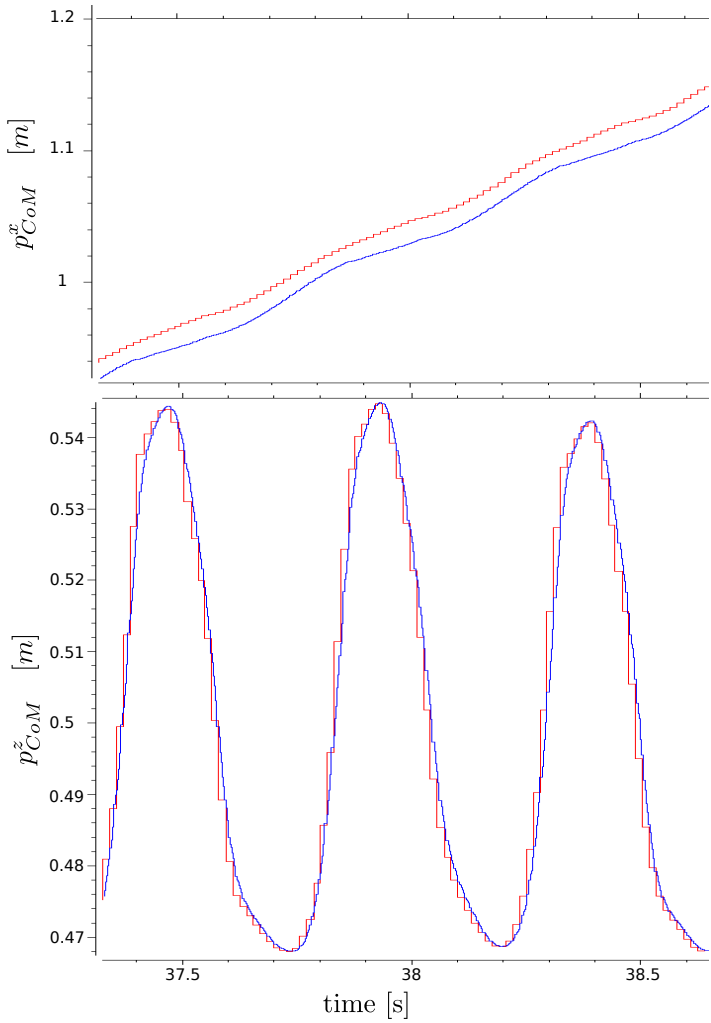


FIGURE 6.9: The x and z components of the center of mass motion generated to execute a pronking gait. The reference motion is in red, while the measured one is depicted in blue.

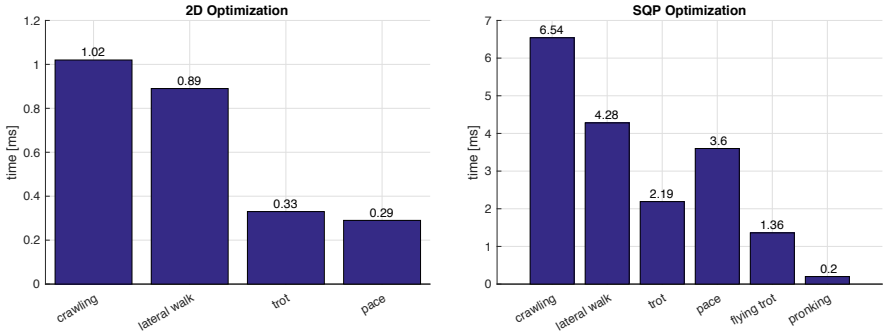


FIGURE 6.10: Comparison of the average computation time needed to optimize for different gaits in our current and previous work. The 2D model used in [73] leads to linear ZMP constraints and the resulting optimization can be handled with a simpler QP solver. The full 3D model accounts for the z component of CoM and, due to nonlinear stability constraints, is solved with a more computationally demanding SQP. This, however, allows to vastly increase the variety of gaits that can be executed.

6.7 CONCLUSIONS AND FUTURE WORK

We have shown that by extending our previous results by optimizing for the full center of mass position, as well as parallelizing the optimization of footholds, we are now capable of executing a broader range of gaits, including trotting, dynamic lateral walk, jumping in place and pronking. To execute transitions between them, we rely on a dedicated gait switching module. The motion optimization relies on a Sequential Quadratic Programming framework to solve the nonlinear ZMP constraints. Although more complex and more computationally demanding than our previous work, we are still able to produce motion plans online and execute them in an MPC-like fashion. This means that we are able to continuously plan online using the most recent state of the robot while we track the currently available motion plan. Fig.6.10 summarizes the computation time needed to optimize the torso motion for different gaits. A hierarchical whole-body controller tracks the motion plans and provides disturbance rejection. The natural continuation of this work will be to augment the framework with perception of the environment. This will allow to add additional constraints on the foothold selection. Additionally, this would allow to compute surface normals which would improve the robustness of the optimized motion plans over rough terrain.

7

PAPER 4: ALMA - ARTICULATED LOCOMOTION AND MANIPULATION FOR A TORQUE-CONTROLLABLE ROBOT

7.1 ABSTRACT

The task of robotic mobile manipulation poses several scientific challenges that need to be addressed to execute complex manipulation tasks in unstructured environments, in which collaboration with humans might be required. Therefore, we present ALMA, a motion planning and control framework for a torque-controlled quadrupedal robot equipped with a six degrees of freedom robotic arm capable of performing dynamic locomotion while executing manipulation tasks. The online motion planning framework, together with a whole-body controller based on a hierarchical optimization algorithm, enables the system to walk, trot and pace while executing operational space end-effector control, reactive human-robot collaboration and torso posture optimization to increase the arm's workspace. The torque control of the whole system enables the implementation of compliant behavior, allowing a user to safely interact with the robot. We verify our framework on the real robot by performing tasks such as opening a door and carrying a payload together with a human.

7.2 INTRODUCTION

Legged robots have significant advantages over their wheeled or tracked counterparts. They are capable of traversing challenging terrain and environments designed for human use (e.g., steps and stairs). Walking robots need to break contact and modulate the ground reaction forces to propel themselves through the environment. This modulation not only allows the robot to retain balance but also enables compliant behavior. In particular, quadrupedal robots typically exhibit a larger support area than bipedal systems, easing the design of robust motions.

Typical missions for a quadrupedal robot include exploration, mapping, navigating through challenging terrain, and inspecting scenarios which are undesirable for humans to be in [76]. Direct interaction with the environ-

FIGURE 7.1: The quadrupedal robot ANYmal equipped with a six DOF robotic arm. The system is fully torque-controlled, enabling compliant behavior and safe interaction.

ment, however, has been limited to the contacts used for locomotion, with little to no flexibility in the manipulation capabilities. Few robots use their legs for manipulation, and the possible tasks using the available feet [23] or a gripper tool attached to the feet [77] remain limited and renders simultaneous locomotion and manipulation hard or impossible. Equipping a multi-legged robot with an additional limb that is dedicated to manipulation tasks, greatly extends the possible real-world deployment. Such a robot will be able to carry and move objects, help a human to deliver a payload, open doors and interact with its surroundings in ways that were precluded before.

Similar solutions have been explored over the past few years [78]. The quadrupedal robot *HyQ* [79] is equipped with a six DOF arm and demonstrates a static walking gait while tracking motions of the arm. The authors propose a controller that takes into account internal and external disturbances created by the dynamics of the arm by optimizing for the ground

reaction forces. Impressive results have been achieved by Boston Dynamics' quadrupedal robots *Spot* [80] and *SpotMini* [81]. SpotMini, equipped with a five DOF arm, shows manipulation tasks while walking, e.g., opening a door and carrying a payload. So far, none of the details on the methods and approaches used to control these robots have been made available. In an older work of Boston Dynamics, the quadrupedal robot *BigDog* [82] demonstrates a throwing maneuver with a robotic arm while trotting in place.

Controlling such a system comes with several challenges. It requires robust and fast motion planning and control to enable simultaneous locomotion and manipulation in challenging environments while being able to cope with external disturbances. Such dynamic interaction with the environment through legs and arms of a walking robot requires taking into account the full system dynamics as well as the contact forces at the robot's end-effectors. Optimal contact force distribution for torque-controllable quadrupedal robots was demonstrated in experiments while taking into account equality [83] and inequality [84] constraints. Optimization algorithms to solve the contact force distribution based on the full rigid-body dynamics are shown in [85] and [73]. In these approaches, inequality constraints on the direction and magnitude of the linear contact forces are prescribed, leaving the exact contact force distribution to follow from the other whole-body controller tasks. However, when actively interacting with the environment using an arm (e.g., opening a door), it may be desirable to explicitly prescribe linear contact forces as well as contact torques between the gripper and environment. This increases the complexity of the contact force distribution problem for the entire system.

In this paper, we present ALMA (Articulated Locomotion and Manipulation for ANYmal, see Fig. 7.1), a planning and control framework for a fully torque-controlled quadrupedal manipulator capable of performing dynamic gaits while executing manipulation tasks. The framework allows the robot to compliantly react to external forces and to maintain balance while executing locomotion and manipulation tasks. To the best of our knowledge, this is the first time that such a system is shown performing coordination between dynamic locomotion and manipulation.

7.3 MODEL FORMULATION

We formulate the model of a walking robot equipped with a robotic arm as a free-floating base B to which limbs are attached. The motion of the

entire system can be described with respect to (w.r.t.) a fixed inertial frame I. The position of the Base w.r.t. the inertial frame, expressed in the inertial frame, is written as ${}_I\boldsymbol{r}_{IB} \in \mathbb{R}^3$. The orientation of the Base w.r.t. the inertial frame is parametrized using a Hamiltonian unit quaternion $\boldsymbol{q}_{IB}$. The limb joint angles are stacked in the vector $\boldsymbol{q}_j \in \mathbb{R}^{n_j}$, where $n_j = 18$. We write the generalized coordinate vector $\boldsymbol{q}$ and the generalized velocity vector $\boldsymbol{u}$ as

$$\boldsymbol{q} = \begin{bmatrix} {}_I\boldsymbol{r}_{IB} \\ \boldsymbol{q}_{IB} \\ \boldsymbol{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j}, \quad \boldsymbol{u} = \begin{bmatrix} {}_I\boldsymbol{v}_B \\ {}_B\boldsymbol{\omega}_{IB} \\ \dot{\boldsymbol{q}}_j \end{bmatrix} \in \mathbb{R}^{n_u}, \tag{7.1}$$

where $n_u = 6 + n_j$, ${}_I\boldsymbol{v}_B \in \mathbb{R}^3$ and ${}_B\boldsymbol{\omega}_{IB} \in \mathbb{R}^3$ are the linear and angular velocity of the Base w.r.t. the inertial frame expressed respectively in the I and B frame. The robot (see Fig. 7.1) has $n_u = 24$, with six, twelve, and six DOF describing the floating base, legs, and the arm, respectively. The equations of motion of a floating base system that interacts with the environment are written as $\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}_s^T(\boldsymbol{q})\boldsymbol{\lambda}$, where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n_u \times n_u}$ is the mass matrix and $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) \in \mathbb{R}^{n_u}$ is the vector of Coriolis, centrifugal and gravity terms. The selection matrix $\boldsymbol{S} = \begin{bmatrix} \boldsymbol{0}_{n_\tau \times (n_u - n_\tau)} & \mathbb{I}_{n_\tau \times n_\tau} \end{bmatrix}$ selects which DOF are actuated. If all limb joints are actuated, then $n_\tau = n_j$. The vector of contact forces and contact torques $\boldsymbol{\lambda}$ is mapped to the joint-space torques through the support Jacobian $\boldsymbol{J}_s \in \mathbb{R}^{n_s \times n_u}$, which is obtained by stacking the Jacobians which relate generalized velocities to limb end-effector motion as $\boldsymbol{J}_s = \begin{bmatrix} \boldsymbol{J}_{C_1}^T & \cdots & \boldsymbol{J}_{C_{n_c}}^T \end{bmatrix}^T$, with n_c the number of limbs in contact and n_s the total dimensionality of all contact wrenches. For the point-feet only three dimensional linear contact forces are modeled. In contrast, the gripper exerts a six dimensional contact wrench, when rigidly gripping onto its environment.

7.4 MOTION GENERATION

Thanks to the robot's high number of DOF, it is possible to simultaneously and independently control the motion of the floating base and the gripper. The software framework schematically displayed in Fig. 7.2 allows to send high-level operational space velocity commands in order to drive locomotion in a specified direction, and to move the gripper to the desired pose. For locomotion, these velocity commands, together with the actual robot

state, are transformed to reference footholds[1] and motion reference trajectories for the robot's whole-body COM. This motion generation framework is based on our previous work [86], that describes a reactive receding-horizon ZMP-based motion planner that enables the execution of dynamic gaits such as a trot, pace and running trot. Continuous online replanning of the motion references results in a reactive behavior of the robot. Hence, the system can cope with unexpected disturbances, such as unmodeled irregularities in the terrain or a push by a human, by updating the motion plans to remain balanced.

7.4.1 *Gripper Motion References*

For the gripper we continuously update a desired pose $\boldsymbol{p}_{IG}^{des}$ and reference twist ${}_I\mathbf{w}_{IG}^{des}$ to be tracked by the motion controller. These desired values are computed by a gripper motion planner, or by a twist input $\mathbf{w}$ from a user-operated joystick. In the latter case, the desired pose is updated as $\boldsymbol{p}_{IG_{k+1}}^{des} = \boldsymbol{p}_{IG_k}^{des} +^* \Delta t \mathbf{w}$, where Δt is the duration of the control loop, k refers to the current time step, and $+^*$ is defined as the vector space addition operator for the translational part of $\boldsymbol{p}_{IG}$ and as $\boxplus$ [53] for the rotational part. The reference twist is updated as ${}_I\mathbf{w}_{IG}^{des} = \mathbf{w}$.

The gripper's motion references can be expressed w.r.t. to any coordinate frame, e.g., the inertial frame I, or the base frame B. Expressing the reference motion w.r.t. the inertial frame allows to drive the gripper to a desired position in the world, while the robot is still free to walk, change its posture, and retain balance if an external disturbance is acting on the system. In other situations, such as when the robot has to walk to a different location while carrying a payload, it can be more convenient to express the reference motion of the gripper w.r.t. the base frame B.

7.4.2 *Foothold Planning*

Our previous foothold planning framework was based on including the inverted pendulum model [22] in an online optimization problem [86]. The latter additionally included constraints to avoid the kinematic limits of the legs. In this work, we modify this QP problem to plan the footholds w.r.t. the position of the whole-body center of mass instead of the center of the

1 A *foothold* is defined as the desired contact location for a leg in swing phase.

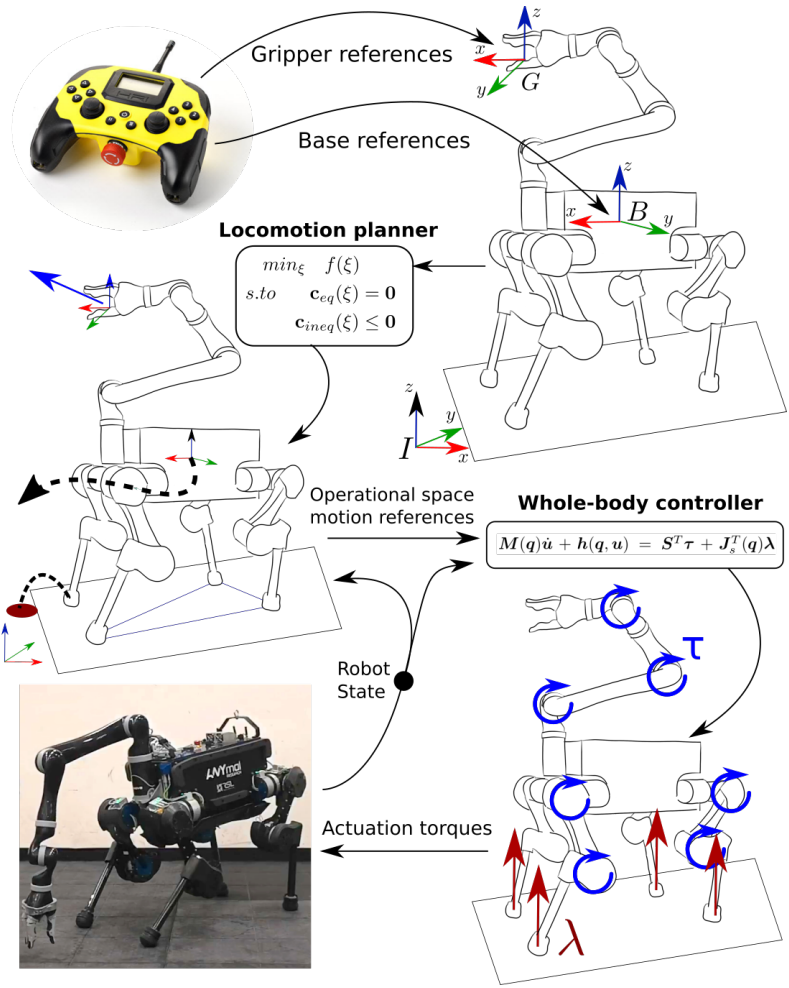


FIGURE 7.2: The planning and control framework described in this paper. High-level velocity references are sent to the floating base or to the gripper. References for the latter are interpreted as velocity updates (w.r.t. the inertial or the floating base frame) for the gripper's desired pose. Velocity references for the base are sent to the online locomotion planner which computes COM trajectories. The operational space references are tracked by a whole-body controller algorithm based on hierarchical optimization that generates torque references for all actuated joints.

TABLE 7.1: The tasks used in the experiments. Each task is associated with a priority (1 is the highest). Of the tasks marked with an asterisk, only one is active at a time.

Priority	Task
1	Equations of Motion
	Torque limits
	Friction cone limits
	No contact motion
2	Center of Mass horizontal motion tracking
	Torso height motion tracking
	Torso angular motion tracking
	Swing foot linear motion tracking
	Torso orientation adaptation
	Gripper spatial motion tracking*
	Gripper contact wrench*
3	Contact wrench minimization

torso. This modification is crucial since changing the arm configuration can impose a significant shift in the overall COM position.

7.4.3 *Whole-Body Center of Mass Motion Planning*

The desired whole-body COM motion reference trajectory is obtained by solving an online nonlinear optimization [86] which guarantees stable locomotion by constraining the robot's ZMP to always lie inside the convex hull of the contact points, i.e., the current and upcoming support polygons. The optimization takes into account different kinds of support polygons (i.e., points, lines, triangles and quadrilaterals). This flexibility makes it possible to generate motion plans for any gait that exhibits these support polygons, from a static walk to a running trot with full flight phases and a pronking gait. This motion planner reduces the model of the robot to a single point mass, being the robot's COM. It does therefore not require any adaptations to apply this planner to a quadrupedal robot equipped with one or more additional limbs.

7.5 CONTROL

Extending on our previous research [65, 73], we track operational space motion and force references with a whole-body control algorithm that generates torque references for all the controllable joints by using hierarchical optimization. The controller computes optimal generalized accelerations $\dot{\boldsymbol{u}}^*$ and contact forces $\boldsymbol{\lambda}^*$ by solving a cascade of prioritized tasks which specify equality and inequality constraints as $\boldsymbol{A\xi} = \boldsymbol{b}$ and $\boldsymbol{C\xi} \leq \boldsymbol{d}$, where $\boldsymbol{\xi} = \begin{bmatrix} \dot{\boldsymbol{u}}^T & \boldsymbol{\lambda}^T \end{bmatrix}^T$. The desired torques $\boldsymbol{\tau}_d$ are obtained from the optimal solution $\dot{\boldsymbol{u}}^*$ and $\boldsymbol{\lambda}^*$ as $\boldsymbol{\tau}_d = \boldsymbol{M}(\boldsymbol{q})_j \boldsymbol{u}^* + \boldsymbol{h}_j(\boldsymbol{q}, \boldsymbol{u}) - \boldsymbol{J}_j(\boldsymbol{q})_s^T \boldsymbol{\lambda}^*$, where $\boldsymbol{M}_j$, $\boldsymbol{h}_j$ and $\boldsymbol{J}_j$ are the rows of the mass matrix, nonlinear terms and support Jacobian associated with the dynamics of the actuated degrees of freedom. Table 7.1 shows the list of prioritized tasks used throughout our experiments. The tasks with the highest priority guarantee dynamic feasibility, compliance with the robot's physical limitations, and adherence to the contact constraints [73]. The tasks at the second priority level specify the robot's desired motion. The lowest priority task removes any internal force redundancy by minimizing the norm of all contact wrenches. The implementation of the tasks dedicated to locomotion is described in [73]. Sections 7.5.1 through 7.5.3 describe the design of tasks tailored for manipulation and how to integrate them into our control framework to allow coordination of locomotion and manipulation.

The manipulation tasks are focused on controlling the gripper's motion and interaction forces with the environment, and increasing the arm's kinematic reachability by adjusting the orientation of the torso. Furthermore, we discuss the adaptation of the motion tracking tasks in order to deal with control instabilities that occur at kinematically singular configurations.

7.5.1 *Gripper Motion and Contact Wrenches*

In order to track translational and rotational motion references, the spatial motion tracking task for the gripper is written as

$$\begin{bmatrix} \boldsymbol{J}_{G\ 6\times n_u} & \boldsymbol{0}_{6\times n_s} \end{bmatrix} \boldsymbol{\xi} = \ddot{\boldsymbol{x}}_{ref} - \dot{\boldsymbol{J}}_G \boldsymbol{u}, \tag{7.2}$$

where $\boldsymbol{J}_G$ is the matrix that maps the robot's generalized velocities to the translational and rotational velocities of the gripper. $\dot{\boldsymbol{J}}_G$ is the Jacobian's time derivative, and $\ddot{\boldsymbol{x}}_{ref}$ is the operational space accelerations reference for the gripper.

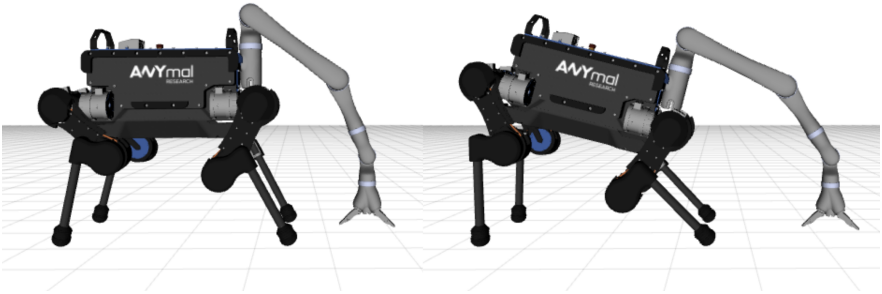


FIGURE 7.3: Reaching on the ground with no torso orientation adaptation (left) results in a limited kinematic reach compared to taking into account the configuration of the arm (right). By including the torso orientation adaption task, the reach in the depicted situation is increased by 15 cm.

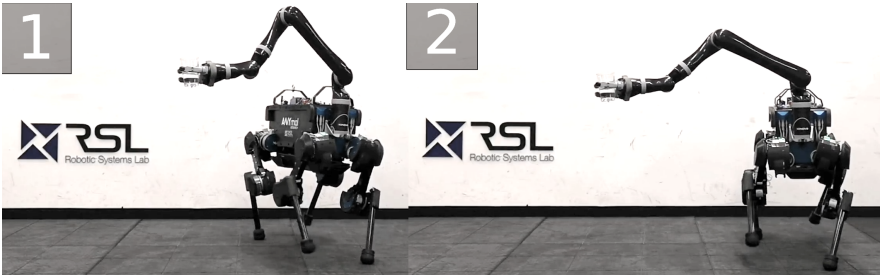


FIGURE 7.4: By combining a ZMP-based locomotion planner with hierarchical whole-body control, the robot is able to keep a glass of water at a fixed pose while the torso is commanded to walk in the desired direction.

When the gripper is rigidly in contact with the environment, we can remove the motion tracking task and instead command an interaction wrench. Since the contact forces and torques at the gripper appear explicitly in the optimization vector ξ, the required task can be written as $\begin{bmatrix}\mathbf{0}_{6\times n_u} & S_{G\,6\times n_s}\end{bmatrix}\xi = w_{ref}$, where S_G is the selection matrix that selects the contact forces and contact torques belonging to the gripper, and $w_{ref} \in \mathbb{R}^6$ is the contact wrench reference. When commanding a gripper contact wrench, the support Jacobian J_s needs to be updated to include the arm. As a result, the arm is included in the "No contact motion" task.

7.5.2 *Torso Orientation Adaptation*

A typical task to execute for a mobile manipulator is to reach for and grasp an object. The latter might be, however, out of the kinematic reach (e.g., when on the ground or on a high shelf). This limitation is addressed by exploiting the kinematic redundancy introduced by the floating base through adaptation of the torso orientation without interfering with the reference positions of the COM (to avoid interfering with the stability criterion in Section 7.4.3) and the gripper (see Fig. 7.3). By adapting the torso's orientation appropriately, the kinematic reach of the gripper is significantly improved.

A possible approach for achieving this whole-body reaching behavior is by exploiting the hierarchical setup of the whole-body controller tasks. This approach requires to have different priority levels for the torso angular motion tracking task and the COM linear motion tracking task. Such a hierarchy setup may significantly degrade the execution of walking gaits such as a trot, during which the robot is underactuated when only two point-feet are in contact with the environment.

For this reason, we propose the addition of a control task which explicitly prescribes the desired angular motion of the torso to increase the gripper's reachable workspace, instead of depending on the hierarchical nature of the motion controller. First, we define a desired linear velocity ${}_B\boldsymbol{v}_{IS}^{des}$ for the shoulder (i.e., the mounting point of the arm), expressed in the Base frame. We define ${}_B\boldsymbol{v}_{IS}^{des}$ as ${}_B\boldsymbol{v}_{IS}^{des} = k_p \ {}_B\hat{\boldsymbol{r}}_{SG}$, with k_p a scalar which multiplies the unit vector ${}_B\hat{\boldsymbol{r}}_{SG}$ that points along the direction from shoulder to the gripper. Rotational velocities of the base along its vertical z and frontal x axes produce an instantaneous velocity at the shoulder in the same direction. For this reason, the desired shoulder velocity ${}_B\boldsymbol{v}_{IS}^{des}$ cannot be mapped directly to desired angular base velocities. Instead, we project it to a desired angular velocity for the floating base only around its x and y axes as ${}_B\boldsymbol{\omega}_{IB_{xy}}^{des} = \boldsymbol{S}({}_B\boldsymbol{r}_{BS})_B^{\dagger}\boldsymbol{v}_{IS}^{des}$, where $\boldsymbol{S}({}_B\boldsymbol{r}_{BS})^{\dagger}$ denotes the pseudo-inverse of the skew-symmetric matrix, computed such that $\boldsymbol{S}({}_B\boldsymbol{r}_{BS})_B\boldsymbol{v}_{IS}^{des} = {}_B\boldsymbol{r}_{BS} \times_B \boldsymbol{v}_{IS}^{des}$, with ${}_B\boldsymbol{r}_{BS}$ the position of the shoulder w.r.t. the base. Subsequently, ${}_B\boldsymbol{\omega}_{IB_z}^{des}$ is set proportional to ${}_B\boldsymbol{\omega}_{IB_x}^{des}$ because their relation to the direction of the velocity of the shoulder is identical. The resulting task is integrated into the task hierarchy as

$$\begin{bmatrix} \boldsymbol{J}_{B_r\ 6\times n_u} & \boldsymbol{0}_{3\times n_s} \end{bmatrix} \boldsymbol{\xi} = k_d({}_B\boldsymbol{\omega}_{IB_{xyz}}^{des} - {}_B\boldsymbol{\omega}_{IB_{xyz}}) - \dot{\boldsymbol{J}}_{B_r}\boldsymbol{u}, \tag{7.3}$$

where $\boldsymbol{J}_{B_r}$ is the Jacobian of rotational motions of the base, k_d is a scalar derivative gain, ${}_B\boldsymbol{\omega}^{des}_{IB_{xyz}}$ and ${}_B\boldsymbol{\omega}_{IB_{xyz}}$ are the reference and measured angular velocity respectively, and $\dot{\boldsymbol{J}}_{B_r}$ is the time derivative of $\boldsymbol{J}_{B_r}$.

7.5.3 *Kinematic Singularity Robustness*

While executing manipulation tasks, the reference motion for the end-effector can drive a limb to a kinematically singular joint-space configuration. This situation is likely to occur in the form of a full knee or elbow extension, for example when trying to pick up an object that is too far away.

When a limb is in a kinematically singular configuration, it loses one controllable operational space DOF of the end-effector. This situation is characterized by one of the Jacobian's singular values approaching zero, and the controller computing infeasibly high joint torque references, leading to control instabilities. For traditional analytic inverse kinematics and inverse dynamics control approaches, this problem is typically addressed through the use of the damped pseudo-inverse [87]. This approach, however, is not relevant for the presented control framework which relies on numerical optimization to solve for the task hierarchy. Therefore, we address the issue of kinematic singularities by setting a lower bound on the Jacobian's singular values, as mentioned in [88]. The singular value decomposition of the limb's Jacobian that is used, e.g., inside the motion control task in (7.2) is performed by setting a minimum non-zero value to the computed singular values and by using the latter to recompute the Jacobian. This adaptation results in the ability to drive the robot into kinematically singular configurations without any arising motion instabilities. We apply this singular value adjustment for all of the robot's motion tasks: "No contact motion", "Swing foot linear motion tracking", and "Gripper spatial motion tracking".

7.6 EXPERIMENTS

Our experiments were conducted on ANYmal [56], an accurately torque-controllable quadrupedal robot, equipped with the Jaco^2 [89] six DOF robotic arm from Kinova. The arm is light-weight (4.4 kg) and allows for torque-control of all six actuators. The control references are generated in a 400 Hz control loop that runs on the robot's onboard computer (Intel i7-7600U, 2.7 - 3.5GHz, dual-core 64-bit) together with state estimation [90].

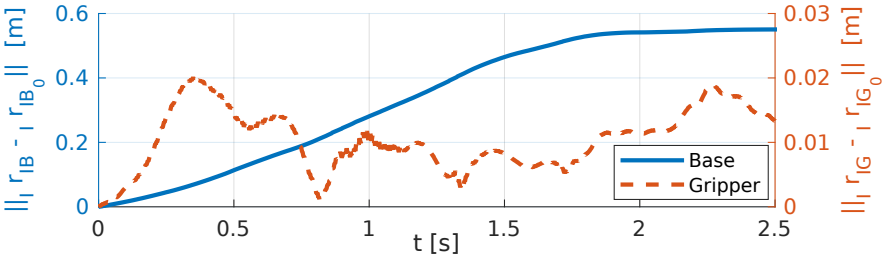


FIGURE 7.5: The motion of the torso and gripper positions ${}_I\boldsymbol{r}_{IB}$ and ${}_I\boldsymbol{r}_{IG}$ w.r.t. the initial positions ${}_I\boldsymbol{r}_{IB_0}$ and ${}_I\boldsymbol{r}_{IG_0}$ when the gripper is controlled to remain at a fixed position during locomotion. While the torso travels 55 cm, the gripper's deviation from its initial position is at most 2 cm.

We use the open-source Rigid Body Dynamics Library [58] (RBDL), a C++ implementation of the algorithms described in [59], to generate the model of the kinematics and dynamics of the system. The locomotion planner uses a custom SQP framework that iteratively solves a sequence of QP problems by using a custom version of the open-source QuadProg++ [60] library, a C++ implementation of the Goldfarb-Idnani active-set method [61]. The same algorithm is used to numerically solve the cascade of prioritized tasks in the whole-body controller. The following experiments are supported by the video submission[2].

7.6.1 *Locomotion and End-effector Motion Control*

We show the strength of the combination of our ZMP-based locomotion planning framework and whole-body control by commanding the robot to walk in various directions while commanding the gripper to stay at a fixed pose in the inertial frame (see Fig. 7.4). As depicted in Fig. 7.5, the robot can accurately track the gripper's desired pose while trotting away from it.

7.6.2 *Reactive Behavior and Posture Adaptation*

The torque-controlled system and the applied motion and control framework allow to robustly deal with external disturbances. At the actuation level, the robot exhibits compliant behavior because of its torque-controlled joints and the inverse dynamics-based whole-body controller. Instead of

2 Available at https://youtu.be/XrcLXX4AEWE

FIGURE 7.6: The robot carrying a 3.3 kg payload together with a human collaborator. Locomotion is triggered when the force on the gripper in the $x-y$ plane perpendicular to gravity is greater than a user-defined threshold. The online motion planner described in Section 7.4.3 computes the required motion reference trajectories for locomotion in the direction of the detected force.

the robot being rigidly stiff, external forces on the gripper or the torso will result in a spring-damper type of motion. On the planning level, the system displays compliance and reactive behavior w.r.t. its environment through the locomotion planning framework that continuously replans in a receding-horizon fashion. The robot's motion references are updated accordingly when external disturbances, such as rough terrain or external forces, act on the robot. The human-robot collaboration scenario, depicted in Fig. 7.6, demonstrates the system's compliance and reactive behavior. The gripper is commanded to pick up a 3.3 kg box together with a human collaborator. When the box is lifted, the desired locomotion velocity is computed based on the direction of any horizontal forces acting on the gripper. By pulling the box, the human initiates locomotion in the direction of the detected forces. Despite the human pulling on the gripper, and the robot carrying an object with unmodeled weight, a collaborative payload delivery task is accomplished.

Section 7.5.2 describes a whole-body controller task that extends the kinematic reach of the gripper by adapting the angular velocity references of the torso. Fig. 7.8 depicts how the torso adapts to different configurations of the arm, allowing the robot to reach the ground around itself easily.

7.6.3 *Opening a Door*

To illustrate the ability of our whole-body control framework to handle commanded contact forces and torques at the gripper, while accurately controlling the robot's COM motion, we demonstrate the execution of a door opening task while using a trotting gait. To push the door and open it, desired contact forces at the gripper are commanded as described in

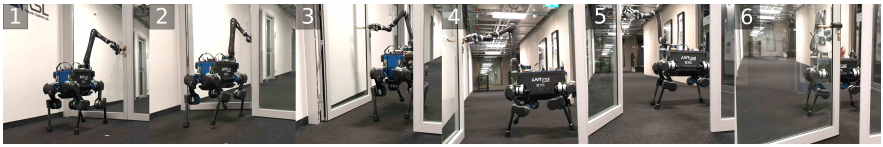


FIGURE 7.7: Commanding a contact force task instead of a motion task for the gripper allows the robot to open a spring-loaded door without requiring exact knowledge of the door kinematics. The integration of this task in the whole-body controller combined with the ZMP-based locomotion controller enables the robot to execute a trotting gait while passing through the door.

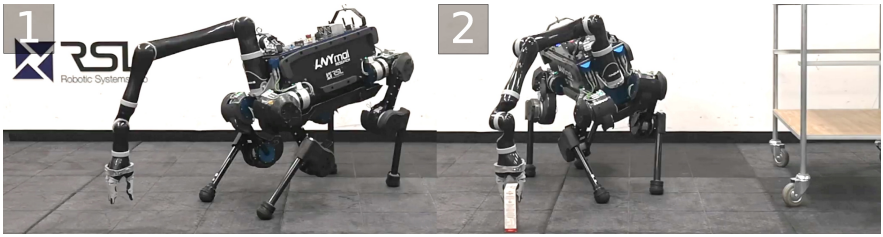


FIGURE 7.8: The torso orientation adaptation task in the whole-body controller task hierarchy results in an emerging whole-body reaching behavior when the gripper is commanded to move to various locations. Frame 1 shows a robot configuration where the torso's pitch contributes to the arm's reach, whereas in frame 2 it is the torso's roll and yaw that contribute significantly.

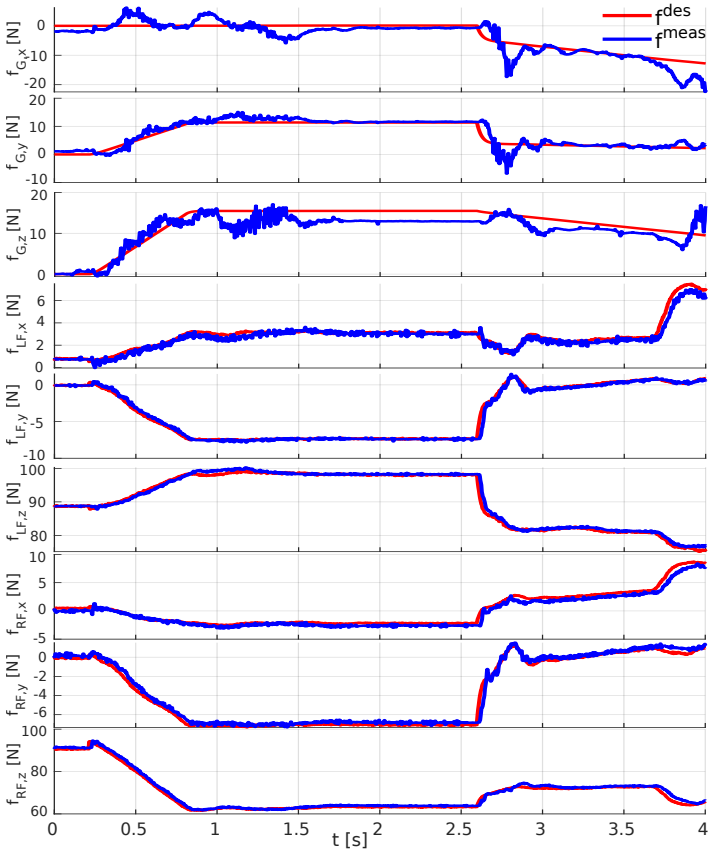


FIGURE 7.9: The time evolution of the commanded and estimated reaction forces acting on the gripper and the left and right front foot when the robot is standing and the gripper is interacting with the handle of a door. The first three plots refer to the gripper being commanded to push both vertically and laterally on the gripper to unlock the door. The commanded gripper force is planned offline. The three middle and lower plots depict the force distribution on the left and right fore leg respectively. The actual forces are estimated based on the measured joint torques. After 2.5 s, the gripper is commanded to push against the door to open it.

Section 7.5.1. The direction and magnitude of the desired contact force at the gripper is computed at each moment in time based on the currently estimated door opening angle and the error between a desired and actual gripper velocity. Simultaneously the robot is commanded to trot forward to pass through the door. Fig. 7.7 shows the execution of this task for a spring-loaded door. The advantage of commanding contact forces for the gripper instead of the desired motion is that the gripper can passively follow the kinematically constrained path prescribed by the door motion, even when contact forces are commanded that are not perfectly tangential to the motion path of the door handle. The whole-body controller task setup, and most specifically the compliance with the high-priority equations of motion task, guarantees a dynamically consistent contact force distribution over the robot's limbs during the execution of this task. The force distribution that is computed while interacting with the door handle is depicted in Fig. 7.9.

7.7 CONCLUSIONS AND FUTURE WORK

We present results on ANYmal, a quadrupedal robot, equipped with a six DOF robotic arm, forming a fully torque-controlled mobile manipulator. This system is able to perform dynamic locomotion while executing manipulation tasks, e.g., payload delivery, human-robot collaboration, and opening doors. An online ZMP-based motion planning framework is employed on the system to enable robust and reactive locomotion while executing manipulation tasks. Joint torque references are generated by a whole-body controller which takes into account the dynamics of the whole system, in contrast to other works in this field where the arm is seen as a disturbance that needs to be compensated for. The torque control approach allows a compliant and safe interaction with the environment.

Future work will focus on extending the motion planning framework to take into account the contact locations of the hand and the forces that it produces on the environment. Taking these quantities into account will produce motion plans that are consistent with the increased complexity of non-coplanar contact configurations. Specifically, trajectory optimization algorithms should be explored that plan for contact locations for the arm's end-effector. Adding these quantities to the planner's optimization variables allows the generation of complex maneuvers, such as the robot holding itself on a rail while walking on stairs.

8

PAPER 5: OPTIMIZATION-BASED MOTION GENERATION AND WHOLE-BODY CONTROL: DYNAMIC LOCOMOTION AND MANIPULATION FOR MULTI-LIMB ROBOTS

8.1 ABSTRACT

In this article, we present an online motion generation algorithm, coupled with a hierarchical whole-body controller, that enables a multi-limbed robot to perform dynamic gaits and manipulation tasks. Whole-body motion references are obtained by solving, in parallel, a set of optimization-based problems. The COM reference motion is generated by solving a nonlinear optimization problem which guarantees dynamic stability by constraining the ZMP to lie inside the support polygon. Using this reduced-complexity dynamics model, the planner efficiently generates motion plans for a wide variety of gaits, including walking, running, and galloping, while providing robustness against external disturbances and unknown terrain irregularities. Further, a whole-body controller solves a hierarchical sequence of tasks by taking into account the full rigid-body dynamics, as well as equality and inequality constraints, e.g., friction cone limits and torque limits. We evaluate our approach by executing a wide variety of tasks on ANYmal, a torque-controlled quadrupedal robot. By equipping ANYmal with a six-degree-of-freedom robotic arm, we show how our framework exploits the additional limb to execute manipulation tasks, e.g., opening doors and grasping objects. Finally, we demonstrate the performance of our planning and control framework while trotting over steep and challenging real-world terrains.

8.2 INTRODUCTION

Legged robots have unique mobility capabilities which make them more suitable than wheeled or tracked systems for deployment in challenging real-world scenarios. These include search and rescue, inspection, and exploration tasks, which typically require machines capable of traversing challenging terrain and negotiating obstacles. Some tasks may require ma-

FIGURE 8.1: ANYmal [56] is a torque-controlled quadrupedal robot actuated by series-elastic actuators. To execute manipulation tasks, ANYmal is equipped with a six DOF torque-controllable robotic arm.

nipulation skills as well, such as grasping objects or opening doors. The natural counterparts of these machines have impressive capabilities that robots still struggle to achieve. Animals such as goats and cheetahs are capable of running, jumping, and performing dynamic and agile maneuvers. Moreover, humans are not only capable of performing similar tasks, but also use their arms and hands to interact with their surroundings, e.g., manipulate objects.

A walking robot generates its motion by producing reaction forces between its legs and the terrain on which it is locomoting. Several challenges specific to legged locomotion arise when implementing the set of skills necessary to navigate in a real-world environment: switches in the contact state of the legs and regulation of the interaction forces affect the control complexity, which further increases when additional limbs are added to the machine for manipulation. To retain balance in real-world environments, the reference motions should be generated according to a stability

criterion. Fast replanning is thus necessary in case of unexpected events such as unseen terrain irregularities and external perturbations. An ideal control framework is able to generate stable locomotion for legged robots of various morphologies, and regardless of gait type.

Thanks to technological advancements in computational capabilities and mechanical design, legged robots have shown impressive results over the last years. *HyQ* [11], a quadrupedal robot from IIT, has demonstrated walking over challenging terrain such as on stepping stones [63] and between sloped surfaces [91], using a statically stable creeping gait [41]. In [44], a trajectory optimization framework is proposed to execute dynamic gaits such as trotting and pacing. In [71], agile running motions have been demonstrated on the humanoid robot Toro [72], albeit only in a simulated environment. Despite the fact that the targeted robot platform is morphologically different (i.e., humanoid versus quadruped), the problem formulation introduced in [71] (e.g., parametrization of the motion with fifth-order splines and the dynamic model used when in full flight phase) shares features with the one we describe in this work. Although the results are impressive, the approach is limited to the planning and execution of running gaits, on flat terrain or stepping stones. The Biomimetic Robotics Lab at MIT has developed the quadrupedal robot *Cheetah 3* [92], an enhanced design of its predecessor *Cheetah 2* [64]. In [93], MPC allows Cheetah 3 to execute a wide selection of gaits, including trotting, pacing, bounding, galloping, a three-legged walk, and blind locomotion over debris-covered stairs. While this results in remarkable locomotion capabilities, the applied control framework does depend strongly on the assumption that the dynamics of the robot's mass is negligible. This might introduce difficulties when this assumption has to be violated, for example when the legs constitute a significant part of the robot's total mass, when executing fast motions of the legs, or when adding a limb dedicated to manipulation tasks.

Equipping a legged robot with one or more limbs for manipulation of its environment can greatly improve the system's real-world applicability. Various implementations of this concept have been explored over the past few years [78]. Only few of these multi-legged mobile manipulators, however, possess torque-controllability, which is required for dynamic and compliant interaction with the environment. Of these few, a notable example is presented in [79], where the quadrupedal robot *HyQ* is equipped with a six-DOF arm and demonstrates a static walking gait while tracking prescribed motions of the arm. The authors propose a controller which takes into account internal and external disturbances created by the manipula-

tor while optimizing the ground reaction forces for tracking of the robot's motion references. Boston Dynamics has achieved impressive results in terms of coordination between locomotion and manipulation. In recent years, they have presented a wide range of robots (a humanoid robot *Atlas* [94] and quadrupedal robots *Spot* and *SpotMini* [95]) which are capable of performing highly dynamic maneuvers, locomoting in very challenging terrain, and coordinating locomotion and manipulation to execute tasks such as picking and placing objects or opening doors. Unfortunately, very little is known about the methods that are used on these systems due to the missing publications.

The complex interaction between a multi-limb robot and the environment requires careful execution of contact forces by taking into account the system's dynamics. Optimal contact force distribution for torque-controlled quadrupedal robots was demonstrated in experiments while taking equality [83] and inequality [84] constraints into account. Optimization algorithms to solve the contact force distribution based on the whole-body dynamics are shown in [85]. In these methods, inequality constraints on the direction and magnitude of the ground reaction forces are defined by friction cone constraints. The exact required contact forces (and corresponding joint torques) are then computed based on the desired motions and the full system dynamics. However, when actively interacting with the environment via an arm (e.g., opening a door), it can be desirable to explicitly track references for linear contact forces as well as contact torques between an end-effector and the environment.

The Robotic Systems Lab at ETH Zürich has developed *ANYmal* [56] (see Fig. 8.1), a fully torque-controlled quadrupedal robot. Over the last years, we have demonstrated an online optimization approach which has enabled ANYmal to execute a wide variety of gaits and smooth transitions between them [73, 86]. This approach is based on an online ZMP-based [9] motion planner that produces motion plans for the robot's COM in a receding-horizon fashion. The computed motion references are tracked using a whole-body controller based on hierarchical inverse dynamics, which takes into account the full rigid-body dynamics. In contrast to planning only for the DOF of the COM, in this article we show planning for angular motions as well. This contribution allows us to plan motions for a new set of gaits (e.g., bounding and galloping) which we test on the real hardware and for which we thoroughly report the experimental results.

Recently we equipped ANYmal with a six DOF torque-controlled arm. We presented an extension of our control framework for application on

this new system, achieving fine coordination of locomotion and manipulation in order to execute complex tasks, such as door opening and human-robot collaboration [26]. This is achieved by incorporating the arm into the whole-body control framework in a way that enables both motion and interaction force control of arm's gripper. Furthermore, a method is presented to include the torso's orientation for the motion control of the arm, thereby significantly increasing the arm's kinematic reach.

In this article, we unify our recent contributions in robot locomotion and manipulation and provide a comprehensive description that integrates planning and whole-body control into a single framework. Our online motion optimization algorithm produces motion reference trajectories for stable and versatile locomotion, which are tracked by a whole-body controller. This approach enables ANYmal to execute a wide variety of gaits such as walking, trotting, pacing, pronking, a running trot, dynamic walking, and galloping. When ANYmal is equipped with an arm, the same framework can effectively handle tasks that involve both locomotion and manipulation. This diversity of tasks underlines the versatility of the proposed framework.

8.3 MODEL

We model a multi-limbed robot as a floating-base system that interacts with its surroundings by making contact at the end of its legs and arms (see Figure 8.2). We attach a coordinate system $\mathcal{B}$ to the floating base, to which we also refer to as *base*. We indicate the position of the base w.r.t. the inertial frame $\mathcal{I}$ with ${}_I\boldsymbol{r}_{IB} \in \mathbb{R}^3$. The relative orientation from $\mathcal{B}$ to $\mathcal{I}$ is described by the unit quaternion $\boldsymbol{q}_{IB}$. In a standard configuration, ANYmal has four legs, each with three DOF. When used for manipulation tasks, it is additionally equipped with a six-DOF arm. The joint positions are represented by $\boldsymbol{q}_j \in \mathbb{R}^{n_j}$, where $n_j = 12$ for ANYmal alone, and $n_j = 18$ when it is equipped with an arm. We write the absolute linear velocity of the base w.r.t. the inertial frame as ${}_I\boldsymbol{v}_B \in \mathbb{R}^3$, and its angular velocity in base frame as ${}_B\boldsymbol{\omega}_{IB} \in \mathbb{R}^3$. The generalized coordinates $\boldsymbol{q}$ and generalized velocities $\boldsymbol{u}$ of the whole system are given by

$$\boldsymbol{q} = \begin{bmatrix} {}_I\boldsymbol{r}_{IB} \\ \boldsymbol{q}_{IB} \\ \boldsymbol{q}_j \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} {}_I\boldsymbol{v}_B \\ {}_B\boldsymbol{\omega}_{IB} \\ \dot{\boldsymbol{q}}_j \end{bmatrix}, \tag{8.1}$$

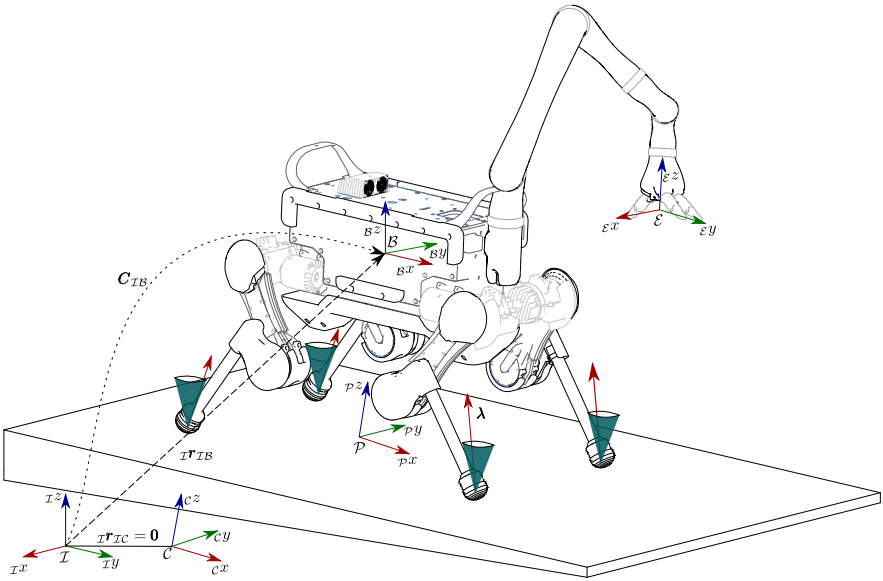


FIGURE 8.2: A sketch of the floating-base system we consider in this article.

where $\dot{\boldsymbol{q}}_j \in \mathbb{R}^{n_j}$ represents the vector of the joint velocities. We assume that contact points are stationary. Hence, when one of the end-effectors (feet or hands) $\mathcal{E}$ is in contact, we write ${}_I\boldsymbol{r}_{I\mathcal{E}} = const$, with ${}_I\boldsymbol{r}_{I\mathcal{E}}$ its position w.r.t. the inertial frame $\mathcal{I}$. Time differentiating this equality twice leads to

$$\begin{aligned} {}_I\boldsymbol{v}_E &= {}_I\boldsymbol{J}_E\boldsymbol{u} = 0, \\ {}_I\boldsymbol{a}_E &= {}_I\dot{\boldsymbol{v}}_E = {}_I\boldsymbol{J}_E\dot{\boldsymbol{u}} + {}_I\dot{\boldsymbol{J}}_E\boldsymbol{u} = 0, \end{aligned} \tag{8.2}$$

where ${}_I\boldsymbol{J}_E = \partial {}_I\boldsymbol{r}_{I\mathcal{E}}/\partial \boldsymbol{q}$ is the Jacobian that maps generalized velocities to absolute linear end-effector velocities expressed in the inertial frame, and ${}_I\dot{\boldsymbol{J}}_E$ is the time derivative of ${}_I\boldsymbol{J}_E$. We use this to write the contact constraints as

$${}_I\boldsymbol{J}_S\dot{\boldsymbol{u}} + {}_I\dot{\boldsymbol{J}}_S\boldsymbol{u} = \mathbf{0}, \tag{8.3}$$

where the *support Jacobian* ${}_I\boldsymbol{J}_S^T = \begin{bmatrix} {}_I\boldsymbol{J}_{E_1}^T & \dots & {}_I\boldsymbol{J}_{E_{n_c}}^T \end{bmatrix}$ is the stack of the Jacobians of all the end-effectors which are in contact, ${}_I\dot{\boldsymbol{J}}_S$ is its time derivative and n_c is the number of contacts.

Finally, the equations of motion of the entire system are written as

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}_S^T\boldsymbol{\lambda}, \tag{8.4}$$

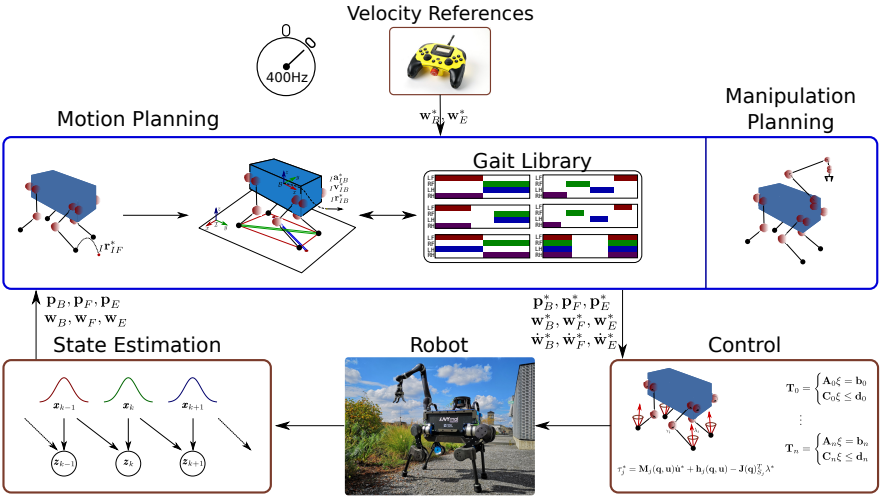


FIGURE 8.3: Overview of the locomotion and manipulation framework. Our framework coordinates locomotion and manipulation by processing velocity commands that drive the robot, or the arm's end-effector, in a desired direction. These external velocity commands are provided from an operator via a joystick or from a navigation module. The planning framework processes these to produce footholds in a specific direction, which in turn define a sequence of support polygons depending on the contact schedule. The sequence is used by the motion planner to generate a dynamically stable motion plan for the COM. The operational space references for the COM, the orientation of the floating base, and the end-effectors that are not in contact with the environment are tracked by a whole-body controller. By solving a hierarchy of tasks, the controller produces reference torques which are sent to low-level controllers on each actuated joint. The main control loop runs at 400Hz, while the motion modules run in parallel loops that are updated at different frequencies.

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n_u \times n_u}$ is the system mass-matrix, $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) \in \mathbb{R}^{n_u}$ is the vector of Coriolis, centrifugal, and gravity terms, $\boldsymbol{S} = \begin{bmatrix} \boldsymbol{0}_{n_\tau \times 6} & \mathbb{I}_{n_\tau \times n_\tau} \end{bmatrix}$ is the selection matrix that describes which DOF are actuated with n_τ the number of actuated joints, $\boldsymbol{\tau} \in \mathbb{R}^{n_\tau}$ is a vector of generalized actuation torques, and $\boldsymbol{\lambda} \in \mathbb{R}^{3n_c}$ is the stack of the reaction forces.

8.4 WHOLE-BODY MOTION PLANNING

Recent advances that address the problem of generating motions for multi-body systems (such as the one in Figure 8.1) have described solutions that are based on trajectory optimization, in which one monolithic optimiza-

tion problem is set up to produce whole-body motion references. These approaches can compute complex motion trajectories but may require, depending on the complexity of the model used in the planner, computation times in a range of hundreds of milliseconds to seconds. The method described in this paper breaks down the typical trajectory optimization scheme into separate optimization-based motion planners. We propose a motion planning framework (see Figure 8.3) that computes reference motions for the so-called *COM frame*, a frame attached at the COM and aligned with the floating base. Hence, the planner generates motions for both translational and rotational DOF of the COM frame, Moreover, our framework computes reference positions for future contact locations and desired motions for all the end-effectors.

We formulate a quadratic optimization problem to compute references for future contacts locations of the feet, which we also refer to as *footholds*. The latter drive the locomotion, and are computed based on the desired walking direction and speed, and a balancing criterion. Two separately executed modules generate motion trajectories for the feet and the gripper.

The motion planner for the COM frame constitutes the core of our planning framework. It implements a receding-horizon nonlinear optimization problem that computes trajectories for the x, y, and z of the COM and for roll, pitch, and yaw angles of the floating base by constraining the ZMP to always lie inside the support polygons formed by the feet that are in contact with the ground. The use of a reduced-complexity model as a balancing criterion allows us to rapidly compute trajectories and to continuously update them. This makes it possible to generate motions that, when tracked, enable the robot to effectively react to unexpected external disturbances that may act on it.

As depicted in Figure 8.3, the input to the motion planner is an external twist command. This originates from a human-handled device (e.g., a joystick), or from a navigation or path following module. The twist commands are used to guide either locomotion or the motion of the gripper.

In the following sections, we describe the underlying optimization problems that are numerically solved to generate the robot's motion references.

8.4.1 *Foot Trajectory and Contact Locations Planning*

Planning footholds for a walking robot is a crucial task that involves retaining balance and reacting to events such self-collision. The reference contact locations should be computed to drive locomotion in a desired direction

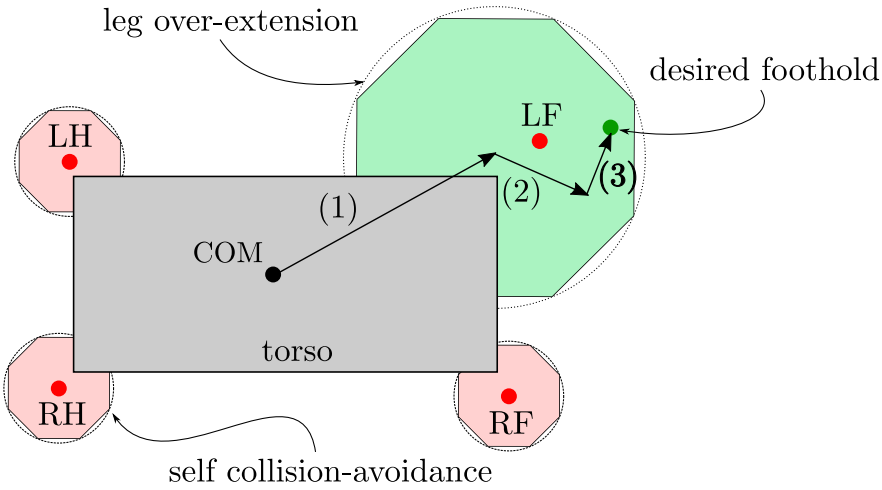


FIGURE 8.4: Top view of the torso. The black arrows illustrate the steps for determining the position of the desired foothold according to the inverted pendulum model [22]: (1) default foothold, (2) velocity projection, and (3) feedback correction. The green octagon visualizes the inequality constraints for avoiding over-extension of the leg and the red octagons exemplify the constraints to avoid collision with the other end-effectors for the left-front (LF) foot.

without violating kinematic constraints, such as maximum leg extension, while allowing the robot to track a motion plan. Additionally, we wish to constrain the computed footholds to avoid self-collision of the legs. As we have previously done in [86], we formulate these requirements as contributions to a cost function and as linear constraints of a QP problem, formulated as

$$\min_{\boldsymbol{\xi}_f} \quad \frac{1}{2}\boldsymbol{\xi}_f^T \boldsymbol{Q}\boldsymbol{\xi}_f + \boldsymbol{c}^T\boldsymbol{\xi}_f \qquad \text{s. t.} \qquad \boldsymbol{D}\boldsymbol{\xi}_f \leq \boldsymbol{f}, \tag{8.5}$$

where $\boldsymbol{\xi}_f$ is a vector of x and y coordinates of the desired footholds for all four feet on the local estimation of the terrain. The latter is computed by fitting a plane through the latest contact locations of each foot (see [22]). We execute the optimization online, together with the main control loop. Table 8.1 summarizes the cost function and the constraints included in equation (8.5). In the cost function of eq. (8.5), we penalize the deviation from nominal footholds that are defined by the user. This is used as a regularization term for the optimized solution. To avoid jumps in the computation of the footholds, we penalize deviation from the previous solution as well.

Finally, to guide locomotion, we penalize deviation from a location that is obtained by forward integrating the external velocity commands that are sent to the motion planning framework. As shown in [86], we obtain reactive behaviour by adding a cost to the deviation from footholds computed based on an inverted pendulum model [22].

To avoid over-extension of the legs, we constrain the desired footholds to lie within a circle whose center is located under the hip joint of the corresponding leg. To get linear constraints, we approximate the circle with a polygon having eight vertices (see Figure 8.4).

TABLE 8.1: A list of the contributions to the cost function and the constraints introduced in the foothold optimization problem.

Type	Task	Purpose
Objective	Inverted pendulum	React to disturbances
Objective	Min. deviation from previous solution	Smooth solutions
Objective	Velocity projection	Guide locomotion
Objective	Nominal foothold	Regularization
Constraint	Maximum leg extension	Avoid kinematic limits
Constraint	Self-collision	Avoid self impacts

The second set of inequality constraints is dedicated to end-effector self-collision avoidance. For each leg we define a circular region on the ground, approximated as an octagon centered at each neighboring end-effector. This circle indicates a region in which a foothold should not be placed. Although they are non-convex, these constraints can be converted into a set of convex QPs that are solved in a cascade: In a first step, the collision constraints are disabled. If the solution lies outside of the feasible region, we solve for each line constraint of the polygon as a separate optimization problem. The global solution is the one that produces the smallest cost.

As done in previous works [22], the planned contact locations for the feet are used to compute swing foot trajectories by fitting a fifth-order polynomial spline through the lift-off location and the desired foothold for each foot. The spline is obtained by specifying desired lift-off and touch-down velocities and accelerations.

8.4.2 *Center of Mass and Floating-Base Orientation Motion Planning*

To allow a walking robot to navigate in its surroundings and keep balance at all times, appropriate trajectories must be designed. The latter is a challenging problem: we want these motions to be rapidly computed to react to unexpected variations in the surroundings or to external disturbances acting on the system.

At the core of our motion planning framework lies a planner that generates trajectories for the COM and for the orientation of the floating-base. The planner assumes that the whole system is equivalent to a single rigid body of mass equal to the total mass of the system and with inertia tensor equal to the one of the floating base. This assumption enables the use of simplified dynamics that lead to the integration of balancing criterions [96] that can be efficiently evaluated on the system. In particular, we use constraints based on the desired ZMP location as a balancing criterion. The ZMP [43] is formulated as a function of the position $\boldsymbol{r}_{COM}$ and acceleration $\ddot{\boldsymbol{r}}_{COM}$ of the COM of the whole system, and the rate of change of angular momentum $\dot{\boldsymbol{l}}$ of the multi-body system. This fact allows the ZMP to be integrated in a motion planning framework to obtain trajectories for the COM and the orientation of the floating base.

The motion planner is implemented as a receding-horizon optimization problem that computes trajectories for the pose, velocities and accelerations of the COM frame $\boldsymbol{p}_{PC}$ w.r.t. to the plan frame P, the latter being a frame that we introduce in more detail in Section 8.4.2.2. We write the pose

of the COM frame $\boldsymbol{p}_{PC}$ as $\begin{bmatrix}\boldsymbol{r}_{PC}^T & \boldsymbol{\chi}_{PB}^T\end{bmatrix}^T$, where $\boldsymbol{r}_{PC}$ is the position of the COM and $\boldsymbol{\chi}_{PB} = \begin{bmatrix}\psi & \theta & \phi\end{bmatrix}^T$ is the orientation, parametrized as ZYX Tait-Bryan Euler angles, of the floating base. Henceforth, we refer to $\boldsymbol{p}_{PC}$ as the pose of the *COM frame*: a frame located at the COM and aligned with the floating base. The trajectories are computed w.r.t. the so-called *virtual plane frame P*, a frame that is located at the center of the feet locations and that is aligned with the local estimation of the terrain. The planner computes positions, velocities, and accelerations for both the linear $\boldsymbol{r}_{PC}$ and angular $\boldsymbol{\chi}_{PB}$ part of $\boldsymbol{p}_{PC}$. These quantities are computed by solving a nonlinear optimization problem written as

$$\min_{\boldsymbol{\xi}} \quad f(\boldsymbol{\xi}) \quad \text{s. t.} \quad \boldsymbol{h}(\boldsymbol{\xi}) = \boldsymbol{0}, \quad \boldsymbol{g}(\boldsymbol{\xi}) \leq \boldsymbol{0}, \tag{8.6}$$

where $\boldsymbol{h}(\boldsymbol{\xi})$ and $\boldsymbol{g}(\boldsymbol{\xi})$ are equality and inequality constraint functions which can be nonlinear in the solution vector $\boldsymbol{\xi}$. As will be described in the following sections, we define $\boldsymbol{\xi}$ as the vector of coefficients that parametrize the trajectories of the COM frame. The nonlinearity of the constraints originates from the definition of the ZMP that is nonlinear w.r.t. optimization vector.

In the following sub-sections, we thoroughly discuss the implementation of the optimization problem and its formulation. We initially introduce the way we describe a sequence of contacts and the frame that we use to compute the trajectories of the COM frame. Finally, we describe the ZMP-based balancing and feasibility constraints and the contributions to the cost function and constraints introduced in eq. (8.6).

8.4.2.1 *Contact Schedules*

During locomotion, a walking system is continuously changing the contact state of its supporting end-effectors. The sequence of switches in the contact state defines a contact schedule, or *gait*. For instance, slower walking gaits break contact with one foot at a time, while a trotting gait has two feet in contact at all times. The gaits that we consider in this article are periodic, meaning that the contact sequence repeats over time. For this reason, the contact sequence is also termed *gait pattern*. As done in our previous works, we describe a gait by defining *lift-off* ϕ_{lo} and *touch-down* ϕ_{td} phases for each foot. The time duration of a gait pattern, or stride duration, is represented by τ_s. The stride duration can be used to convert phases to time domain by writing $\tau_s\phi_{lo}$ and $\tau_s\phi_{td}$. The phase φ_s of the pattern, with

values in the range $[0, 1)$, represents which part of the pattern is currently being processed.

For each gait pattern, the user pre-specifies both the timing of the contact sequence and the stride duration. The latter, however, is not kept constant during the execution of the gait but is continuously updated as a function of both the user-specified duration τ_s and a term dependent on the velocity tracking error. We write the function for the updated τ_s^* as

$$\tau_s^* = \tau_s - \sqrt{\begin{bmatrix} \boldsymbol{e}_v & \boldsymbol{e}_\omega \end{bmatrix} \boldsymbol{K}_p^T \boldsymbol{K}_p \begin{bmatrix} \boldsymbol{e}_v \\ \boldsymbol{e}_\omega \end{bmatrix}}, \tag{8.7}$$

where $\boldsymbol{K}_p$ is a diagonal and positive definite feedback gain, $\boldsymbol{e}_v$ the filtered linear velocity tracking error, and $\boldsymbol{e}_\omega$ the filtered angular velocity tracking error. To ensure that the stride duration does not decrease to a negative value, the velocity dependent term in eq. (8.7) is saturated such that $\tau_s^* > \tau_{s,min} > 0$, where $\tau_{s,min}$ is a user defined tuning parameter.

To allow the motion optimizer to handle user-commanded switches between gait patterns, we add a standing gait (i.e., all legs are planned to be in contact) between the current and the next gait.

8.4.2.2 *Virtual Plane Frame*

The trajectories computed by the motion planner must be computed w.r.t. a reference frame. Since the motions are continuously updated online, and because the estimated location of the floating base is subject to drift over time [57], we introduce the so-called *virtual plane frame* P. Moreover, planning in such a frame that is aligned with the floating base allows to easily specify any planner parameters in the heading and lateral direction of locomotion.

The pose of frame P depends on the local estimation of the terrain [22], that is obtained by fitting a plane through the most recent contact locations. The virtual plane frame P is aligned with the local estimation of the terrain and is located at the footprint center[1] projected onto the terrain. Along the optimization horizon of $\boldsymbol{p}_{PC}$, we keep the location of the virtual plane frame fixed.

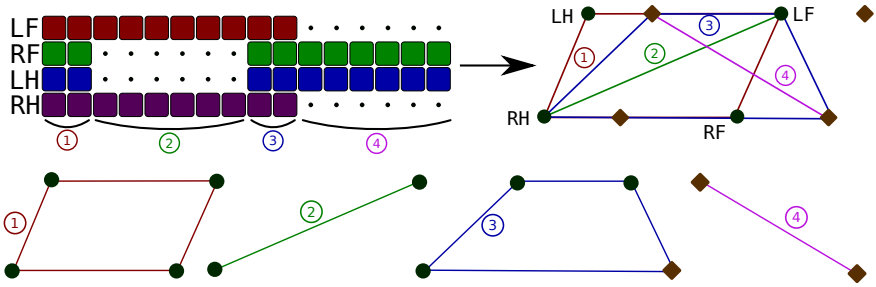


FIGURE 8.5: An illustration of how a sequence of support polygons is obtained from the gait pattern and the set of current and desired contact locations. The gait pattern depicts a trotting gait. The circles represent the current contact locations, while the diamonds represent the planned ones. When only two feet are in contact, the support polygon is a line. In our optimization, we expand the line to a quadrilateral.

8.4.2.3 *Support Polygon Sequence*

The prespecified contact schedule and the planned contact locations allow us to describe a sequence of future support polygons. The latter provides information about where the ZMP must lie to ensure both balancing and dynamic feasibility of the planned trajectories. As shown in Figure 8.5, we obtain a sequence of support polygons by combining information provided by the current contact locations and by the gait pattern. The sequence is generated by advancing through the gait pattern, starting at the current phase, and storing the vertices of a polygon every time a contact state change is found. We end the sequence when a full stride duration is covered. This, in turn, also defines the optimization horizon τ_h to be equal to the stride duration τ_s.

8.4.2.4 *Optimization Domain*

We represent the trajectories for each DOF of the COM frame as a sequence of fifth-order polynomial splines, which allows to set position, velocity and acceleration constraints . This parametrization allows us to set up an optimization problem whose solution will consist of optimal spline coefficients. A fifth-order spline is written as

$$s(t) = \alpha_{i,5}t^5 + \alpha_{i,4}t^4 + \alpha_{i,3}t^3 + \alpha_{i,2}t^2 + \alpha_{i,1}t + \alpha_{i,0}. \tag{8.8}$$

1 The footprint center is the average of the current foot positions.

By representing the i-th spline coefficients with

$$\boldsymbol{\alpha}_i = \begin{bmatrix} \alpha_{i5} & \alpha_{i4} & \alpha_{i3} & \alpha_{i2} & \alpha_{i1} & \alpha_{i0} \end{bmatrix}^T, \tag{8.9}$$

and by collecting the time-dependent terms in the so-called time vector $\boldsymbol{t}(t) = \begin{bmatrix} t^5 & t^4 & t^3 & t^2 & t & 1 \end{bmatrix}^T$, we write the i-th spline $\boldsymbol{s}_i(t)$ as

$$\boldsymbol{s}_i(t) = \boldsymbol{\alpha}_i^T \boldsymbol{t}(t). \tag{8.10}$$

Differentiating equation (8.10) w.r.t. time yields

$$\dot{\boldsymbol{s}}_i(t) = \boldsymbol{\alpha}_i^T \dot{\boldsymbol{t}}(t), \tag{8.11}$$

with $\dot{\boldsymbol{t}}(t) = \begin{bmatrix} 5t^4 & 4t^3 & 3t^2 & 2t & 1 & 0 \end{bmatrix}^T$. We write the second derivative as

$$\ddot{\boldsymbol{s}}_i(t) = \boldsymbol{\alpha}_i^T \ddot{\boldsymbol{t}}(t), \tag{8.12}$$

with $\ddot{\boldsymbol{t}}(t) = \begin{bmatrix} 20t^3 & 12t^2 & 6t & 2 & 0 & 0 \end{bmatrix}^T$. The total number of splines n_s per DOF depends on the number of support polygons and the number of splines used to describe the motion through each support polygon[2]. We write the stack of coefficients to be optimized for as

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\alpha}_x^T & \boldsymbol{\alpha}_y^T & \boldsymbol{\alpha}_z^T & \boldsymbol{\alpha}_\psi^T & \boldsymbol{\alpha}_\theta^T & \boldsymbol{\alpha}_\phi^T \end{bmatrix}^T, \tag{8.13}$$

where

$$\begin{aligned} \boldsymbol{\alpha}_x &= \begin{bmatrix} \boldsymbol{\alpha}_{x,1} \\ \vdots \\ \boldsymbol{\alpha}_{x,n_s} \end{bmatrix}, \boldsymbol{\alpha}_y = \begin{bmatrix} \boldsymbol{\alpha}_{y,1} \\ \vdots \\ \boldsymbol{\alpha}_{y,n_s} \end{bmatrix}, \boldsymbol{\alpha}_z = \begin{bmatrix} \boldsymbol{\alpha}_{z,1} \\ \vdots \\ \boldsymbol{\alpha}_{z,n_s} \end{bmatrix}, \\ \boldsymbol{\alpha}_\psi &= \begin{bmatrix} \boldsymbol{\alpha}_{\psi,1} \\ \vdots \\ \boldsymbol{\alpha}_{\psi,n_s} \end{bmatrix}, \boldsymbol{\alpha}_\theta = \begin{bmatrix} \boldsymbol{\alpha}_{\theta,1} \\ \vdots \\ \boldsymbol{\alpha}_{\theta,n_s} \end{bmatrix}, \boldsymbol{\alpha}_\phi = \begin{bmatrix} \boldsymbol{\alpha}_{\phi,1} \\ \vdots \\ \boldsymbol{\alpha}_{\phi,n_s} \end{bmatrix}, \end{aligned} \tag{8.14}$$

where $\boldsymbol{\alpha}_{x,i} = \begin{bmatrix} \alpha_{i5}^x & \alpha_{i4}^x & \alpha_{i3}^x & \alpha_{i2}^x & \alpha_{i1}^x & \alpha_{i0}^x \end{bmatrix}^T$. The same notation applies to the spline coefficients of the remaining DOF.

2 We use one fifth-order spline per support polygon and DOF.

Using the notation introduced so far, we express the position $\boldsymbol{r}_{PC}$ of the COM frame w.r.t. the virtual plane frame P at time t as

$$\begin{aligned}\boldsymbol{r}_{PC}(t) &= \begin{bmatrix} s_i^x(t-\Delta t_{i-1}) \\ s_i^y(t-\Delta t_{i-1}) \\ s_i^z(t-\Delta t_{i-1}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{t}^T(t-\Delta t_{i-1})\boldsymbol{\alpha}_{x,i} \\ \boldsymbol{t}^T(t-\Delta t_{i-1})\boldsymbol{\alpha}_{y,i} \\ \boldsymbol{t}^T(t-\Delta t_{i-1})\boldsymbol{\alpha}_{z,i} \end{bmatrix} \\ &= \boldsymbol{T}(t-\Delta t_{i-1}) \begin{bmatrix} \boldsymbol{\alpha}_{x,i} \\ \boldsymbol{\alpha}_{y,i} \\ \boldsymbol{\alpha}_{z,i} \end{bmatrix},\end{aligned} \tag{8.15}$$

with

$$\boldsymbol{T}(t) = \begin{bmatrix} \boldsymbol{t}^T(t) & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{t}^T(t) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{t}^T(t) \end{bmatrix}, \tag{8.16}$$

where Δt_{i-1} is the sum of the time durations of all splines from index 0 to $i-k$, and k is the index of the currently active spline, with $\Delta t_{i-1} \leq t$. Similarly, we write the main body orientation $\boldsymbol{\chi}_{PB}$ w.r.t. the plan frame at time t as

$$\boldsymbol{\chi}_{PB}(t) = \begin{bmatrix} s_i^{\psi}(t-\Delta t_{i-1}) \\ s_i^{\theta}(t-\Delta t_{i-1}) \\ s_i^{\phi}(t-\Delta t_{i-1}) \end{bmatrix} = \boldsymbol{T}(t) \begin{bmatrix} \boldsymbol{\alpha}_{\psi,i} \\ \boldsymbol{\alpha}_{\theta,i} \\ \boldsymbol{\alpha}_{\phi,i} \end{bmatrix}. \tag{8.17}$$

The spatial positions, velocities, and accelerations of the COM frame state are compactly written as

$$\begin{aligned}\boldsymbol{p}_{PC}(t) &= \boldsymbol{V}(t-\Delta t_{i-1})\boldsymbol{\xi}_i, \\ \dot{\boldsymbol{p}}_{PC}(t) &= \dot{\boldsymbol{V}}(t-\Delta t_{i-1})\boldsymbol{\xi}_i, \\ \ddot{\boldsymbol{p}}_{PC}(t) &= \ddot{\boldsymbol{V}}(t-\Delta t_{i-1})\boldsymbol{\xi}_i,\end{aligned} \tag{8.18}$$

with the stacked matrix

$$\boldsymbol{V}(t) = \begin{bmatrix} \boldsymbol{T}(t) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T}(t) \end{bmatrix}. \tag{8.19}$$

8.4.2.5 *Motion Regularization*

To avoid large deviations of the COM, to hold the torso at a desired height, to ensure velocity tracking along the trajectory, and to substitute

the motion plan for the non optimized DOF, we define a nominal trajectory $\boldsymbol{\pi}(t) \in \mathbb{R}^6$ for the COM frame that we call *path regularizer*. It represents a high-level approximation of the entire motion plan. We implement the path regularizer as a single fifth-order polynomial spline. The coefficients of the latter are computed by setting initial and final constraints on the linear and angular state of $\boldsymbol{\pi}(t)$. The constraints are formulated as

- Initial linear constraints: The initial position is equal to the measured COM position for x and y coordinates, and the z component is set to a desired height above the plane frame. The initial linear velocity is set equal to the desired velocity $\boldsymbol{v}_{des}$ originated from the external twist commands, and the acceleration is set to zero.

- Final linear constraints: The final position of $\boldsymbol{\pi}(t)$ is set on the curved path defined defined by integrating the external commanded velocities $\boldsymbol{v}_{des}$ and $\boldsymbol{\omega}_{des}$. Since we assume that the latter are constant during the optimization horizon τ_h, the final position is located along that curve (see Figure 8.6). When $\boldsymbol{\omega}_{des} = \mathbf{0}$, the resulting path regularizer $\boldsymbol{\pi}(t)$ is a line. The final linear acceleration is set to zero.

- Initial angular constraint: The initial orientation is set equal to the one described by $\boldsymbol{C}_{PC}$, i.e., the orientation between the control and the plane frame. The initial angular velocity is set equal to the desired one $\boldsymbol{\omega}_{des}$, and the initial angular acceleration is zero.

- Final angular constraint: The final angle is obtained by adding a velocity projection $\boldsymbol{\omega}_{des}\tau_h$ to the initial knot. The final angular velocity is obtained by rotating the initial velocity into the final configuration $\boldsymbol{C}_{PC}\boldsymbol{C}_z(\boldsymbol{\omega}_{des}\tau_h)\boldsymbol{\omega}_{des}$, where $\boldsymbol{C}_z$ defines an elementary rotation around the z axis. The final angular acceleration is set to zero.

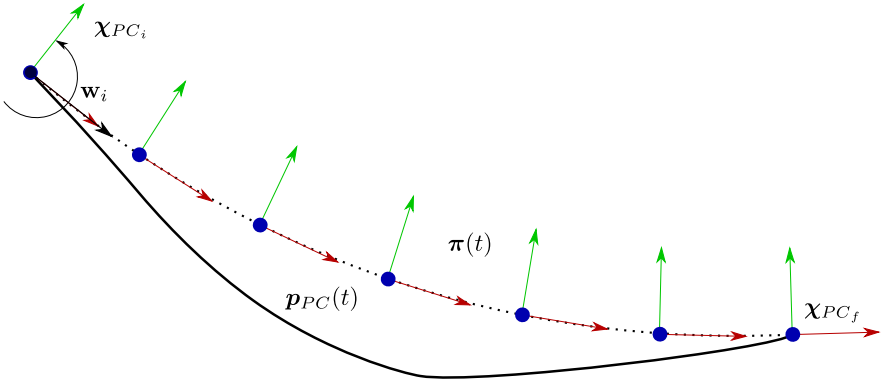


FIGURE 8.6: The path regularizer $\pi(t)$ is an approximation of the motion trajectories $p_{PC}(t)$ computed by the COM frame planner. It is implemented as a fifth-order spline with constraints on linear and angular position, velocity and acceleration on the initial and final points. The coordinate frames shown in the figure represent the desired orientation at a specific sample point of the path regularizer. When the external desired angular velocity ω_{des} is non-zero, $\pi(t)$ is equal to a curve. When $\omega_{des} = \mathbf{0}$, the path regularizer is a straight line.

8.4.2.6 *Soft Equality Constraints*

To obtain a smooth motion plan, we minimize the acceleration over the full trajectory by adding a quadratic term $Q_{\ddot{p},i}$ to the cost function, defined as [62]

$$Q_{\ddot{p},i} = \begin{bmatrix} (400/7)t_{f,i}^7 & 40t_{f,i}^6 & 24t_{f,i}^5 & 10t_{f,i}^4 & \\ 40t_{f,i}^6 & 28.8t_{f,i}^5 & 18t_{f,i}^4 & 8t_{f,i}^3 & \vdots \\ 24t_{f,i}^5 & 18t_{f,i}^4 & 12t_{f,i}^3 & 6t_{f,i}^2 & \mathbf{0}_{4\times 2} \\ 10t_{f,i}^4 & 8t_{f,i}^3 & 6t_{f,i}^2 & 4t_{f,i} & \vdots \\ \cdots & \mathbf{0}_{2\times 4} & \cdots & & \mathbf{0}_{2\times 2} \end{bmatrix}, \tag{8.20}$$

where i is the index of the i-th spline and $t_{f,i}$ its duration in seconds. The Hessian matrix in eq. 8.20 is obtained by solving the integral

$$\begin{aligned} \int_0^{t_f} \left(\ddot{\boldsymbol{t}}^T \boldsymbol{\alpha}_i\right)^T \ddot{\boldsymbol{t}}^T \boldsymbol{\alpha}_i dt &= \boldsymbol{\alpha}_i^T \left(\int_0^{t_f} \ddot{\boldsymbol{t}}\ddot{\boldsymbol{t}}^T dt\right) \boldsymbol{\alpha}_i \\ &= \boldsymbol{\alpha}_i^T \boldsymbol{Q}_{\ddot{p},i} \boldsymbol{\alpha}_i. \end{aligned} \tag{8.21}$$

The overall contribution of the acceleration minimization to the hessian Q of the planner's cost function is then

$$Q_{\ddot{p}} = \begin{bmatrix} Q_{\ddot{p},1} & \cdots & \cdots & \mathbf{0}_{6\times 6} \\ \mathbf{0}_{6\times 6} & Q_{\ddot{r},2} & \cdots & \vdots \\ \vdots & \mathbf{0}_{6\times 6} & \ddots & \mathbf{0}_{6\times 6} \\ \mathbf{0}_{6\times 6} & \cdots & \cdots & Q_{\ddot{p},n_s} \end{bmatrix}. \tag{8.22}$$

TABLE 8.2: A summary of the main contributions to the cost function and the constraints introduced in the COM frame optimization.

Type	Task	Purpose
Objective	Path regularizer	Trajectory approximation
Objective	Min. acceleration	Smooth trajectories
Objective	Deviation from previous solution	Consistency with previous plan
Constraint	ZMP	Balancing
Constraint	Spline junctions	Connected splines
Constraint	$\boldsymbol{n}^T \boldsymbol{f}^{gi} < 0$	Unilateral forces
Constraint	Relaxed ZMP	Feasibility of the solution

To penalize the deviation of the optimal plan from the regularizer trajectory $\boldsymbol{\pi}(t)$ described in Section 8.4.2.5, we add for each sample point of the COM trajectory the costs

$$\begin{aligned}
&(\boldsymbol{p}_{PB}(t) - \boldsymbol{\pi}(t))^T \boldsymbol{Q}_{\pi} (\boldsymbol{p}_{PB}(t) - \boldsymbol{\pi}(t)), \\
&(\dot{\boldsymbol{p}}_{PB}(t) - \dot{\boldsymbol{\pi}}(t))^T \boldsymbol{Q}_{\dot{\pi}} (\dot{\boldsymbol{p}}_{IB}(t) - \dot{\boldsymbol{\pi}}(t)), \\
&(\ddot{\boldsymbol{p}}_{PB}(t) - \ddot{\boldsymbol{\pi}}(t))^T \boldsymbol{Q}_{\ddot{\pi}} (\ddot{\boldsymbol{p}_{PB}}(t) - \ddot{\boldsymbol{\pi}}(t)),
\end{aligned} \tag{8.23}$$

where $\boldsymbol{Q}_{\pi}$, $\boldsymbol{Q}_{\dot{\pi}}$, and $\boldsymbol{Q}_{\ddot{\pi}}$ are diagonal positive-definite weighting matrices. Since the planner continuously recomputes motions, we avoid large deviations between two successive solution by minimizing deviation of the current plan from the previous one $\boldsymbol{p}_{PB,previous}$, $\dot{\boldsymbol{p}}_{PB,previous}$, and $\ddot{\boldsymbol{p}}_{PB,previous}$ by sampling the previous trajectory and incorporating the objective described in eq. (8.23).

8.4.2.7 *Hard Inequality Constraints: Balancing and Feasibility*

The trajectories generated by the motion planner must be designed to be feasible (i.e., there exists a set of contact forces that can track them) and guarantee balance when tracked by a controller. As discussed in [43, 96], the stability of a walking system can be ensured by constraining the ZMP to lie in the support polygon. The ZMP is computed as [43]

$$\boldsymbol{r}_{ZMP} = \frac{\boldsymbol{n} \times \boldsymbol{\tau}_P^{gi}}{\boldsymbol{n}^T \boldsymbol{f}^{gi}}, \tag{8.24}$$

with $\boldsymbol{n}$ the normal to the plane that approximates the terrain on which the robot is walking. The moment $\boldsymbol{\tau}_P^{gi} \in \mathbb{R}^3$ around the origin of the plane frame P and the force $\boldsymbol{f}^{gi} \in \mathbb{R}^3$ are the components of the so-called *gravito-inertial* wrench [97], defined as

$$\boldsymbol{f}^{gi} = m(\boldsymbol{g} - \ddot{\boldsymbol{r}}_{COM}), \quad \boldsymbol{\tau}_P^{gi} = \boldsymbol{r}_{COM} \times \boldsymbol{f}^{gi} - \dot{\boldsymbol{l}}, \tag{8.25}$$

with m the total mass of the robot, $\boldsymbol{g}$ the gravitational acceleration and $\dot{\boldsymbol{l}} \in \mathbb{R}^3$ the rate of change of the angular momentum of the whole system. In our previous work [86], we assumed that $\dot{\boldsymbol{l}} = \boldsymbol{0}$ and hence did not generate rotational motions. In this article, we drop this assumption to generate motions for gaits which require angular references as well (e.g., galloping and bounding).

The constraints on the location of the ZMP formulated as a function of the edges of the support polygons introduced in Section 8.4.2.3. Each edge

of the polygons is written as $ax + by + c = 0$, with a, b, and c the coefficients of the line to which the edge belongs. We use this formulation to write the ZMP constraints as

$$\begin{aligned} \boldsymbol{d}^T \boldsymbol{r}_{ZMP} + c = \\ \boldsymbol{d}^T \frac{\boldsymbol{n} \times \boldsymbol{\tau}_P^{gi}}{\boldsymbol{n}^T \boldsymbol{f}^{gi}} + c \geq 0, \end{aligned} \tag{8.26}$$

where $\boldsymbol{d}^T = \begin{bmatrix} a & b & 0 \end{bmatrix}$. Multiplying both sides of the last equation by $\boldsymbol{n}^T \boldsymbol{f}^{gi}$ yields

$$\underbrace{\boldsymbol{d}^T (\boldsymbol{n} \times \boldsymbol{\tau}_P^{gi}) + c \boldsymbol{n}^T \boldsymbol{f}^{gi}}_{h(\boldsymbol{r}_{ZMP})} \leq 0. \tag{8.27}$$

The inequality sign change in eq. (8.27) derives from the additional constraint $\boldsymbol{n}^T \boldsymbol{f}^{gi} = m \boldsymbol{n}^T (\boldsymbol{g} - \ddot{\boldsymbol{r}}_{COM}) < \beta$, with $\beta > 0$. The constraint ensures that the forces required to track the planned trajectories only push against the terrain. The positive parameter β ensures that the constraint is strictly negative. In our planner, we use $\beta = 10^{-7}$.

To include the constraints in eq. (8.27) in our planner, we compute the gradient of the ZMP constraint $\boldsymbol{\nabla} h_{ZMP}$ analytically. The reader is referred to the Appendix, where we fully describe the derivatives of $\boldsymbol{\nabla} h_{ZMP}$ w.r.t. $\boldsymbol{r}_{PG}$, $\ddot{\boldsymbol{r}}_{PG}$, $\boldsymbol{\chi}$, $\dot{\boldsymbol{\chi}}$, and $\ddot{\boldsymbol{\chi}}$.

8.4.2.8 *Soft Inequality Constraints*

Due to tracking errors, the initial ZMP position might lie outside of the first support polygon. To avoid infeasibility of the optimization problem and rough accelerations originating from the planner trying to push the ZMP into the support area, we soften the first n_{ineq} ZMP constraints described in (8.26), where n_{ineq} is a tuning parameter. To do so, we introduce the slack variables $\boldsymbol{\epsilon}_{ineq} \geq \boldsymbol{0}$ and relax the constraints $\boldsymbol{c}_{ineq} \geq -\boldsymbol{\epsilon}_{ineq}$, where $\boldsymbol{c}_{ineq} = [h_{ZMP,1}, \ldots, h_{ZMP,n_{ineq}}]^T$. The slack variables are then minimized using the additional cost $\lambda_{lin} \boldsymbol{\epsilon}_{ineq} + \lambda_{quad} ||\boldsymbol{\epsilon}_{ineq}||_2^2$.

Large deviations from the path regularizer are suppressed by minimizing the largest overshoot. This is particularly useful when generating motions for a running trot, where we wish to limit the deviation of the z

component of the COM from the path regularizer. For this purpose, we introduce another slack variable ϵ_π and add the additional constraints

$$\begin{aligned} \boldsymbol{p}_{PB}(t) - \boldsymbol{\pi}(t) &\leq \epsilon_\pi \\ \boldsymbol{p}_{PB}(t) - \boldsymbol{\pi}(t) &\geq -\epsilon_\pi \\ \epsilon_\pi &\geq 0, \end{aligned} \tag{8.28}$$

together with the cost $\lambda_{lin}\boldsymbol{\epsilon}_\pi + \lambda_{quad}||\boldsymbol{\epsilon}_\pi||_2^2$. Since $\boldsymbol{\pi}$ is a function of time, the trajectory needs to be sampled and each sample point introduces $2n_{DOF}$ additional inequality constraints of the form introduced in eq. (8.28).

8.4.2.9 *Hard Equality Constraints*

The initial conditions for the linear position and the velocity are implemented as hard constraints, where the target values are obtained by combining, with an α-filter, the measured state with the one obtained from the previous motion plan.

Since the motion plan is parametrized as a sequence of polynomial splines (see Section 8.4.2), we ensure that the overall trajectory is connected, i.e., we set hard equality constraints to connect the final state of spline i with the initial one of spline $i+1$. Hence, we introduce the hard constraints

$$\begin{bmatrix} \boldsymbol{V}(t_s)^T & -\boldsymbol{V}(0)^T \\ \dot{\boldsymbol{V}}(t_s)^T & -\dot{\boldsymbol{V}}(0)^T \\ \ddot{\boldsymbol{V}}(t_s)^T & -\ddot{\boldsymbol{V}}(0)^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_i \\ \boldsymbol{\xi}_{i+1} \end{bmatrix} = \boldsymbol{0}. \tag{8.29}$$

Junction constraints for accelerations are dropped if two adjacent support polygons do not intersect, allowing the ZMP to jump in order to overcome the gap between the two consecutive support polygons .

Junction conditions are formulated differently if two spline segments are connected through a *full flight phase*. In this case, the dynamics of the COM are described by $\ddot{\boldsymbol{r}}_{COM} = \boldsymbol{g}$. Integrating the dynamics yields

$$\begin{aligned} \dot{\boldsymbol{r}}_{COM}(t) &= \boldsymbol{g}t + \dot{\boldsymbol{r}}_{COM}(0), \\ \boldsymbol{r}_{COM}(t) &= \frac{1}{2}\boldsymbol{g}t^2 + \dot{\boldsymbol{r}}_{COM}(0)t + \boldsymbol{r}_{COM}(0). \end{aligned} \tag{8.30}$$

We use eq. 8.30 to write equality constraints for touch-down after a full flight phase as

$$\begin{aligned}\ddot{\boldsymbol{V}}(0)\boldsymbol{\xi}_{s+1} &= \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{0} \end{bmatrix}, \\ \dot{\boldsymbol{V}}(0)\boldsymbol{\xi}_{s+1} &= \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{0} \end{bmatrix} t_f + \dot{\boldsymbol{V}}(t_s)\boldsymbol{\xi}_s, \\ \boldsymbol{V}(0)\boldsymbol{\xi}_{s+1} &= \frac{1}{2}\begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{0} \end{bmatrix} t_f^2 + \left[\dot{\boldsymbol{V}}(t_s)t_f + \boldsymbol{V}(t_s)\right]\boldsymbol{\xi}_s,\end{aligned} \tag{8.31}$$

with t_f the full flight duration. A similar method is employed for initial and final conditions if the first or the last support polygon correspond to a full flight phase.

8.4.2.10 *Line search*

When the optimization returns a solution, the initial conditions of the plan may be significantly deviated from the actual measurements. This is due to the numerical optimization time that, depending on the optimization parameters and the number of support polygons available in the optimization horizon, may become larger than the control frequency. To mitigate this issue, we setup a separate optimization problem formulated as

$$t^* = \arg\min_{\tau_h > t \geq 0} \left\{ ||\boldsymbol{p}_{PB}(t) - \boldsymbol{p}_{PB,\text{meas}}||_2^2 \right\}. \tag{8.32}$$

In eq.(8.32), we search for the time t^* of the optimized plan at which the planned state deviates less from the actual measured one in a least square sense. We use the optimized time to shift the planned motion to reduce the deviation of the plan from the current state of the robot. The optimization problem (8.32) is solved using a back-tracing Newton method algorithm that usually converges within three iterations.

8.5 MOTION PLANNING FOR MANIPULATION

For manipulation tasks, motion references for the gripper's linear and angular motion are required. These can come from different sources such as a specialized path planner, or from a joystick velocity input which updates a reference pose and/or reference velocity of the gripper. Updating the

desired position ${}_I\boldsymbol{r}^d_{IG}$ and orientation quaternion $\boldsymbol{q}_{IG}$ of a reference pose over a time step dt using (joystick-generated) velocity inputs $[{}_I\boldsymbol{v}^d_G, {}_G\boldsymbol{\omega}_{IG}]^T$ is done according to

$$\begin{aligned} {}_I\boldsymbol{r}^d_{IG}(t+dt) &= {}_I\boldsymbol{r}_{IG}(t) + {}_I\boldsymbol{v}^d_G\, dt \\ \boldsymbol{q}^d_{IG}(t+dt) &= \boldsymbol{q}^d_{IG}(t) \boxplus {}_G\boldsymbol{\omega}^d_{IG}\, dt. \end{aligned} \tag{8.33}$$

The $\boxplus$ operator [53] is defined as

$$\begin{aligned} \boxplus : SO(3) \times \mathbb{R}^3 &\to SO(3), \\ \boldsymbol{\Phi}, \boldsymbol{\varphi} &\mapsto \exp(\boldsymbol{\varphi}) \circ \boldsymbol{\Phi}, \end{aligned} \tag{8.34}$$

where $\boldsymbol{\Phi} \in SO(3)$ is a generic $3D$ orientation, $\boldsymbol{\varphi} \in \mathbb{R}^3$ is an Euler vector representing an orientation difference, and $\circ$ is an orientation concatenation operator.

The update rule in eq. 8.33 generates a pose for the gripper w.r.t. the inertial frame, independently of where the floating-base is located. This approach allows to decouple the reference motion of the gripper from that of the base, allowing, e.g., to keep the gripper at fixed pose. While locomoting over long distances, however, we generate references for the gripper w.r.t. the base. Our framework allows to user to choose which of the methods have to be used to generate poses for the gripper.

8.6 CONTROL

The problem of generating control references in a multi-contact scenario can be decomposed into solving a set of smaller problems (called *tasks*), which are solved using a HO in this work. This framework is embedded in a WBC that produces joint torque references using the full rigid-body dynamics. Each of the tasks in the HO is formulated as an optimization problem that computes optimal generalized accelerations $\dot{\boldsymbol{u}}^*$ and reaction forces $\boldsymbol{\lambda}^*$), collected together to form the vector of decision variables $\boldsymbol{\xi}^*_c$. Adding tasks as optimization problems allows to set both equality and inequality constraints on the final solution. Some tasks, such as motion tracking, are added as equality constraints, whereas others, such as actuator limits, are added in the form of inequality constraints. The hierarchical optimization allows for a strict prioritization of the whole-body control tasks whereby the lower priority tasks are fulfilled as well as possible without interrupting the execution of higher priority tasks. For this reason, we chose the

highest priority control task to be a subset of the robot's equations of motion, which links the external forces acting on the robot, to the total change of momentum of the system. This guarantees that the solutions for all other lower priority control tasks (e.g., motion tracking) are always dynamically feasible.

8.6.1 *Hierarchical Optimization*

Each task T_p with priority p in the hierarchy is described as a set of equality and/or inequality constraints as

$$\boldsymbol{T}_p : \begin{cases} \boldsymbol{W}_{eq,p}(\boldsymbol{A}_p\boldsymbol{\xi}_c - \boldsymbol{b}_p) = \boldsymbol{0} \\ \boldsymbol{W}_{ineq,p}(\boldsymbol{D}_p\boldsymbol{\xi}_c - \boldsymbol{f}_p) \leq \boldsymbol{0} \end{cases}, \tag{8.35}$$

where $\boldsymbol{A}_p$ and $\boldsymbol{D}_p$ are the equality and inequality constraint Jacobians respectively, $\boldsymbol{b}_p$ is the vector of target values for the equality constraints, $\boldsymbol{f}_p$ is the vector of max values for the inequality constraints, and $\boldsymbol{W}_{eq,p}$ and $\boldsymbol{W}_{ineq,p}$ are diagonal positive-definite matrices used to weigh tasks that have the same priority. To obtain a solution for a single task T_p, an optimization problem can be set up as

$$\begin{aligned} \min_{\boldsymbol{\xi}_{c,p}, \boldsymbol{v}_p} \quad & \|\boldsymbol{A}_p\boldsymbol{\xi}_{c,p} - \boldsymbol{b}_p\|^2_{\boldsymbol{W}_{eq,p}} + \|\boldsymbol{v}_p\|^2_{\boldsymbol{W}_{ineq,p}} \\ \text{s. t.} \quad & \boldsymbol{D}_p\boldsymbol{\xi}_{c,p} - \boldsymbol{f}_p \leq \boldsymbol{v}_p, \\ & -\boldsymbol{v}_p \leq \boldsymbol{0}. \end{aligned} \tag{8.36}$$

To ensure prioritization, we rewrite (8.36) to search for a solution in the nullspace of the higher priority tasks. We do this by writing $\boldsymbol{\xi}_{c,p} = \boldsymbol{\xi}^*_{c,p-1} + \boldsymbol{Z}_{p-1}\boldsymbol{z}_p$, where $\boldsymbol{Z}_{p-1}$ is a basis of the nullspace of the stack of the equality constraints from priority 1 to $p-1$, and $\boldsymbol{\xi}^*_{c,p-1}$ is the previously found solution vector. To ensure that higher priority inequality constraints are

fulfilled, we stack them together with previously computed slack variables $\boldsymbol{v}_1^*, \cdots, \boldsymbol{v}_{p-1}^*$. Hence, (8.36) is rewritten as

$$\begin{aligned} \min_{\boldsymbol{z}_p, \boldsymbol{v}_p} \quad & \|\boldsymbol{A}_p(\boldsymbol{\xi}_{c,p-1}^* + \boldsymbol{Z}_{p-1}\boldsymbol{z}_p) - \boldsymbol{b}_p\|_{\boldsymbol{W}_{eq,p}}^2 + \|\boldsymbol{v}_p\|_{\boldsymbol{W}_{ineq,p}}^2 \\ \text{s. t.} \quad & \boldsymbol{D}_p(\boldsymbol{\xi}_{c,p-1}^* + \boldsymbol{Z}_{p-1}\boldsymbol{z}_p) - \boldsymbol{f}_p \leq \boldsymbol{v}_p, \\ & \boldsymbol{D}_{p-1}(\boldsymbol{\xi}_{c,p-1}^* + \boldsymbol{Z}_{p-1}\boldsymbol{z}_p) - \boldsymbol{f}_{p-1} \leq \boldsymbol{v}_{p-1}^*, \\ & \vdots \\ & \boldsymbol{D}_1(\boldsymbol{\xi}_{c,p-1}^* + \boldsymbol{Z}_{p-1}\boldsymbol{z}_p) - \boldsymbol{f}_1 \leq \boldsymbol{v}_1^*, \\ & -\boldsymbol{v}_p \leq \boldsymbol{0}. \end{aligned} \tag{8.37}$$

The optimal solution to (8.37) are the optimal vectors $\boldsymbol{z}_p^*$ and $\boldsymbol{v}_p^*$, which we use to update the optimal solution $\boldsymbol{\xi}_{c,p}^*$ by writing $\boldsymbol{\xi}_{c,p}^* = \boldsymbol{\xi}_{c,p-1}^* + \boldsymbol{Z}_{p-1}\boldsymbol{z}_p^*$. The latter can be rewritten in recursive form as

$$\boldsymbol{\xi}_c^* = \sum_{p=1}^{n_p} \boldsymbol{Z}_{p-1}\boldsymbol{z}_p^*, \tag{8.38}$$

with $\boldsymbol{Z}_0 = \mathbb{I}$ and n_p the number of priority levels in the task hierarchy.

8.6.2 *Prioritized Tasks*

In this section, we describe the hierarchy of tasks which are used throughout our experiments. The table 8.3 lists all the tasks used, together with their associated priority (priority 1 is the highest).

8.6.2.1 *Equations of Motion*

Exploiting the decomposition introduced by the selection matrix $\boldsymbol{S}^T$ in eq. (8.4), we write a task that constrains the solution vector $\boldsymbol{\xi}_c$ to be consistent with the system dynamics as

$$\begin{bmatrix} \boldsymbol{M}_b(\boldsymbol{q}) & -\boldsymbol{J}_{s,b}^T(\boldsymbol{q}) \end{bmatrix} \boldsymbol{\xi}_c = -\boldsymbol{h}_b(\boldsymbol{q}, \boldsymbol{u}), \tag{8.39}$$

where the subscript b indicates that only the rows related to the floating-base (i.e., the first six) are taken from $\boldsymbol{M}$, $\boldsymbol{J}_s^T$ and $\boldsymbol{h}$. This task constrains the generalized accelerations $\dot{\boldsymbol{u}}$ to be consistent with the reaction forces $\boldsymbol{\lambda}$ according to the full system dynamics. For this reason, we set this task as the one with the highest priority.

8.6.2.2 *Torque Limits*

To avoid exceeding the maximum or minimum actuation torque, we set up an inequality constraint task as

$$\begin{bmatrix} \boldsymbol{M}_j(\boldsymbol{q}) & -\boldsymbol{J}_{s,j}^T(\boldsymbol{q}) \\ -\boldsymbol{M}_j(\boldsymbol{q}) & +\boldsymbol{J}_{s,j}^T(\boldsymbol{q}) \end{bmatrix} \boldsymbol{\xi}_c \leq \begin{bmatrix} \boldsymbol{\tau}_{max} - \boldsymbol{h}_j(\boldsymbol{q}, \boldsymbol{u}) \\ -\boldsymbol{\tau}_{min} + \boldsymbol{h}_j(\boldsymbol{q}, \boldsymbol{u}) \end{bmatrix}, \tag{8.40}$$

where the subscript j indicates that only the joint-space related rows (i.e., the lower twelve for a quadruped and the lower 18 for a quadruped with a six DOF arm) are taken from $\boldsymbol{M}$, $\boldsymbol{J}_s$ and $\boldsymbol{h}$. The values of maximum torque $\boldsymbol{\tau}_{max}$ and minimum torque $\boldsymbol{\tau}_{min}$ are obtained from the specifications of the actuators.

8.6.2.3 *Contact force limits*

To avoid slipping, the leg reaction forces computed by the controller should not violate the friction limits imposed by the interaction with the environment. Assuming a Coulomb friction model, the limits constrain the ratio of the normal and tangential component of the forces to be smaller than the friction coefficient μ, which is assumed to be known and constant during locomotion. As typically done in the control of legged robots, we linearize the nonlinear cone constraint by replacing it with a friction pyramid aligned with the heading ${}_I\boldsymbol{h}$ and lateral ${}_I\boldsymbol{l}$ unit vectors that lie on the contact surface. The constraints for the k-th foot reaction contact force $\boldsymbol{\lambda}_k$ are written as

$$\begin{aligned} ({}_I\boldsymbol{h} - {}_I\boldsymbol{n}\mu)^T {}_I\boldsymbol{\lambda}_k \leq \boldsymbol{0}, \\ -({}_I\boldsymbol{h} + {}_I\boldsymbol{n}\mu)^T {}_I\boldsymbol{\lambda}_k \leq \boldsymbol{0}, \end{aligned} \tag{8.41}$$

for the heading direction, and as

$$\begin{aligned} ({}_I\boldsymbol{l} - {}_I\boldsymbol{n}\mu)^T {}_I\boldsymbol{\lambda}_k \leq \boldsymbol{0}, \\ -({}_I\boldsymbol{l} + {}_I\boldsymbol{n}\mu)^T {}_I\boldsymbol{\lambda}_k \leq \boldsymbol{0}, \end{aligned} \tag{8.42}$$

for the lateral one.

Since the feet can only push on the ground, positive reaction forces in the direction of the ground normal ${}_I\boldsymbol{n}$ must be ensured. Hence, we add the inequality

$$-{}_I\boldsymbol{n}^T {}_I\boldsymbol{\lambda}_k \leq \boldsymbol{0}. \tag{8.43}$$

8.6.2.4 *No Motion at the Contact Points*

End-effectors which are supporting locomotion by being in contact with the environment should not exhibit any motion. Thus, we constrain their motion by writing

$$\begin{bmatrix} \boldsymbol{J}_s & \boldsymbol{0} \end{bmatrix} \boldsymbol{\xi}_c = -\dot{\boldsymbol{J}}_s \boldsymbol{u}. \tag{8.44}$$

8.6.2.5 *Motion tracking: center of mass, torso orientation, and end-effector motion*

Motion tracking is obtained by defining operational space motion tasks that set constraints on the generalized accelerations $\dot{u}$. We rewrite the task as a function of the solution vector $\boldsymbol{\xi}_c$ as

$$\begin{bmatrix} \boldsymbol{J}(\boldsymbol{q}) & \boldsymbol{0} \end{bmatrix} \boldsymbol{\xi}_c = \ddot{\boldsymbol{x}}_{ref} - \dot{\boldsymbol{J}}(\boldsymbol{q})\boldsymbol{u}. \tag{8.45}$$

The generated motions described in Section 8.4 are tracked by introducing tasks in the form of (8.45) in the task hierarchy. The motion tracking tasks are defined with respect to the control frame C, which has the same origin as the inertial frame I but is oriented along the local estimate of the terrain. To track COM motions defined as position ${}_C\boldsymbol{r}^d_{I,COM}$, velocity ${}_C\dot{\boldsymbol{r}}^d_{COM}$, and acceleration ${}_C\ddot{\boldsymbol{r}}^d_{COM}$, we write

$$\begin{aligned} &\begin{bmatrix} {}_C\boldsymbol{J}_{COM}(\boldsymbol{q}) & \boldsymbol{0} \end{bmatrix} \boldsymbol{\xi}_c = \\ &\left({}_C\ddot{\boldsymbol{r}}^d_{COM} + \boldsymbol{K}_{D,COM}({}_C\dot{\boldsymbol{r}}^d_{COM} - {}_C\dot{\boldsymbol{r}}_{COM} \right) \\ &+ \boldsymbol{K}_{P,COM}({}_C\boldsymbol{r}^d_{I,COM} - {}_C\boldsymbol{r}_{I,COM})) - {}_C\dot{\boldsymbol{J}}_{COM}(\boldsymbol{q})\boldsymbol{u}. \end{aligned} \tag{8.46}$$

The Jacobian ${}_C\boldsymbol{J}_{COM}$ is defined as the weighted sum of the translational Jacobians ${}_C\boldsymbol{J}_{P,b}$ of the COM for each body $\hat{\mathcal{B}}$, which is expressed as

$${}_C\boldsymbol{J}_{COM} = \frac{\sum\limits_{b \in \hat{\mathcal{B}}} m_b \cdot {}_C\boldsymbol{J}_{P,b}}{\sum\limits_{b \in \hat{\mathcal{B}}} m_b}, \tag{8.47}$$

where m_b is the mass of body b.

Orientation references for the torso (defined as a set of a desired configuration $\boldsymbol{q}^d_{IB}$, a desired angular velocity ${}_I\boldsymbol{\omega}^d_{IB}$, and a desired angular acceler-

ation ${}_I\dot{\boldsymbol{\omega}}^d_{IB}$) obtained from the motion planner are tracked according to the following formula:

$$\begin{aligned}&\begin{bmatrix}{}_C\boldsymbol{J}_{R,B}(\boldsymbol{q}) & \boldsymbol{0}\end{bmatrix}\boldsymbol{\xi}_c = \\ &{}_C\dot{\boldsymbol{\omega}}^d_{IB} + \boldsymbol{K}_{D,B}({}_C\boldsymbol{\omega}^d_{IB} - {}_C\boldsymbol{\omega}_{IB}) \\ &+ \boldsymbol{K}_{P,B}(\boldsymbol{q}^d_{CB} \boxminus \boldsymbol{q}_{CB}) - {}_C\dot{\boldsymbol{J}}_{R,B}(\boldsymbol{q})\boldsymbol{u}.\end{aligned} \tag{8.48}$$

The $\boxminus$ operator [53] is defined as

$$\begin{aligned}\boxminus : SO(3) \times SO(3) &\to \mathbb{R}^3, \\ \boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2 &\mapsto \log(\boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_2^{-1}),\end{aligned} \tag{8.49}$$

where $\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2 \in SO(3)$ are generic $3D$ orientations, and $\circ$ is an orientation concatenation operator. The mapping returns an Euler vector which represents an orientation error. The rotational Jacobian ${}_I\boldsymbol{J}_{R,B}$ projects generalized velocities $\boldsymbol{u}$ to base angular velocities ${}_I\boldsymbol{\omega}_{IB}$.

To track the translational reference position ${}_C\boldsymbol{r}_{IE_k}$, velocity ${}_C\dot{\boldsymbol{r}}_{E_k}$, and acceleration ${}_C\ddot{\boldsymbol{r}}_{E_k}$ of the k-th limb's end-effector $\mathcal{E}$, we introduce a task

$$\begin{aligned}&\begin{bmatrix}{}_C\boldsymbol{J}_{P,E_k}(\boldsymbol{q}) & \boldsymbol{0}\end{bmatrix}\boldsymbol{\xi}_c = \\ &{}_C\ddot{\boldsymbol{r}}^d_{E_k} + \boldsymbol{K}_{D,E}({}_C\dot{\boldsymbol{r}}^d_{E_k} - {}_C\dot{\boldsymbol{r}}_{E_k}) \\ &+ \boldsymbol{K}_{P,E}({}_C\boldsymbol{r}^d_{IE_k} - {}_C\boldsymbol{r}_{IE_k}) - {}_C\dot{\boldsymbol{J}}_{P,E_k}(\boldsymbol{q},\boldsymbol{u})\boldsymbol{u},\end{aligned} \tag{8.50}$$

where ${}_C\boldsymbol{J}_{P,E_k}$ is the translational Jacobian of the end-effector of the k-th limb and ${}_C\dot{\boldsymbol{J}}_{P,E_k}(\boldsymbol{q},\boldsymbol{u})$ its time derivative.

For the three-DOF legs, only translational motion of the end-effector is tracked. When the six-DOF arm is added to the robot, the arm's end-effector motion tracking task is augmented with a rotational motion tracking task (tracking the gripper's orientation $\boldsymbol{q}_{IE}$, angular velocity ${}_I\boldsymbol{\omega}_{IE}$, and angular acceleration ${}_I\dot{\boldsymbol{\omega}}_{IE}$). The augmented task is written as

$$\begin{aligned}&\begin{bmatrix}{}_C\boldsymbol{J}_{R,E}(\boldsymbol{q}) & \boldsymbol{0}\end{bmatrix}\boldsymbol{\xi}_c = \\ &{}_C\dot{\boldsymbol{\omega}}^d_{IE} + \boldsymbol{K}_{D,E}({}_C\boldsymbol{\omega}^d_{IE} - {}_C\boldsymbol{\omega}_{IE}) \\ &+ \boldsymbol{K}_{P,E}(\boldsymbol{q}^d_{CE} \boxminus \boldsymbol{q}_{CE}) - {}_C\dot{\boldsymbol{J}}_{R,E}(\boldsymbol{q},\boldsymbol{u})\boldsymbol{u}.\end{aligned} \tag{8.51}$$

8.6.2.6 *Adaptation of Torso Orientation*

When ANYmal is equipped with the arm, a common task is to have the gripper reaching for an object to grasp. To increase the reachable

workspace of the gripper, it is possible to exploit the kinematic redundancy of the robot which is introduced by the floating base. This can be done by adapting the torso's orientation according to the reaching direction of the arm, such that the motion of the torso contributes to that of the gripper. To compute the desired direction of orientation adaptation of the torso in the base frame (${}_B\boldsymbol{\omega}^d_{IB}$), we first compute the desired linear velocity (${}_B\boldsymbol{v}^d_{BS}$) of the "shoulder" which is defined as the mounting point of the arm to the base. To have the torso motion extend the kinematic workspace of the gripper, the desired shoulder velocity is defined to be in the direction of the current position of the gripper and is written as

$$ {}_B\boldsymbol{v}^d_{BS} = k_p \; {}_B\hat{\boldsymbol{r}}_{SG}, \tag{8.52} $$

where k_p is a scalar which multiplies the unit vector that points in the direction from the shoulder to the gripper. When expressing the rotations of the torso in the base frame, rotations around the torso's x-axis and z-axis produce instantaneous velocities of the shoulder in the identical direction. For this reason, we first map the desired shoulder velocity (${}_B\boldsymbol{v}^d_{BS}$) to desired torso velocities around the torso's x-axis and y-axis (${}_B\boldsymbol{\omega}^d_{xy,IB}$) as

$$ {}_B\boldsymbol{\omega}^d_{xy,IB} = \boldsymbol{S}_{xy}({}_B\boldsymbol{r}_{BS})^\dagger \, {}_B\boldsymbol{v}^d_{BS}. \tag{8.53} $$

Here $\boldsymbol{S}_{xy}({}_B\boldsymbol{r}_{BS})^\dagger$ denotes the top two rows of the pseudo-inverse of the skew-symmetric matrix which is computed such that $\boldsymbol{S}({}_B\boldsymbol{r}_{BS}) \, {}_B\boldsymbol{v}^d_{BS} = {}_B\boldsymbol{r}_{BS} \times_B \boldsymbol{v}^d_{BS}$, with $\boldsymbol{r}_{BS}$ the position of the mounting point of the arm w.r.t. the base. Subsequently ${}_B\boldsymbol{\omega}^d_{z,IB}$ can be set proportional to ${}_B\boldsymbol{\omega}^d_{x,IB}$ because their relation to the direction of the instantaneous velocity of the shoulder is identical. The resulting control task is formulated as

$$ \begin{aligned} &\begin{bmatrix} {}_B\boldsymbol{J}_{R,B}(\boldsymbol{q}) & \boldsymbol{0}_{3\times n_c} \end{bmatrix} \boldsymbol{\xi}_c = \\ &k_d({}_B\boldsymbol{\omega}^d_{xyz,IB} - {}_B\,\boldsymbol{\omega}_{xyz,IB}) - {}_B\,\dot{\boldsymbol{J}}_{R,B}(\boldsymbol{q},\boldsymbol{u})\boldsymbol{u}. \end{aligned} \tag{8.54} $$

This control task is added on the same priority level as the regular torso orientation tracking task. The two control tasks conflict with each other, resulting in a behavior where the torso orientation changes to accommodate the gripper's reaching motion, but does not diverge too far from the reference orientation given for the regular torso orientation tracking task. Finally, it can be desirable to only smoothly introduce this task when the arm has extended beyond a specified threshold by setting the weight of

the control task proportional to the amount by which this threshold is exceeded. This results in a behavior where the torso orientation is adapted to aid the robot's reaching motion only when this is necessary.

8.6.2.7 *Arm contact force tracking*

Whenever the arm's gripper is in contact with the environment, the robot controls the contact wrench (which includes both contact force and contact torque) between the gripper and the environment. Because the contact forces appear directly in the optimization vector ξ_c, we design a contact force task as

$$\begin{bmatrix} \mathbf{0} & S_G \end{bmatrix} \xi_c = \lambda_G^d, \tag{8.55}$$

where S_G is the selection matrix selecting the entries of ξ_c corresponding to the gripper's contact forces, and λ_G^d is a six-dimensional vector of desired reaction forces and torques.

8.6.2.8 *Contact force minimization*

To reduce the risk of slippage, we minimize the components of the reaction forces which are perpendicular to the normal to the terrain surface, while equally distributing the contact force components that are parallel to the surface normal. We do as

$$\begin{bmatrix} \mathbf{0} & I \end{bmatrix} \xi_c = \mathbf{0}. \tag{8.56}$$

Moreover, this task resolves the redundancy issue associated with generation of internal forces.

8.6.3 *Actuator Torque References*

Solving the task hierarchy produces a vector of optimal accelerations $\dot{u}^*$ for all the system's DOF and contact forces λ^* for all the end-effectors that are in contact with the environment. These quantities are used to compute the desired actuator torque commands τ^d as

$$\tau^d = M_j(q)\dot{u}^* + h_j(q,u) - J_{s,j}^T(q)\lambda^*, \tag{8.57}$$

where $M_j(q)$, $h_j(q,u)$, and $J_{s,j}(q)$ are the joint space related rows of the mass matrix, nonlinear terms, and contact Jacobian, respectively. When using pure torque commands for a foot that is swinging to a new contact position, motion tracking can degrade due to imperfect modeling and low

TABLE 8.3: The list of prioritized tasks used to control the robot and track the motion plans. Priority 1 is the highest. The tasks in blue and marked with * are only added when ANYmal is equipped with a six-DOF arm. The task between brackets is optional.

Priority	Task
1	Floating base equations of motion
	Torque limits
	Friction cone and contact force modulation
2	No motion at the contact points
3	Center of mass linear motion
	Torso angular motion
	Swing leg motion tracking
	Arm motion tracking*
	Torso orientation adaptation*
	(Arm contact force tracking*)
4	Contact force minimization

mass of the legs. To address this issue, we add joint space motion references to the desired torques. The final torque reference command $\boldsymbol{\tau}^{ref}$ sent to the actuators is computed as

$$\boldsymbol{\tau}^{ref} = \boldsymbol{\tau}^{d} + \boldsymbol{K}_{P,j}(\boldsymbol{q}_j^d - \boldsymbol{q}_j) + \boldsymbol{K}_{D,j}(\dot{\boldsymbol{q}}_j^d - \dot{\boldsymbol{q}}_j), \tag{8.58}$$

where $\boldsymbol{q}_j^d$ and $\boldsymbol{q}_j$ are desired and measured joint angles respectively, and $\dot{\boldsymbol{q}}_j^d$ and $\dot{\boldsymbol{q}}_j$ are desired and measured joint velocities respectively. The matrices $\boldsymbol{K}_{P,j}, \boldsymbol{K}_{D,j}$ are positive-definite diagonal control gain matrices that define a PD controller in the joint space.

8.7 EXPERIMENTS

We have extensively validated our planning and control framework on ANYmal [56], a torque-controlled quadrupedal robot that we have used to perform locomotion experiments. For manipulation tasks, we equipped ANYmal with a torque-controlled six-DOF Kinova Jaco^2 robotic arm. The software running on the robot is executed in a 400 Hz loop. The latter includes obtaining readings from the actuators, estimating the state of the system [90], generating control signals, and sending control references

back to the drives. The motion optimization modules presented in the previous sections are executed in separate dedicated parallel loops. The frequency at which they run depends on several factors, including the gait timings and the weights chosen in the cost functions.

ANYmal is equipped with an on-board computer (Intel i7-7600U, 2.7 - 3.5GHz, dual-core 64-bit) dedicated to running the entire control framework. To model kinematics and dynamics, we use a custom version of the open-source Rigid Body Dynamics Library (RBDL, [58]), a C++ implementation of the Featherstone algorithms [59].

All the QP problems described in this article are numerically solved using the QuadProg++ library [60], an off-the-shelf open-source QP solver based on the Goldfarb-Idnani active-set method [61]. The nonlinear optimization problem described in Section 8.4 is solved using our custom SQP solver.

8.7.1 *Locomotion: Walking Gaits*

The motion planning framework described in Section 8.4 is capable of generating motions for a wide variety of gaits based on a list of user-defined contact schedules that are loaded at software initialization. Our tests range from slower walking gaits, to faster and more demanding running ones. In this Section, we describe the results obtained by showing the performance of the framework both on the flat ground of the laboratory, as well as outdoors in real-world environment scenarios (see Figure 8.10).

A summary of the computation time for several gaits is depicted in Figure 8.9. For each gait, we define which DOF to be included in the optimization of the COM frame trajectories. For instance, we optimize only for the x and y coordinates of the COM for walking and trotting. For running trot, which requires vertical accelerations to jump during full flight phases, we additionally optimize for the z component of the COM. For more demanding demanding gaits such as bounding and gallop, we include the pitch angular motion in the optimization. The versatility of our implementation allows to keep optimization times low whenever possible. In fact, the number of DOF included in the optimization strongly influences the computational efficiency of the planner. When optimizing for x and y translational coordinates only, the ZMP constraints become linear w.r.t. the planner's decision variables, allowing us to compute the trajectories by solving a QP problem. When the z motion is optimized as well (which, as mentioned, is the case for gaits that exhibit full flight phases,

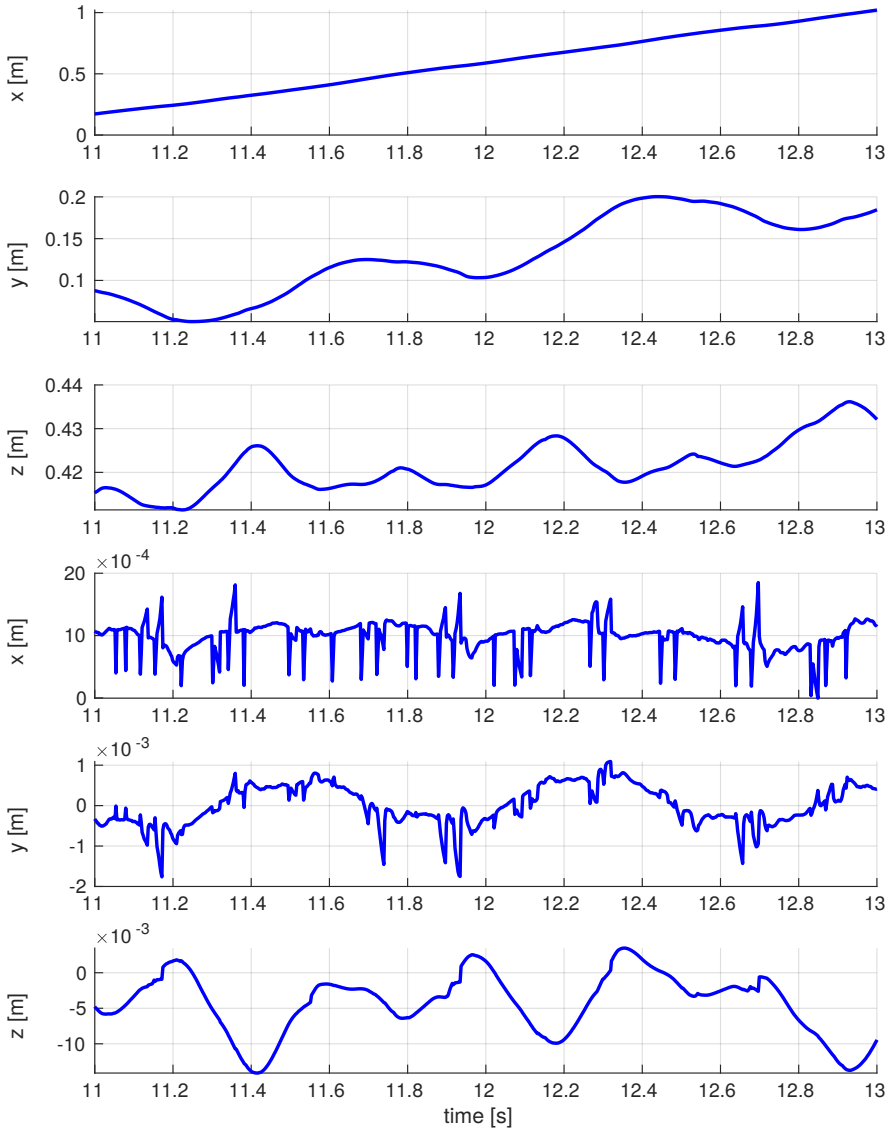


FIGURE 8.7: The motion of the base during the dynamic walk. The top three rows depict the measured center of mass position, while the bottom three ones show the tracking error.

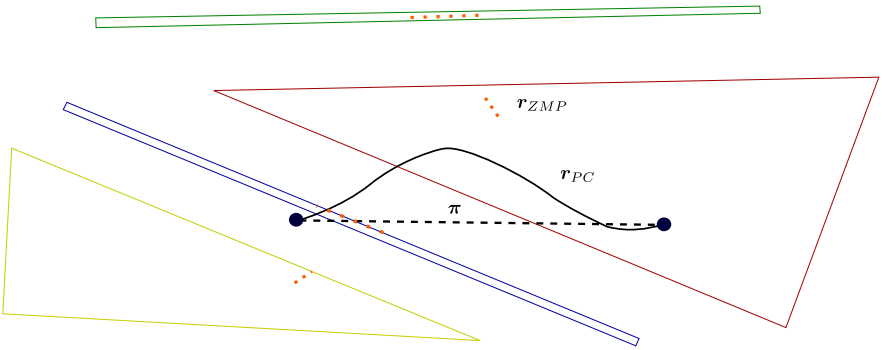


FIGURE 8.8: The optimized motion for a dynamic walk. The gait pattern switches between lines and triangles, in which the ZMP is constrained to be located for balance.

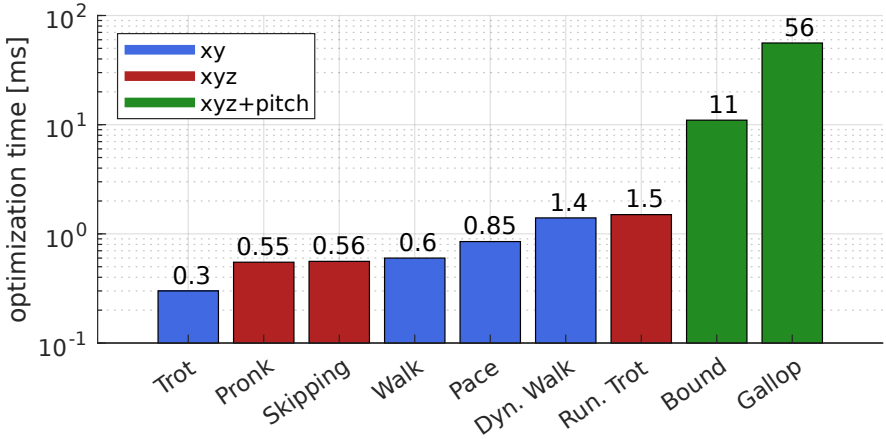


FIGURE 8.9: Optimization times for different gaits. The colors indicate which DOF are optimized for each gait.

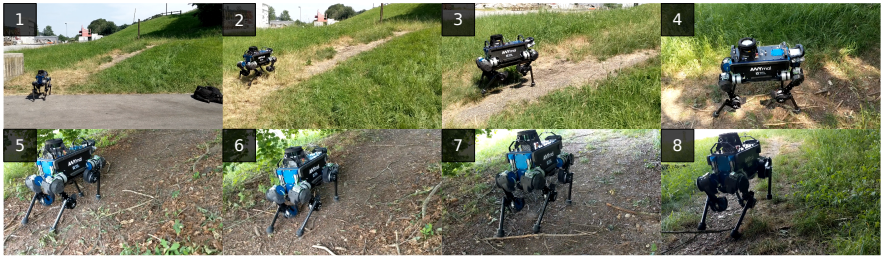
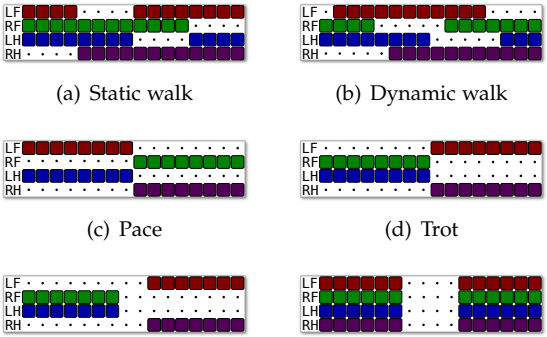


FIGURE 8.10: The sequence depicts ANYmal blindly trotting up a steep hill. The fast online replanning allows to negotiate the rough terrain and to react to walking on sticks and stones, as can be seen in frames 7 and 8. Throughout this experiment, a trotting gait is used. The robot is commanded by the user to trot forward, while the motion planning framework locally generates the motion plans tracked by the whole-body controller.



(a) Static walk (b) Dynamic walk

(c) Pace (d) Trot

(e) Running trot (f) Pronk

FIGURE 8.11: The gait patterns tested in our experiments. The colored squares represent contact, while the dots represent swing. When only one leg is swinging (a), the gait is a static walk. When two successive swing phase overlap (b), the static walk becomes a dynamic walk. When they fully overlap, the gait is a trot (d, the two diagonal legs are swinging) or a pace (c, the two lateral legs are swinging). A trot becomes a running trot (e) when there is a full flight phase. Pronking (f) is when there is an alternation between a full flight phase and a full stance phase.

TABLE 8.4: A summary of the experimental results obtained for a selection of gaits. The reported results are obtained by command the robot to walk in the forward direction at 0.25 m/s. The minimum and maximum torques τ (measured in [N m]) and joint velocities $\dot{q}_j$ (measured in rad/s) are reported for the three joints (HAA, Hip Abdutcion-Adduction; HFE, Hip Flexion-Extension; KFE, Knee Flexion-Extension) of the left fore (LF) leg. The table also shows the stride duration τ_s, as well as the time τ_{opt} required by the optimizer to generate a motion plan. We recall that the optimization horizon τ_h is equal to the stride duration.

Gait	**LF - HAA**	**LF - HFE**	**LF - KFE**	τ_s	τ_{opt}
Walk	$\tau \in [-9.98, 6.33]$ $\dot{q}_j \in [-1.26, 1.13]$	$\tau \in [-20.27, 9.25]$ $\dot{q}_j \in [-6.10, 2.85]$	$\tau \in [-0.80, 27.48]$ $\dot{q}_j \in [-5.84, 3.90]$	1.7 s	0.6 ms
Dynamic Walk	$\tau \in [-3.88, 22.20]$ $\dot{q}_j \in [-0.81, 1.31]$	$\tau \in [-23.04, 11.03]$ $\dot{q}_j \in [-5.00, 2.51]$	$\tau \in [-0.79, 32.77]$ $\dot{q}_j \in [-4.30, 4.58]$	0.9 s	1.4 ms
Trot	$\tau \in [-15.49, 9.57]$ $\dot{q}_j \in [-0.56, 0.83]$	$\tau \in [-12.06, 14.64]$ $\dot{q}_j \in [-5.62, 2.90]$	$\tau \in [-0.81, 36.19]$ $\dot{q}_j \in [-4.84, 5.22]$	0.85 s	0.3 ms
Running Trot	$\tau \in [-16.07, 11.27]$ $\dot{q}_j \in [-0.59, 0.87]$	$\tau \in [-40.00, 12.18]$ $\dot{q}_j \in [-4.83, 3.61]$	$\tau \in [-1.02, 40.00]$ $\dot{q}_j \in [-6.07, 6.23]$	0.65 s	1.5 ms
Pace	$\tau \in [-13.90, 35.87]$ $\dot{q}_j \in [-1.09, 0.98]$	$\tau \in [-14.23, 13.28]$ $\dot{q}_j \in [-6.50, 2.91]$	$\tau \in [-0.95, 32.24]$ $\dot{q}_j \in [-5.28, 5.29]$	1.0 s	0.85 ms

e.g., running trot or pronk), the ZMP constraints become nonlinear. In this case, we use an SQP approach to numerically solve the nonlinear optimization problem. The computation times, however, are in the same order of magnitude. The larger computation time of a dynamic walk w.r.t. a trot lies primarily in the number of support polygons in the optimization horizon, which depends on the number of contact changes in the gait pattern. For each contact change, a new support polygon is included, which in turn increases the total number of splines and, hence, the number of decision variables included in the optimization. Finally, including rotational DOF in the optimization (e.g., bounding and galloping) drastically increases the planner's computational effort. It is worth mentioning at this point that for each set of gaits that include the same number of DOF in the optimization share the same cost function used in the COM frame motion planner.

Table 8.4 shows data collected on the real hardware demonstrating the joint torque and joint velocity requirements for a selection of gaits. In the following, we describe the differences between the walking patterns.

A static walk is a gait which exhibits at maximum one swinging leg at any moment (see Figure 8.11(a)), i.e., the quadrupedal robot always has at least three legs in contact with the environment. The support polygons switch between a quadrilateral (if there are any full stance phases between two successive swings) and a triangle. It has been shown that this kind of gait is ideal for slow walking and careful selection of footholds [23]. While balance can be obtained by constraining the COM to lie in the support polygon [41], constraining the ZMP instead results in faster and smoother motion plans.

When two successive swing phases of a static walk partially overlap, the gait becomes a dynamic walking gait (see Figure 8.11(b)). The resulting support polygons alternate between triangles and lines, as depicted in Figure 8.8. Although the motion plan for this kind of gait resembles that of a static walking gait, it allows for faster locomotion speed. Figure 8.7 depicts the measured motion of a dynamic walk during forward locomotion, as well as the tracking errors.

A recent analysis of walking gaits for a quadrupedal robot [98] has showed that the trotting gait (see Figure 8.11(d)) to be the most efficient one in terms of cost of transport (COT) for a wide range of speeds below 1m/s. The gait is characterized by always having the two diagonal legs swinging at the same time, while the two legs in contact are supporting the whole weight of the robot (see Figure 8.12). To increase balance robustness, a brief full stance can be added between the contact switches. As

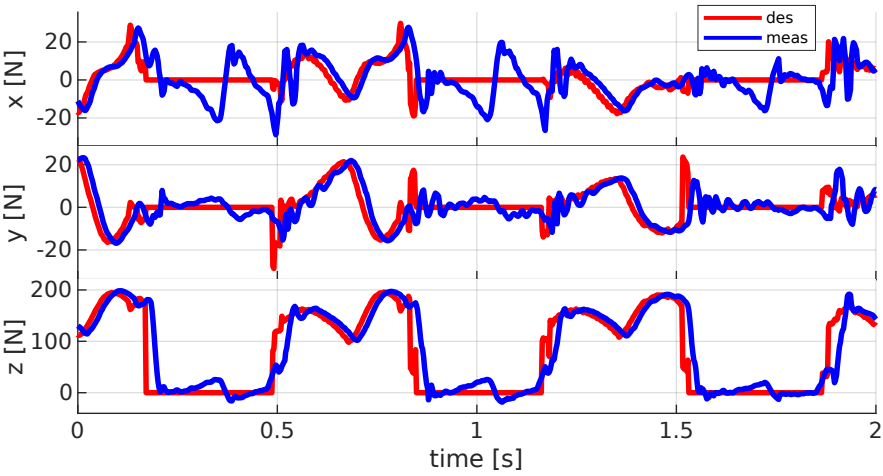


FIGURE 8.12: The desired and estimated reaction forces for the left fore leg while executing a trotting gait.

trotting typically does not produce large lateral shifts, we use it as the default gait for recovery against pushes. Moreover, we have used it for the experiments with the arm mounted on the quadruped.

When a full flight phase is added to the contact schedule of a trot, the gait is a running trot (see Figure 8.11(e)). This allows the robot to achieve higher running speeds. In our experiments, we have reached 1.2m/s in the heading direction. To produce full flight phases, the motion planner generates accelerations in the z direction to allow the robot to briefly jump (see Figure 8.13). Pacing has two lateral legs swinging together, unlike trotting which has two diagonal pairs swinging together (see Figure 8.11(c)). The support polygons alternate between lines connecting the two lateral feet in contact. To execute this motion, accelerations in the lateral direction must be produced to keep the ZMP on the lateral lines, while trying to keep the COM close to the middle of the footprint. Pronking is a locomotion behaviour that alternates a full stance with a full flight phase (see Figure 8.11(f)). The gait results in a sequence of jumps, which drive the robot according to the external reference twist. To further test the capabilities of our motion planner, we have implemented a galloping gait (see Figure 8.14). The optimizer produces motions for the full translational components of the COM, as well as pitch angles for the torso. The execution of this maneuver is depicted in Figure 8.15.

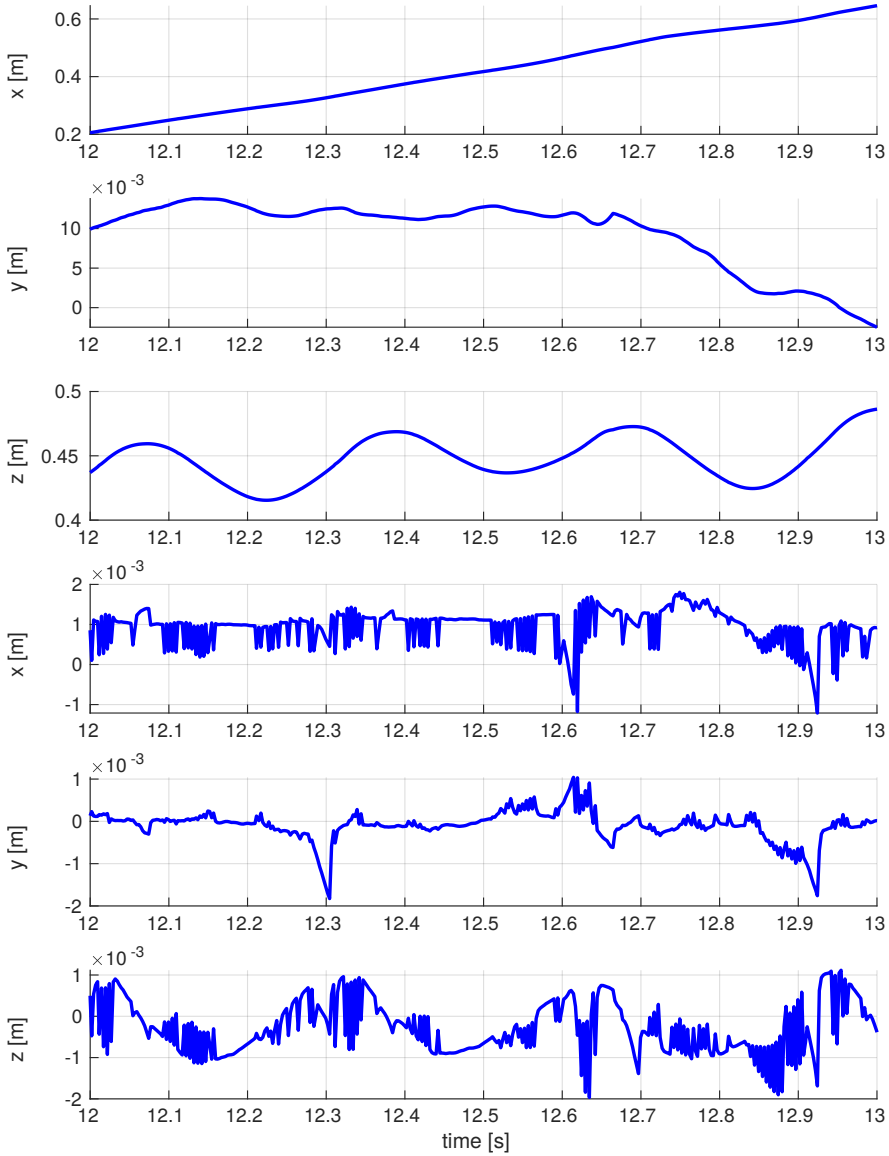


FIGURE 8.13: The motion of the base during a running trot. The top three rows depict the measured center of mass position, while the bottom three ones show the tracking error. The optimization produces accelerations in the z direction to let the robot jump when a full flight phase is planned.

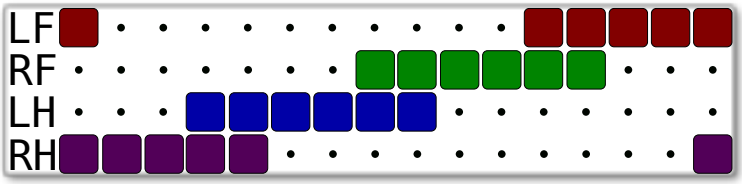


Figure 8.14: The contact schedule for a galloping gait, which alternates two contact feet with a single contact location.

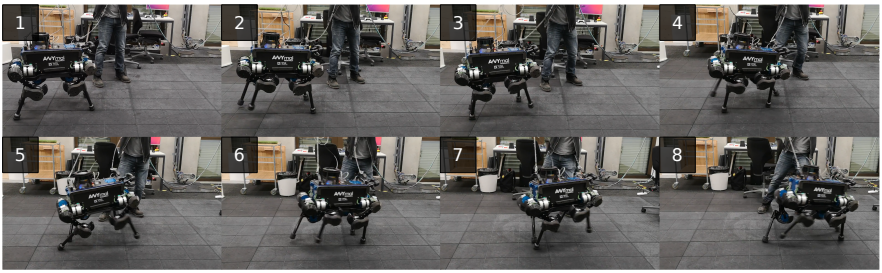


Figure 8.15: The sequence depicts a galloping gait. For this kind of gait, we setup the motion planner to generate motions for the COM and for the pitch of the torso.

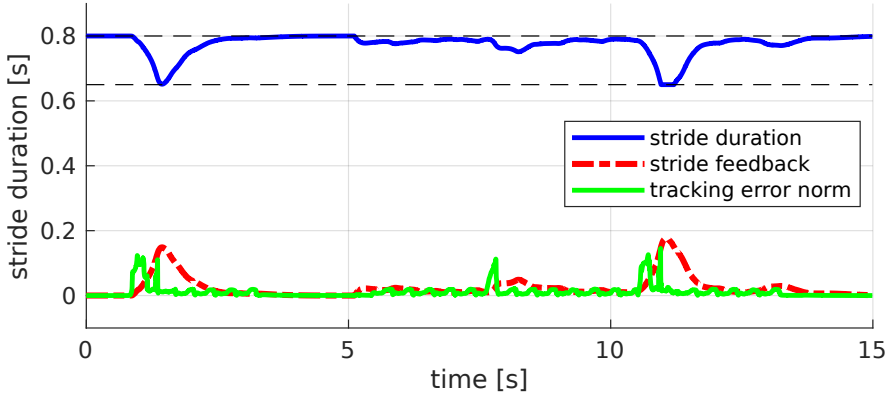


FIGURE 8.16: Velocity tracking errors are used to adapt the stride duration for a trotting gait. The dotted lines represent the minimum and maximum stride durations. When the velocity tracking error increases (e.g., when the robot is pushed), the stride duration decreases according to eq. (8.7). In turn, the stepping frequency increases.

8.7.2 *Reactive Behaviour*

To allow the robot to autonomously react to an external disturbance (e.g., a kick) while standing or to start walking when a reference velocity is given by the user, we have implemented a state machine that switches between standing and walking. The state machine decides to switch from standing to walking either when the norm of the commanded velocity is greater than a threshold, or when the position and velocity tracking errors are larger than a user defined value.

During locomotion, as discussed in Section 8.4, we adapt the stride duration τ_s based on a feedback term that is function of the linear and angular velocity tracking errors (see eq. (8.7)). As shown in Figure 8.16, τ_s automatically adapts to velocity tracking errors.

8.7.3 *Manipulation*

The WBC enables to control the motion of the gripper independently from the motion of the robot's COM. This allows using the same locomotion motion planner for the robot with and without an arm since the planner plans motions for the robot's COM. Figure 8.19 shows an experiment that demonstrates this capability. While the hand is controlled to remain at a

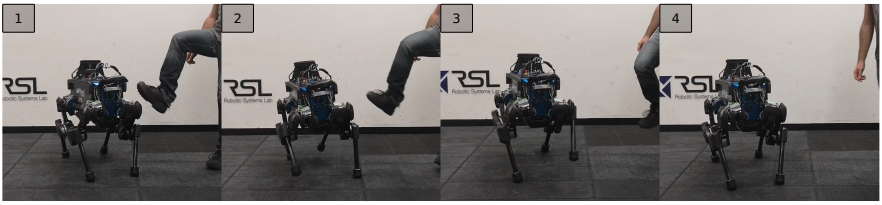


Figure 8.17: ANYmal can react to external disturbances (e.g., a push or a kick) by autonomously switching to the trotting gait.

Figure 8.18: ANYmal opening a door and walking through it. The door is opened by commanding a contact force between the gripper and the door handle, which depends on the estimated opening angle of the door. Since no motion reference is given for the gripper, it passively follows the motion of the door. A trotting gait is executed to walk through the door while it is being pushed open.

fixed pose w.r.t. the world, the robot's base is commanded to trot laterally. This dynamic gait requires precise tracking of the robot's COM reference motion as generated by the locomotion motion planner. The WBC successfully achieves this tracking. The effect of the torso orientation adaptation task (Section 8.6.2.6) can be seen when commanding the gripper to move away from the robot's torso. Figure 8.20 shows how the torso is rolled and yawed w.r.t. its neutral orientation in order to improve the workspace reachability of the arm when the gripper is moved away horizontally from the torso.

Instead of controlling the motion of the gripper, it is also possible to control the interaction forces between the gripper and the environment (8.55). To demonstrate a real-life application for this, we show ANYmal autonomously opening a door using its gripper, while trotting (see Figure 8.18).

Motions of the gripper towards (before opening) and away from (after opening) the door handle are executed via the motion tracking task described in equations (8.50) and (8.51) in the WBC task setup. For the actual opening of the door, however, no motion references are given. Instead, we provide references for the contact forces and torques between the gripper

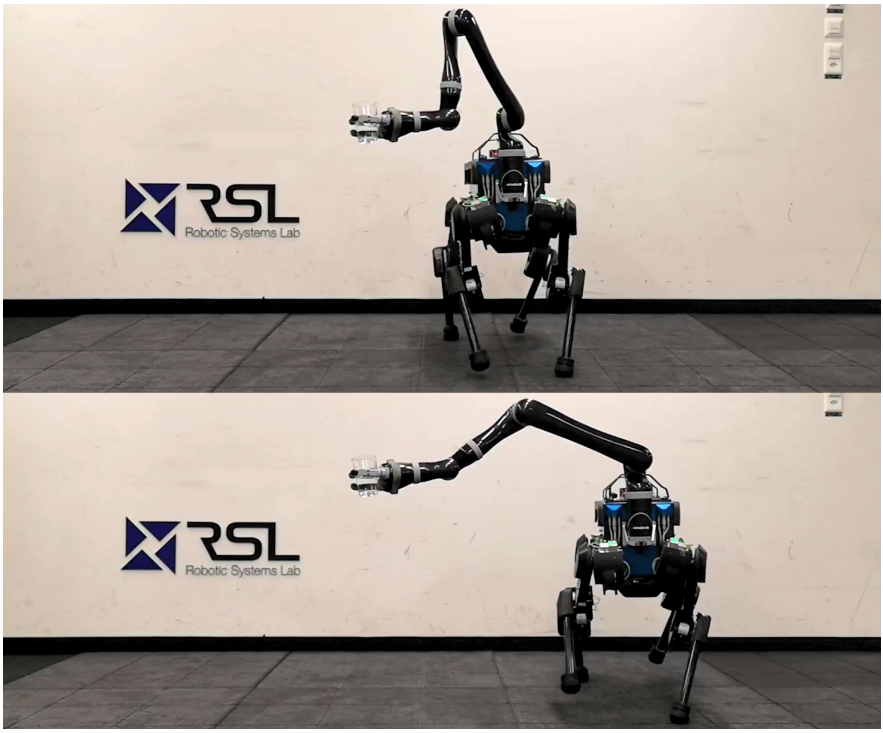


FIGURE 8.19: The gripper, holding a glass of water, is commanded to remain at a fixed pose. The motion of the robot's COM is controlled independently and follows the trajectory computed by the locomotion planner in order to perform a dynamic trotting gait.

Figure 8.20: The inclusion of the torso orientation adaptation control task results in a behaviour where the torso orientation deviates from its neutral reference orientation such that the reachability of the gripper is increased. In this case the roll and yaw of the torso enable the gripper to reach the bin.

and the door handle. When the end-effector is commanded to execute a wrench, the motion tracking task is reduced to the zero end-effector acceleration task described in eq. (8.44) to constrain the generalized accelerations of the arm. As a result, the direction of the commanded contact force needs only to be roughly similar to the motion direction corresponding to opening the door, while the actual motion of the gripper will passively follow the constrained motion of the door. When actively commanding contact forces and torques between the gripper and the environment, the inclusion of the motion and friction cone tasks in the WBC will implicitly guarantee the computation of a corresponding dynamically consistent contact force distribution over the feet.

8.8 conclusions

This article presents a motion planning and control framework for a multi-limbed torque-controlled robot. By using a reduced model, motions are efficiently evaluated online, allowing fast replanning that allows to cope with external disturbances or challenging terrain. A WBC, that exploits the full system dynamics, instantaneously tracks the reference motion by solving a hierarchy of prioritized tasks, such as dynamic feasibility, contact force limits, and torque limits. The framework has been evaluated on ANYmal, a torque-controlled quadrupedal robot. The modularity and generality of the framework allow to control an additional limb, a six-DOF

arm mounted on the robot, to execute manipulation tasks such as opening a door or grasping objects. To this end, we also prove the generality by applying the described framework to a wheeled quadrupedal robot [25].

Future work will focus on including perception in the planning and control pipeline. For locomotion, this addition will allow to take integrate information about the robot's surroundings into the motion optimization to allow safe selection of footholds and adaptation of the configuration of the system to rough terrain. The integration of vision information will also allow to increase the autonomy of the system for manipulation tasks, allowing the robot to autonomously detect, e.g., the location of door handles or objects to grasp. However, several challenging issues would be introduced, including higher computation times, unreliable sensor data, and higher complexity in guaranteeing balancing.

8.9 ACKNOWLEDGMENTS

This work was supported in part by the Swiss National Science Foundation (SNF) through the National Centre of Competence in Research Robotics (NCCR Robotics) and Digital Fabrication (NCCR dfab).
This work has been conducted as part of ANYmal Research, a community to advance legged robotics.

8.10 APPENDIX

This section describes the derivation of the gradients of the ZMP constraints introduced in Section 8.4.

8.10.1 *Tensor Notation*

Used in the computation of the gradient of the ZMP constraint, tensors are defined in this section.

8.10.1.1 *Tensor Definition*

A k-dimensional tensor $\boldsymbol{T} \in \mathbb{R}^{d_1 \times \cdots \times d_k}$ is the generalization of a 2-dimensional matrix and has $d_1 \cdot d_2 \ldots d_k$ coefficients denoted as $\boldsymbol{T}_{i,j,\ldots,m}$.

8.10.1.2 *Tensor Multiplication*

Given two tensors $\boldsymbol{P} \in \mathbb{R}^{d_1 \times \cdots \times d_j}$ and $\boldsymbol{Q} \in \mathbb{R}^{d_j \times \cdots \times d_n}$. The tensor $\boldsymbol{T}$ resulting from the tensor multiplication of $\boldsymbol{P}$ and $\boldsymbol{Q}$ along dimension j is defined as

$$\boldsymbol{T} := \boldsymbol{P} \times_{d_j} \boldsymbol{Q} \in \mathbb{R}^{d_1 \cdots \times d_{j-1} \times d_{j+1} \cdots \times d_n}$$

$$\boldsymbol{T}_{i_1,\ldots i_{j-1},k_{j+1},\ldots k_n} := \sum_{l=0}^{d_j} \boldsymbol{P}_{i_1,\ldots i_{j-1},l} \cdot \boldsymbol{Q}_{l,k_{j+1},\ldots k_n}.$$

8.10.1.3 *Matrix Differentiation*

In this section we summarize the derivatives of matrices and vectors that involve tensor multiplication. The results are used to derive the Jacobians of the nonlinear ZMP constraints.

Consider the matrices and vector $\boldsymbol{A}(\boldsymbol{x}) \in \mathbb{R}^{d_1 \times d_2}$, $\boldsymbol{B} \in \mathbb{R}^{d_2 \times d_3}$, $\boldsymbol{x} \in \mathbb{R}^{d_4}$, and $\boldsymbol{b} \in \mathbb{R}^{d_2}$. The derivative of $\boldsymbol{A}(\boldsymbol{x})$ w.r.t. $\boldsymbol{x}$ is a third-order tensor defined as

$$\frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial \boldsymbol{x}} := \left\{ \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_1}, \quad \ldots, \quad \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_{d_4}} \right\} \in \mathbb{R}^{d_1 \times d_2 \times d_4}, \tag{8.59}$$

i.e., the i-th $d_1 \times d_2$ frontal slice of the tensor is $\partial \boldsymbol{A} / \partial x_i \in \mathbb{R}^{d_1 \times d_2}$.

The derivative of $\boldsymbol{A}(\boldsymbol{x})\boldsymbol{b}$ w.r.t. $\boldsymbol{x}$ is defined as

$$\begin{aligned} \frac{\partial (\boldsymbol{A}(\boldsymbol{x}) \cdot \boldsymbol{b})}{\partial \boldsymbol{x}} &= \left[\frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_1} \cdot \boldsymbol{b} \quad \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_2} \cdot \boldsymbol{b} \quad \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_3} \cdot \boldsymbol{b} \right] \\ &= \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial \boldsymbol{x}} \times_{d_2} \boldsymbol{b} \in \mathbb{R}^{d_1 \times d_4}. \end{aligned} \tag{8.60}$$

Similarly, the derivative of $\boldsymbol{A}(\boldsymbol{x}) \cdot \boldsymbol{B}$ w.r.t. $\boldsymbol{x}$ is a third-order tensor written as

$$\begin{aligned} \frac{\partial (\boldsymbol{A}(\boldsymbol{x}) \cdot \boldsymbol{B})}{\partial \boldsymbol{x}} &= \left\{ \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_1} \cdot \boldsymbol{B}, \ \cdots, \ \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_{d_4}} \cdot \boldsymbol{B} \right\} \\ &= \left\{ \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_1}, \ \cdots, \ \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial x_{d_4}} \right\} \times_{d_2} \boldsymbol{B} \\ &= \frac{\partial \boldsymbol{A}(\boldsymbol{x})}{\partial \boldsymbol{x}} \times_{d_2} \boldsymbol{B} \in \mathbb{R}^{d_1 \times d_3 \times d_4}. \end{aligned} \tag{8.61}$$

8.10.2 *Derivatives of the ZMP Constraints*

In this section we derive the gradients of the nonlinear ZMP constraints used in the COM frame motion optimization. In the following, we use the

skew-matrix operator $\boldsymbol{S}(\cdot) \in \mathbb{R}^{3\times3}$ to write $\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{S}(\boldsymbol{a})\boldsymbol{b}$. The operator has the following properties:

- $\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b} = -\boldsymbol{S}(\boldsymbol{b})\boldsymbol{a}$
- $\boldsymbol{S}^T(\boldsymbol{a}) = -\boldsymbol{S}(\boldsymbol{a})$

Each edge of a support polygon is written as $ax + by + c = 0$, where a, b and c are the coefficients of the line equation, where a and b define the normal to the line on the plane on which the support polygon lies. We collect the two parameters in vector $\boldsymbol{d} = \begin{bmatrix} a & b & 0 \end{bmatrix}^T$. The parameters x and y are the coordinates of the points belonging to the line. Referring to eq. (8.24), we use the equation of each edge to introduce an inequality constraint for the position of the ZMP w.r.t. the plan frame origin $\boldsymbol{r}_{PZ}$ that we write as

$$\boldsymbol{d}^T\boldsymbol{r}_{PZ} + c = \boldsymbol{d}^T \frac{\boldsymbol{n} \times \boldsymbol{\tau}_P^{gi}}{\boldsymbol{n}^T\boldsymbol{f}^{gi}} + c \geq 0. \tag{8.62}$$

Using the fact that $\boldsymbol{n}^T\boldsymbol{f}^{gi} \leq 0$, we multiply both sides of eq. (8.62) by $\boldsymbol{n}^T\boldsymbol{f}^{gi}$ to obtain

$$\begin{aligned} \boldsymbol{d}^T(\boldsymbol{n} \times \boldsymbol{\tau}_P^{gi}) + c\boldsymbol{n}^T\boldsymbol{f}^{gi} = \\ \boldsymbol{d}^T(\boldsymbol{n} \times \boldsymbol{r}_{PG} \times (\boldsymbol{f}^{gi} - \dot{\boldsymbol{l}})) + c\boldsymbol{n}^T\boldsymbol{f}^{gi} = \\ \underbrace{\boldsymbol{d}^T(\boldsymbol{S}(\boldsymbol{n})\boldsymbol{S}(\boldsymbol{r}_{PG})(\boldsymbol{f}^{gi} - \dot{\boldsymbol{l}}) + c\boldsymbol{n}^T\boldsymbol{f}^{gi}}_{h(\boldsymbol{r}_{PZ})} \leq 0. \end{aligned} \tag{8.63}$$

We can now use eq. (8.63) to compute the gradient of $h(\boldsymbol{r}_{PZ})$ w.r.t. the decision variables $\boldsymbol{r}_{PG}$, $\ddot{\boldsymbol{r}}_{PG}$, $\boldsymbol{\chi}_{PB}$, $\boldsymbol{\omega}_{PG}$, and $\dot{\boldsymbol{\omega}}_{PG}$. We first derive the derivatives w.r.t. to the linear motion, and then w.r.t. the angular motion. We remind here that the gravito-inertial force $\boldsymbol{f}^{gi}$ is a function of the position $\boldsymbol{r}_{PG}$ and acceleration $\ddot{\boldsymbol{r}}_{PG}$ of the COM. Recalling that in our optimization we approximate the system dynamics to the equations of motion of a frame attached at the COM and aligned with the base frame, the rate of change of angular momentum $\dot{\boldsymbol{l}}$ is a function of the orientation of the base w.r.t. the plan frame $\boldsymbol{\chi}_{PB}$, its rotational velocity $\dot{\boldsymbol{\chi}}_{PB}$, and its rotational acceleration $\ddot{\boldsymbol{\chi}}_{PB}$.

8.10.2.1 *Translational Derivatives*

We obtain the partial derivative $\partial h(\boldsymbol{r}_{PZ})/\partial \boldsymbol{r}_{PG}$ by writing

$$\frac{\partial h}{\partial \boldsymbol{r}_{PG}} = -\boldsymbol{d}^T\boldsymbol{S}(\boldsymbol{n})\boldsymbol{S}(\boldsymbol{f}^{gi}) \tag{8.64}$$

Transposing the last result and using the properties of the skew-matrix operator, we obtain

$$\begin{aligned}\boldsymbol{\nabla}_{\boldsymbol{r}_{PG}} h &= -\boldsymbol{S}(\boldsymbol{f}^{gi})\boldsymbol{S}(\boldsymbol{n})\boldsymbol{d} \\ &= \underbrace{\boldsymbol{S}\left(\boldsymbol{S}(\boldsymbol{n})\boldsymbol{d}\right)}_{\boldsymbol{\Gamma}} \boldsymbol{f}^{gi} = \boldsymbol{\Gamma}\boldsymbol{f}^{gi}\end{aligned} \tag{8.65}$$

Differentiating eq. (8.63) w.r.t. $\ddot{\boldsymbol{r}}_{PG}$ we obtain

$$\frac{\partial h}{\partial \ddot{\boldsymbol{r}}_{PG}} = -m\boldsymbol{d}^T\boldsymbol{S}(\boldsymbol{n})\boldsymbol{S}(\boldsymbol{r}_{PG}) - cm\boldsymbol{n}^T. \tag{8.66}$$

The gradient is then written as

$$\begin{aligned}\boldsymbol{\nabla}_{\ddot{\boldsymbol{r}}_{PG}} h &= -m\boldsymbol{S}(\boldsymbol{r}_{PG})\boldsymbol{S}(\boldsymbol{n})\boldsymbol{d} - cm\boldsymbol{n} \\ &= m\boldsymbol{S}(\boldsymbol{S}(\boldsymbol{n})\boldsymbol{d})\boldsymbol{r}_{PG} - cm\boldsymbol{n} \\ &= m\boldsymbol{\Gamma}\boldsymbol{r}_{PG} - cm\boldsymbol{n}.\end{aligned} \tag{8.67}$$

In summary, the gradient of $h(\boldsymbol{r}_{PZ})$ w.r.t. the linear position and acceleration of the COM, $\boldsymbol{\nabla}_{lin} h$, is computed as

$$\boldsymbol{\nabla}_{lin} h = \begin{bmatrix} \boldsymbol{\Gamma}\boldsymbol{f}^{gi} \\ m\boldsymbol{\Gamma}\boldsymbol{r}_{PG} - cm\boldsymbol{n} \end{bmatrix} \tag{8.68}$$

8.10.2.2 *Rate of Change of Angular Momentum*

The angular momentum $\boldsymbol{l}$ expressed in the plane frame P is expressed by $\boldsymbol{l} = \boldsymbol{\mathcal{J}}_B \cdot \boldsymbol{\omega}_{PB}$, with $\boldsymbol{\mathcal{J}}_B$ the inertia of the robot, including the contribution from all moving bodies, in a nominal standing joint configuration, expressed at the COM and in the plane frame. We assume, as in [99], that the angular momentum induced by the movement of the joints is negligible. Hence, we compute the inertia for a nominal joint configuration. The inertia expressed in the body coordinate frame ${}_B\boldsymbol{\mathcal{J}}_B$ is independent of time. The change of frame of the inertia tensor is taken into account by writing

$$\boldsymbol{l} = \boldsymbol{R}_{PB} \cdot {}_B\boldsymbol{\mathcal{J}}_B \cdot \boldsymbol{R}_{PB}^T \cdot {}_P\boldsymbol{\omega}_{PB}. \tag{8.69}$$

Using the fact that $\dot{\boldsymbol{R}}_{PB} = \boldsymbol{S}({}_P\boldsymbol{\omega}_{PB})\boldsymbol{R}_{PB}$, taking the time derivative of the last equation yields

$$\begin{aligned}\dot{\boldsymbol{l}} = &\boldsymbol{S}({}_P\boldsymbol{\omega}_{PB})\boldsymbol{R}_{PB} \cdot {}_B\boldsymbol{\mathcal{J}}_B \cdot \boldsymbol{R}_{PB}^T \cdot {}_P\boldsymbol{\omega}_{PB} \\ &+ \boldsymbol{R}_{PB} \cdot {}_B\boldsymbol{\mathcal{J}}_B \cdot \boldsymbol{R}_{PB}^T \cdot {}_P\dot{\boldsymbol{\omega}}_{PB},\end{aligned} \tag{8.70}$$

that is the well known form of the angular part of the Newton-Euler equations, i.e., $\mathcal{J}\dot{\omega} + \omega \times (\mathcal{J}\omega)$. In the following derivations, we use $\times_{d_i}$ to represent the multiplication along the i-th dimension of a tensor.

Since we use Euler Angles ZYX in our optimization $\chi = (\psi, \theta, \phi)^T$, we define the projections from $\dot{\chi}$ and $\ddot{\chi}$ to ω and $\dot{\omega}$ defined as

$$\omega = C(\chi)\dot{\chi}, \quad \dot{\omega} = \dot{C}(\chi, \dot{\chi})\dot{\chi} + C(\chi)\ddot{\chi}, \tag{8.71}$$

with $C(\chi)$ and defined as

$$C(\chi) = \begin{bmatrix} 0 & -s_\psi & c_\psi c_\theta \\ 0 & c_\psi & c_\theta s_\psi \\ 1 & 0 & -s_\theta \end{bmatrix}, \tag{8.72}$$

where we use s_a for $\sin(a)$ and c_a for $\cos(a)$, and

$$\begin{aligned} \dot{C}(\chi, \dot{\chi}) &= \frac{\partial C(\chi)}{\partial \chi} \times_{d_3} \dot{\chi} \\ &= \begin{bmatrix} 0 & -\dot{\psi}c_\psi & -\dot{\psi}c_\theta s_\psi - \dot{\theta}c_\psi s_\theta \\ 0 & -\dot{\psi}s_\psi & \dot{\psi}c_\psi c_\theta - \dot{\theta}s_\psi s_\theta \\ 0 & 0 & -\dot{\theta}c_\theta \end{bmatrix}. \end{aligned} \tag{8.73}$$

8.10.2.3 *Derivatives of Angular Momentum Rate*

Using (8.70), the chain and the product rules, we compute the derivatives of $\dot{l}$ w.r.t. χ, $\dot{\chi}$ and $\ddot{\chi}$.

We begin with $\partial\dot{l}/\partial\ddot{\chi}$. Using the chain rule, it is written as

$$\frac{\partial \dot{l}}{\partial \ddot{\chi}} = \frac{\partial \dot{l}}{\partial \dot{\omega}} \cdot \frac{\partial \dot{\omega}}{\partial \ddot{\chi}} = \mathcal{J}_B C(\chi). \tag{8.74}$$

The derivative of $\dot{l}$ w.r.t. $\dot{\chi}$ is obtained by using once more the chain rule by writing

$$\frac{\partial \dot{l}}{\partial \dot{\chi}} = \frac{\partial \dot{l}}{\partial \omega} \cdot \frac{\partial \omega}{\partial \dot{\chi}} + \frac{\partial \dot{l}}{\partial \dot{\omega}} \cdot \frac{\partial \dot{\omega}}{\partial \dot{\chi}}, \tag{8.75}$$

where the partial derivatives in the last equation are computed as

$$\begin{aligned}
\frac{\partial \dot{l}}{\partial \omega} &= S(\omega)\mathcal{J}_B - S(\mathcal{J}_B \cdot \omega) \\
\frac{\partial \omega}{\partial \dot{\chi}} &= C(\chi), \quad \frac{\partial \dot{l}}{\partial \dot{\omega}} = \mathcal{J}_B \\
\frac{\partial \dot{\omega}}{\partial \dot{\chi}} &= \frac{\partial \left(\dot{C}(\chi, \dot{\chi}) \cdot \dot{\chi} \right)}{\partial \dot{\chi}} \\
&= \left(\frac{\partial C(\chi)}{\partial \chi} \times_{d_3} + \frac{\partial C(\chi)}{\partial \chi} \times_{d_2} \right) \dot{\chi}
\end{aligned} \tag{8.76}$$

To conclude, we derive $\partial \dot{l} / \partial \chi$. Expanding and writing the derivatives, we obtian

$$\begin{aligned}
\frac{\partial \dot{l}}{\partial \chi} = &\frac{\partial S(\omega_{PB})}{\partial \chi} \times_{d_2} \mathcal{J}_B \cdot \omega_{PB} \\
&+ S(\omega_{PB}) \cdot \left(\frac{\partial \mathcal{J}_B}{\partial \chi} \times_{d_2} \omega_{PB} + \mathcal{J}_B \cdot \frac{\partial \omega_{PB}}{\partial \chi} \right) \\
&+ \frac{\partial \mathcal{J}_B}{\partial \chi} \times_{d_2} \dot{\omega}_{PB} + \mathcal{J}_B \cdot \frac{\partial \dot{\omega}_{PB}}{\partial \chi}.
\end{aligned} \tag{8.77}$$

Using the identity in eq. (8.60), we write

$$\frac{\partial \omega_{PB}}{\partial \chi} = \frac{\partial C(\chi)}{\partial \chi} \times_{d_2} \dot{\chi} \tag{8.78}$$

and

$$\begin{aligned}
\frac{\partial \dot{\omega}_{PB}}{\partial \chi} = &\frac{\partial \dot{C}(\chi, \dot{\chi}) \dot{\chi}}{\partial \chi} + \frac{\partial C(\chi) \ddot{\chi}}{\partial \chi} \\
= &\frac{\partial^2 C(\chi)}{\partial \chi \partial \chi} \times_{d_4} \dot{\chi} \times_{d_2} \dot{\chi} \\
&+ \frac{\partial C(\chi)}{\partial \chi} \times_{d_2} \ddot{\chi},
\end{aligned} \tag{8.79}$$

where $\partial^2 C(\chi) / \partial \chi \partial \chi$ is a four-dimensional tensor of Hessians of $C(\chi)$.

The term $\partial S(\omega_{PB}) / \partial \chi$ is computed as

$$\frac{\partial S(\omega_{PB})}{\partial \chi} = \mathcal{S} \left(\frac{\partial \omega_{PB}}{\partial \chi} \right), \tag{8.80}$$

where $\mathcal{S}(\cdot)$ is an operator that produces a three-dimensional tensor whose slices are the skew-symmetric matrix of the columns of the input matrix.

We write the derivative of ${}_P\boldsymbol{\mathcal{J}}_B$ w.r.t. $\boldsymbol{\chi}$ as

$$\begin{aligned}\frac{\partial {}_P\boldsymbol{\mathcal{J}}_B}{\partial \boldsymbol{\chi}} =& \frac{\partial \boldsymbol{R}_{PB}}{\partial \boldsymbol{\chi}} \times_{d_2} ({}_P\boldsymbol{\mathcal{J}}_B \cdot \boldsymbol{R}_{PB}^T) \\ &+ (\boldsymbol{R}_{PB} \cdot {}_P\boldsymbol{\mathcal{J}}_B) \times_{d_2} \frac{\partial \boldsymbol{R}_{PB}^T}{\partial \boldsymbol{\chi}}.\end{aligned} \tag{8.81}$$

To compute the three-dimensional tensor $\partial \boldsymbol{R}_{PB}/\partial \boldsymbol{\chi}$ from the last equation, we recall that $\boldsymbol{R}_{PB} = \boldsymbol{R}_z(\psi) \cdot \boldsymbol{R}_y(\theta) \cdot \boldsymbol{R}_x(\phi)$, where $\boldsymbol{R}_x(\cdot)$, $\boldsymbol{R}_y(\cdot)$, and $\boldsymbol{R}_z(\cdot)$ are elementary rotations around the x, y, and z axes, respectively. Each of the elementary rotations are function of only one of the $\boldsymbol{\chi}$ angles. Recalling that the derivative of a generic rotation matrix $\boldsymbol{R}$ is given by $\boldsymbol{S}(\hat{\boldsymbol{n}})\boldsymbol{R}$, with $\hat{\boldsymbol{n}}$ the direction around which the rotation is happening, we write

$$\frac{\partial \boldsymbol{R}_{PB}}{\partial \boldsymbol{\chi}} = \begin{cases} \boldsymbol{S}(\hat{\boldsymbol{z}})\boldsymbol{R}_{PB}, \\ \boldsymbol{S}\left(\boldsymbol{R}_z(\psi)\hat{\boldsymbol{y}}\right)\boldsymbol{R}_{PB}, \\ \boldsymbol{S}\left(\boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\hat{\boldsymbol{x}}\right)\boldsymbol{R}_{PB} \end{cases}, \tag{8.82}$$

where $\hat{\boldsymbol{x}} = (1,0,0)^T$, $\hat{\boldsymbol{y}} = (0,1,0)^T$, and $\hat{\boldsymbol{z}} = (0,0,1)^T$.

In summary, the gradients of the ZMP constraints in eq. (8.63) w.r.t. $\boldsymbol{\chi}$, $\dot{\boldsymbol{\chi}}$, and $\ddot{\boldsymbol{\chi}}$, are written as

$$\boldsymbol{\nabla}_{\boldsymbol{\chi}} \dot{\boldsymbol{l}} = \frac{\partial^T \dot{\boldsymbol{l}}}{\partial \boldsymbol{\chi}} \gamma \quad , \boldsymbol{\nabla}_{\dot{\boldsymbol{\chi}}} \dot{\boldsymbol{l}} = \frac{\partial^T \dot{\boldsymbol{l}}}{\partial \dot{\boldsymbol{\chi}}} \gamma \quad , \boldsymbol{\nabla}_{\ddot{\boldsymbol{\chi}}} \dot{\boldsymbol{l}} = \frac{\partial^T \dot{\boldsymbol{l}}}{\partial \ddot{\boldsymbol{\chi}}} \gamma, \tag{8.83}$$

where $\gamma = \boldsymbol{S}(\boldsymbol{n})\boldsymbol{d}$.

9

PAPER 6: ADVANCES IN REAL-WORLD APPLICATIONS FOR LEGGED ROBOTS

9.1 ABSTRACT

This paper provides insight into the application of the quadrupedal robot ANYmal in outdoor missions of industrial inspection (ARGOS Challenge) and search and rescue (ERL Emergency Robots). In both competitions, the legged robot had to autonomously and semi-autonomously navigate in real-world scenarios to complete high-level tasks such as inspection and payload delivery. In the ARGOS competition, ANYmal used a rotating LiDAR sensor to localize on the industrial site and map the terrain and obstacles around the robot. In the ERL competition, additional Real-Time Kinematic (RTK)-Global Positioning System (GPS) was used to co-localize the legged robot with respect to a Micro Aerial Vehicle (MAV) that creates maps from the aerial view. The high mobility of legged robots allows overcoming large obstacles, e.g. steps and stairs, with statically and dynamically stable gaits. Moreover, the versatile machine can adapt its posture for inspection and payload delivery. The paper concludes with insight into the general learnings from the ARGOS and ERL challenges.

9.2 INTRODUCTION

Deploying and using a mobile robot in an outdoor scenario is coupled with several challenges. A real-world environment is typically unstructured, unknown and can exhibit obstacles which have to be negotiated. Proper sensing capabilities (e.g. a global positioning system or a laser scanner) are required to localize the robot and map such environments. Moreover, these sensors should be properly integrated into the mobile robot's hardware, as water and dust can pose severe limitations to the reliability of the entire machine. The execution of the path planning and navigation algorithms, together with the control framework which is implemented to execute robust locomotion, can require very high computation capabilities.

Although these are very demanding requirements, recent years have seen several advances in hardware design, computation capabilities and

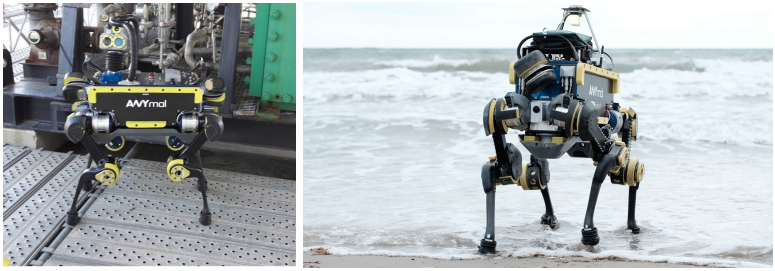


FIGURE 9.1: The quadrupedal robot ANYmal can be employed for industrial inspection (left) and search and rescue missions (right) in harsh environments.

control algorithms which enhance the autonomy and robustness of a mobile robot. Thanks to this progress, mobile robots become ready for new applications to tackle problems like industrial inspection, mining, agriculture, search and rescue missions, and other areas. These environments require robots that can cope with obstacles, rough outdoor terrain and steps. When deployed in an environment designed for humans, the robot needs to deal with stairs, doors, and other impediments.

Instead of creating a specialized robot for every application, over the last years legged robots have proven to serve as a multi-purpose machine that can be easily adapted to suit all kind of environments. As a proof of concept, our work discusses two applications where such a robot can be potentially deployed. Fig. 9.1 shows the quadrupedal robot ANYmal [56] in use for industrial inspection on an industrial site [100] (left) and as medkit delivery robot in a search and rescue scenario (right).

The ARGOS Challenge[1], initiated by TOTAL SA, aims at the application of mobile robotic solutions for offshore oil and gas site inspection. According to [101], the organizers expect a major impact with respect to *i) health, safety and environment (i.e. a reduction of risk to personnel, environment and installation* as well as *ii) operation (i.e. cost reduction, increase of efficiency and production)*. In the ARGOS Challenge, the robots must be able to autonomously navigate on the industrial site and inspect various objects such as pressure gauges, water level gauges, or valve handle positions. They need to analyze the sound of the running pumps in order to identify malfunctioning systems, detect alarm signals, find gas leaks as well as hot spots, and recognize changes that were made on the site (e.g. missing or moved objects). To make the scenario as realistic as possible,

1 http://www.argos-challenge.com

the applied robots must satisfy ATEX (explosion protection) and Ingress Protection (IP) standards. Moreover, during the missions that happen on multiple floors connected by steep stairs, the robots are facing different hurdles such as unexpected obstacles, heavy waterfalls, strong winds, or humans that are working on the site. In contrast to specific robotic devices that are already commercially used in tanks, vessels, or pipes[2] the ARGOS Challenge seeks for very generic robots that can one-to-one take over tasks performed by human specialists. In particular, the requirements regarding mobility are demanding such that the few existing solutions like the wheel-based robots MIMROex [102] or SENSABOT [103] are not applicable. To address these issues, four of the five ARGOS Challenger teams selected in 2013 use tracked vehicles [101], while we propose an innovative solution based on a versatile legged robot.

The initiative to push robots for search and rescue missions started in the early 90s after the Hanshin-Awajii earthquake in Kobe, Japan, and the bombing of the Murrah federal building in Oklahoma City, United States [104, 105]. Recent accidents like the disaster at the Fukushima nuclear power plant in Japan emphasized the need for robotic solutions that can be used to assist human search and rescue teams in hazardous environments. The ERL Emergency Robots (formerly known as *euRathlon*) is an outdoor robotics competition focused on multi-domain (air, land and sea) emergency response scenarios. Inspired by the disaster in Japan, it aims to create real-world robotics challenges for outdoor robots in demanding emergency response events [106, 107].

The 2017 competition was based in Piombino, Italy. It focused on the collaboration of land, air and underwater robots to inspect a simulated disaster scenario and search for survivors to whom a medkit should be delivered. The environment in which the robots had to move in was a very challenging one. The path to be followed was a very rough one and featured different types of terrain, including seaside shore, sticks and high slopes, which can push the locomotion capabilities of a walking robot to the limits.

Similar to the projects above, the Darpa Robotics Challenge (DRC) aims to work up human-scaled robots for disaster response [108]. In contrast to other solutions for search and rescue, the DRC had high requirements regarding mobility and demanded robots that could interface with human environments and use human tools. These high requirements cre-

2 For example, see http://inspection-robotics.com

ated many legged-based systems like the Atlas robot [109–111], WALK-MAN [112], HRP-2 [113], DRC-Hubo [114], and more.

The examples above focus on bipedal robots developed for complex disaster environments. Despite all the advances throughout the DRC, the locomotion performance of humanoid robots is still far behind the performance of multi-legged robots in terms of speed, energetic efficiency, and obstacle negotiation skills. Examples of multi-legged robots with greater locomotion capabilities include HyQ [11] and its successor HyQ2Max [115], MIT's Cheetah [116] and Cheetah II [117], ETH's StarlETH [118]. Boston Dynamics' Spot and SpotMini robot, a direct successor to Big Dog [119], show impressive results of which unfortunately no scientific publications are available.

In contrast to many of the above-mentioned research platforms, ANYmal has been extensively tested in real world applications and under harsh conditions. In this paper we focus on presenting the evaluation of this machine in different outdoor scenarios, namely autonomous industrial inspection and tele-operated search and rescue, together with a description of the choices made regarding system design and the planning and control algorithms which have enabled the robot to locomote autonomously and in tele-operated mode through rough and challenging environments.

9.3 MECHATRONIC SETUP OF ANYMAL

We build upon the modular and lightweight quadrupedal robot ANYmal [56] as a transporter platform for inspection and as a mobile platform for search and rescue missions (Fig. 9.2). The legs of this versatile machine are driven by twelve equal series elastic actuator units [120] mounted at the joints. The robot is designed to achieve a large range of motion, allowing to overcome obstacles and stairs. The structure also allows convenient transportation, compact storage, and simple deployment by a single operator. To keep the design as lightweight as possible, most of the structure is manufactured using carbon fibers. For fall protection, the robot features a rollover bar, a Kevlar belly plate, and shock absorbers. Moreover, force sensors in the feet provide haptic feedback of the environment, enabling safe locomotion even with no exteroceptive feedback. ANYmal is designed in a hierarchical manner: On joint level, every actuator module is connected over a Controller Area Network (CAN) bus and works independently. This allows component-level ingress and ATEX protection as well as fast and simple maintenance in case of hardware failure. On

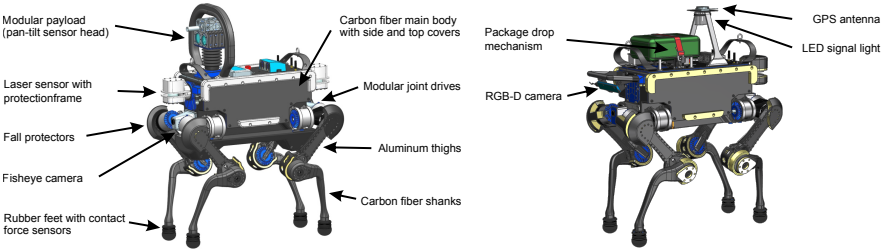


FIGURE 9.2: ANYmal can be embedded with different sets of sensors and tools depending on the scenario in which the robot is deployed. A pan-tilt sensor head (left) is used for inspection tasks. A payload drop mechanism and a GPS mounting system (right) are employed for search and rescue scenarios.

system level, the software stack runs on three independent computers that are connected through an internal network. The first computer (locomotion) hosts all real-time critical elements required for locomotion control and interfaces with the joint modules. The second computer (navigation) is responsible for environmental perception, localization, navigation, and mission execution, i.e. all software parts that are required to autonomously operate the robot. The third computer (inspection) runs all algorithms for inspection and detection.

9.3.1 *Transporter Platform for Inspection*

To perceive the characteristics of its surroundings, ANYmal is equipped with two rotating Hokuyo Light Detection and Ranging (LiDAR) sensors (Hokuyo UTM-30LX-EW). Fig. 9.2 depicts the mounting of the two sensors. They provide detailed scans of the environment and the terrain. Using our point cloud based localization framework, the laser scans can be matched against ground truth data. This is the approach taken in the ARGOS challenge, where TOTAL provided a Computer Aided Design (CAD) model of the site to be inspected.

To execute inspection tasks, we employ a pan-tilt head containing a high-quality zoom camera with high infrared (IR) sensitivity, a thermal camera, an ultrasonic and a regular microphone, a gas detection sensor, and Light Emitting Diode (LED) illuminators. Since our robot can move its base in all directions, there is no need to employ the robot with an extension mechanism or arm.

For possible tele-operation scenarios, the robot is additionally equipped with two wide-angle cameras pointing forward and backward, providing an omnidirectional view of the robot's surroundings. Combining the camera images with the 3D laser point cloud allows us to create a virtual reconstruction of the environment and provides the operator with a 3rd person view. Additionally, these cameras are used to detect humans around the robot for safe interaction with humans.

The proposed system can operate fully autonomous with onboard batteries for more than 2.5 hours. To extend this lifetime, ANYmal can autonomously dock to recharge the battery and to pressurize the main body with Nitrogen for ATEX compliance [121].

9.3.2 *Mobile Platform for Search and Rescue Missions*

Outdoor missions, such as those executed during the ERL challenge, might be executed in environments which do not have the geometric richness needed to properly make use of localization algorithms. To overcome this limitation, ANYmal can be equipped with a Piksi Multi[3] module, an RTK GPS receiver mounted on the main body of the robot. This component, in combination with a fixed RTK base station, is able to provide position measurements with an accuracy between 1.0 cm to 5.0 cm on the horizontal plane and between 1.5 cm to 8.0 cm in vertical direction. However, position measurements become unreliable when the robot is in the proximity of tall structures which interfere with the sensing capabilities of the Piksi Multi. For this reason, ANYmal is also equipped with one rotating laser sensor in the back, as described in Section 9.3.1. Our localization framework fuses the multiple sources of position measurements and relies more on the laser sensor when the robot is locomoting close to buildings.

To deliver a med-kit to a victim at the ERL challenge, the pan-tilt head in Section 9.3.1 is replaced by a custom payload delivery system. The first aid kit is strapped onto a carbon fiber plate, which is rigidly attached to the upper part of ANYmal's main body. Beneath the carbon fiber plate, a servo motor and a belt mechanism are installed. By rotating the servo, the belt mechanism is unlatched, similar to the ones used in cars as safety belts and pulled in by a rotational spring.

For navigation and foothold planning, the machine is equipped with an RGB-D camera in the front, which provides detailed scans of the environ-

3 Piksi Multi Robot Operating System (ROS) driver available online at https://github.com/ethz-asl/ethz_piksi_ros

FIGURE 9.3: The robot ANYmal walking blindly on a sandy slope at the ERL challenge (left) and climbing a set of stairs (right). Blind walking is based on an optimization-based approach to controlling a dynamic gait, i.e., trotting gait. Climbing over challenging obstacles like stairs includes map-based motion planning. Moreover, footholds (red dots) and base and leg motions (red lines) are generated by processing a terrain map with the help of onboard sensors [16].

ment and the terrain. During the ERL challenge, we have been working with the RealSense ZR300 [122], which outputs high-resolution depth images at a higher update rate compared to the rotating laser sensor. The sensor is based on active stereo vision which uses a projected light pattern to improve the stereo matching algorithm. The active light projection also allows to perceive objects with homogeneous appearance and stereo matching is also possible in dark environments [123].

9.4 LOCOMOTION IN ROUGH TERRAIN

Autonomous locomotion on ground that is not perfectly deterministic requires locomotion control strategies that are robust against irregularities. Moreover, larger obstacles and more challenging terrain features require a full control pipeline including terrain perception, map processing, and map-based motion planning. Fig. 9.3 shows both control approaches in action.

9.4.1 *Quadrupedal Robot Locomotion through Optimization-Based Control*

Several modules contribute to the robustness of locomotion in rough terrain. The following section gives an overview of terrain estimation and adaptation, motion generation and motion execution.

9.4.1.1 *Terrain Estimation and Adaptation*

Legged locomotion on high gradient slopes like in Fig. 9.3 (left) imposes several challenges since the robot's configuration needs to be adapted to avoid slipping and kinematic limits. Without exteroceptive sensing, the robot is only able to estimate the terrain properties by incorporating state estimation and joint measurements.

We locally model the terrain using a three dimensional plane, which is defined by a fixed point $\boldsymbol{r}_0 = \begin{bmatrix} x_0, y_0, z_0 \end{bmatrix}^T$ and a normal vector $\boldsymbol{n} = \begin{bmatrix} a, b, c \end{bmatrix}^T$. As described in [52], given a generic point $\boldsymbol{r} = \begin{bmatrix} x, y, z \end{bmatrix}^T$, we model the plane by writing

$$(\boldsymbol{r} - \boldsymbol{r}_0)^T \boldsymbol{n} = \boldsymbol{0} \implies z = \frac{d - ax - by}{c}, d = ax_0 + by_0 + z_0. \tag{9.1}$$

Assuming that $c = 1$, i.e. the plane will never be perpendicular to the world-frame z-axis, there are three parameters which need to be estimated, namely $\boldsymbol{\pi} = \begin{bmatrix} a, b, d \end{bmatrix}^T$. We solve a least-squares problem by fitting the plane through the most recent contact location $\boldsymbol{r}_i = \begin{bmatrix} x_i, y_i, z_i \end{bmatrix}$ of each leg. The height z_i of each of these positions can be written as a linear function of the parameters $\boldsymbol{\pi}$

$$z_i = \begin{bmatrix} -x_i & -y_i & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ d \end{bmatrix}. \tag{9.2}$$

By collecting the four measured contact heights, we compute the estimated plane parameters $\hat{\boldsymbol{\pi}} = \begin{bmatrix} \hat{a}, \hat{b}, \hat{d} \end{bmatrix}^T$ by writing

$$\underbrace{\begin{bmatrix} \vdots \\ z_i \\ \vdots \end{bmatrix}}_{\boldsymbol{\zeta}} = \underbrace{\begin{bmatrix} & \ddots & 1 \\ -x_i & -y_i & \vdots \\ & \ddots & 1 \end{bmatrix}}_{\boldsymbol{H}} \begin{bmatrix} \hat{a} \\ \hat{b} \\ \hat{d} \end{bmatrix} \implies \hat{\boldsymbol{\pi}} = \boldsymbol{H}^{\dagger} \boldsymbol{\zeta}, \tag{9.3}$$

where $\boldsymbol{H}^{\dagger}$ is the pseudo-inverse of $\boldsymbol{H}$. From the estimation of these parameters we define a so called *Control Frame C* which is aligned to the estimated

local terrain and to the heading direction of the robot. This enables the generation of motion plans which keep the torso of the robot always parallel to the local estimation of the terrain, i.e. this approach allows to blindly and robustly [22] overcome challenging environments, e.g., sloped terrain.

9.4.1.2 *Motion Generation*

The locomotion behavior of ANYmal can be specified through a periodic contact schedule (or *gait pattern*), or through incoming contact events. In the former case, the gait pattern defines, in phase domain, the lift-off and touch-down timings of each leg. This information, coupled with the actual estimated contact state, defines whether a leg is sustaining the overall motion of the robot (i.e., it is a *support* leg) or it is *swinging* to a new desired contact location. Based on the specific timing of the gait pattern, different gaits can be achieved. During the ERL and the ARGOS challenges we used a trotting gait on flat and on terrain like the one depicted in Fig. 9.3 (left). A trot is a dynamic gait that schedules two diagonal legs to be in ground contact at each time and the other two legs to be in swing motion. This results in faster movements with respect to other static gaits and at the same time achieves robust locomotion with respect to disturbances due to the high step frequency that allows for step corrections [22, 52]. Experimental results at the ERL challenge have shown that adding a small stance phase between each swing phase lead to an increase in the overall locomotion robustness.

Several motion planning and control strategies are implemented to guarantee balance while walking. First, the location of the planned footholds is computed based on a modified version of the *inverted pendulum* model [22], which computes footholds as a function of the torso velocity error. To define the complete swing leg trajectory of each foot, a fifth-order spline is fitted through the lift-off and the touch-down locations. Second, the torso motion is planned such that the whole-body center of mass lies, at all times, over the support polygon (i.e., the convex hull of the contact locations). The height of the main body is kept constant to a user-specified value w.r.t. the local estimation of terrain. The orientation of the estimated terrain, computed in 9.4.1.1, is used as a reference orientation of the main body.

9.4.1.3 *Motion Execution*

To track the torso motion reference described in 9.4.1.2, a *Virtual Model Controller* is implemented which generates a virtual wrench $\boldsymbol{w}^*$ that acts on the main body of the robot. We compute the components of this wrench as a function of the reference torso position $\boldsymbol{r}^*_{IB}$, velocity $\boldsymbol{v}^*_B$ and acceleration $\boldsymbol{a}^*_B$, as well as desired rotation quaternion $\boldsymbol{q}^*_{IB}$ and angular velocity $\boldsymbol{\omega}^*_{IB}$. The two components $\boldsymbol{f}^*$ and $\boldsymbol{t}^*$ of the wrench are given by

$$\begin{aligned}\boldsymbol{f}^* &= \boldsymbol{K}^f_p(\boldsymbol{r}^*_{IB} - \boldsymbol{r}_{IB}) + \boldsymbol{K}^f_i \int (\boldsymbol{r}^*_{IB} - \boldsymbol{r}_{IB}) + \boldsymbol{K}^f_d(\boldsymbol{v}^*_B - \boldsymbol{v}_B) + m\boldsymbol{a}^*_B - \sum_{k\in\mathcal{B}} \boldsymbol{f}^g_k \\ \boldsymbol{t}^* &= \boldsymbol{K}^t_p(\boldsymbol{q}^*_{IB} \boxminus \boldsymbol{q}_{IB}) + \boldsymbol{K}^t_i \int (\boldsymbol{q}^*_{IB} \boxminus \boldsymbol{q}_{IB}) + \boldsymbol{K}^t_d(\boldsymbol{\omega}^*_{IB} - \boldsymbol{\omega}_{IB}) - \sum_{k\in\mathcal{B}} \hat{\boldsymbol{r}}_{IS_k} \boldsymbol{f}^g_k,\end{aligned} \tag{9.4}$$

where $\boldsymbol{K}$ are diagonal gain matrices, $\mathcal{B}$ is the set of the actuated rigid bodies, $\boldsymbol{f}^g_k$ is the gravitational force acting at the center of mass of the k-th body, $\hat{\boldsymbol{r}}_{IS_k}$ is the skew-symmetric matrix function of the position of the center of mass of body k such that $\hat{\boldsymbol{r}}_{IS_k}\boldsymbol{f}^g_k = \boldsymbol{r}_{IS_k} \times \boldsymbol{f}^g_k$, and $\boxminus$ is the box-minus operator [53]. The wrench integrates feed-forward terms such as gravity compensation and reference acceleration, as well as feed-back terms which encode deviation of the main body estimated pose and twist from the reference motion. At each control loop, the virtual wrench is distributed to the legs which are in contact through a *Contact Force Distribution* algorithm, a constrained optimization problem which computes reference contact forces for each leg. These are computed such that certain constraints are not violated, e.g. such that the contact forces are inside the friction cone (to avoid slipping) and such that the actuator commands are limited (i.e. to avoid torque limits). This is formulated as a Quadratic Programming problem as

$$\begin{aligned}\boldsymbol{\lambda}^* = \arg\min \quad & (\boldsymbol{A}\boldsymbol{\lambda} - \boldsymbol{b})^T \boldsymbol{S}(\boldsymbol{A}\boldsymbol{\lambda} - \boldsymbol{b}) + \boldsymbol{\lambda}^T \boldsymbol{W}\boldsymbol{\lambda} \\ & \underbrace{\begin{bmatrix} \cdots & \boldsymbol{I} & \cdots \\ \cdots & \hat{\boldsymbol{r}}_{BF_i} & \cdots \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \vdots \\ \boldsymbol{\lambda}_i \\ \vdots \end{bmatrix}}_{\boldsymbol{\lambda}} = \underbrace{\begin{bmatrix} \boldsymbol{f}^* \\ \boldsymbol{t}^* \end{bmatrix}}_{\boldsymbol{b}} \\ \text{s. t.} \quad & \lambda_{N,i} \geq \lambda_{N,min} \\ & \|\boldsymbol{\lambda}_{T,i}\| \leq \mu\lambda_{N,i} \\ & \tau_{min} \leq \tau_{j,j}(\lambda) \leq \tau_{max},\end{aligned} \tag{9.5}$$

where $\boldsymbol{S}$ is a diagonal weighting matrix used to penalize different components of the cost function, μ is an estimated friction coefficient, $\boldsymbol{r}_{BF_i}$ is the position vector of the i-th foot, λ_N is the normal component of the contact force, which is aligned with the terrain normal estimated in (9.2), and $\boldsymbol{\lambda}_T$ is the contact force projected onto the estimated terrain. Finally, assuming quasi-static dynamics (i.e. only the gravity terms from the equations of motion are considered), the actual actuator commands τ^* are computed as

$$\tau_k^* = -\sum_{i \in \mathcal{I}(\mathrm{II})} \boldsymbol{J}_i^T \boldsymbol{\lambda}_i^* - \sum_{k \in \mathcal{B}} \boldsymbol{J}_k^T \boldsymbol{f}_k^g, \tag{9.6}$$

where $\boldsymbol{J}_k$ is the translation Jacobian of the k-th leg respectively. To track the motion of the swing legs, a joint-space motion controller is implemented which tracks joint positions and velocities obtained through inverse kinematics of Cartesian space reference states (see Fig. 9.4).

To decide whether a leg is part of the contact force distribution or if it should be swinging, knowledge about its contact state is required. We have implemented our control algorithms based both on force estimation and on optical force sensor measurements. In both cases, we obtain a measurement or estimation of the contact forces $\boldsymbol{F_{ext}} \in \mathbb{R}^{n_e}$ acting on each foot of the robot.

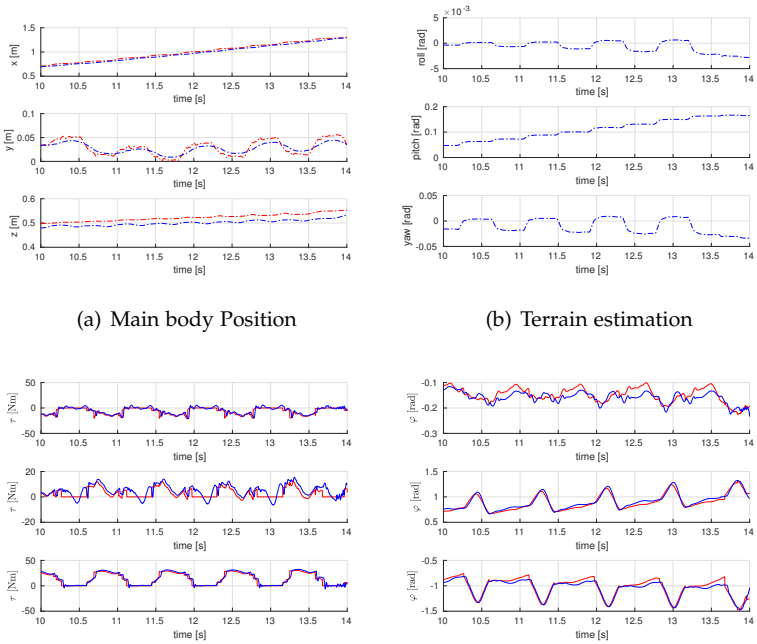
Force sensors can be employed to directly measure the reaction forces acting on the feet. While these provide a direct measurement, they are prone to drift in the raw data. This can be compensated by implementing outlier detection algorithms, which in turn increases the complexity of the contact detection.

When force or contact sensors are not available, we estimate the contact forces using the equations of motion. By incorporating joint positions $\boldsymbol{q_j}$, velocities $\dot{\boldsymbol{q}}_j$ and accelerations $\ddot{\boldsymbol{q}}_j$, we can solve for $\boldsymbol{F_{ext}}$ by writing

$$\boldsymbol{F_{ext}} = (\boldsymbol{J}_{\boldsymbol{ext}}^T)^+ (\boldsymbol{M_j}(\boldsymbol{q_j})\ddot{\boldsymbol{q}}_j + \boldsymbol{b_j}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g_j}(\boldsymbol{q}) - \boldsymbol{\tau}) \tag{9.7}$$

where $\boldsymbol{J_{ext}} \in \mathbb{R}^{n_e \times n_{q_j}}$, $\boldsymbol{M_j}(\boldsymbol{q_j}) \in \mathbb{R}^{n_{q_j} \times n_{q_j}}$, $\boldsymbol{b_j}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n_{q_j}}$, $\boldsymbol{g_j}(\boldsymbol{q}) \in \mathbb{R}^{n_{q_j}}$ and $\boldsymbol{\tau}$ are the geometric Jacobian of the contact location, the sub matrix of the mass matrix associated to joint accelerations only, the lower rows of the vector of Coriolis and centrifugal terms, the lower rows of the vector of gravitational terms and the joint torques, respectively. Finally, the binary contact state is obtained by thresholding $\boldsymbol{F_{ext}}$.

Both approaches are essential when blind locomotion is executed, but both come with their trade-offs. Using force sensors provides a direct forces measurement which can be thresholded to obtain a contact state.



(a) Main body Position

(b) Terrain estimation

(c) Joint Torques

(d) Joint Positions

FIGURE 9.4: Experimental results demonstrating ANYmal trotting up a slope. a) The tracking of the position of the torso while walking. b) The estimation of the terrain using the contact locations. c) Tracking of the joint torques for the three actuators of left-fore leg and d) the joint position tracking for the same leg. Signals depicted in red represent references quantities, while signals in blue are measured ones.

This measurement, however, exhibits both bias and drift which must be compensated to obtain a reliable contact state. On the other hand, using the equations of motion provides a drift-less force estimation. When faster motions are executed, a higher threshold must be employed with a method to increase the robustness in the contact estimation.

9.4.2 *Map-Based Locomotion Planning*

Locomotion planning over unknown terrain requires the integration of several modules that allow the robot to reason about its environment. The

issue is even more crucial in rough terrain, where the robot might not be able to fulfill its task if the terrain properties are not taken into account. The following section shows how we combine terrain mapping and map processing, motion planning and motion control to a fully integrated system [123].

9.4.2.1 *Terrain Mapping and Map Processing*

In our previous work [16], we have presented an approach to generate a two-and-a-half-dimensional (2.5D) elevation map by combining the robot's pose estimation and the measurements from a range sensor. This representation generates grid cells, which include height and uncertainty of the terrain. Fig. 9.3 (right) shows the elevation map generated by the robot while climbing a set of stairs using the robot's onboard sensor, i.e., assisted stereo camera (RealSense ZR300).

Given the continuously updated robot-centric elevation map, the data is further processed to generate more knowledge about the terrain. A surface normal vector is generated for each cell of the elevation map by using the Principal Component Analysis in a circular area around the given grid cell. To evaluate the terrain upfront the robot for feasible footholds, each cell of the elevation map is associated with a foothold score between zero and one. A linear combination of different quality measures gives an indication of whether a foothold is safe or not. The quality measures are based on geometric features like slope, curvature, roughness, and uncertainty. A three-dimension signed distance field is generated from the elevation map to plan collision-free trajectories. The signed distance field holds a distance to the nearest surface [16].

9.4.2.2 *Motion Planning*

Based on the map features described in Section 9.4.2.1 using the robot-centric elevation map, the robot is planning its whole-body motion according to the given terrain information. First, nominal footsteps are generated based on a geometrical approach which is recomputed after every footstep. The nominal footstep generation is based on default step lengths in longitudinal, lateral and yaw direction. By continuously recomputing the footsteps after each control loop, the footstep planner is robust against deviations from the original plan. In a next step the nominal footsteps are optimized based on the foothold scores described in Section 9.4.2.1. Given the nominal foothold position, the location around the foothold is checked

with increasing Euclidian distance from it. If the nominal foothold position is not feasible, this approach results in a new foothold position around the nominal foothold with a valid score. Additionally, a pose optimizer verifies if the new footholds are reachable. The pose optimization takes into account a reachability cost based on kinematic limits and a cost for the static stability margin (distance between the planar projection of the COM of the support polygon). This results in a non-linear optimization problem that is solved as a SQP. Finally, the swing leg trajectories are planned by taking the current and final foothold positions into account, as well as the signed distance field created in Section 9.4.2.1. For this purpose, we optimize over the knot points between the starting and end positions of the spline trajectories. This optimization results in a weighted sum of trajectory length and collision costs with the environment [16].

Given the desired trajectories of the torso and feet of the motion planner, the motion execution tries to follow these trajectories and be robust to external disturbances. The motion execution is based on the software framework Free Gait [66], which internally uses the virtual model controller and contact force distribution introduced in Section 9.4.1.3. In this case, the controller is not using the estimated terrain from Section 9.4.1.1 but instead uses the terrain information from the distance sensor in Section 9.4.2.1. Moreover, the quadratic program that maps the net wrench acting on the torso into contact forces is constrained by tilted friction pyramids. The normals of the friction pyramids are aligned with the surface normal vectors of the terrain (Section 9.4.2.1), and by taking the surface normals inside the force distribution into account the controller prevents slipping on inclined terrain [16].

9.5 MULTI-SENSOR LOCALIZATION

For many applications, mission goals or waypoints are often specified as absolute coordinates in some reference frame (e.g., GPS coordinates) necessitating the robot to localize with respect to this frame. Not all localization techniques work in every environment and situation. Localizing with point cloud Iterative Closest Point (ICP) matching, for example, does not work in environments with little or highly repetitive geometric structure. For this reason, we chose to pursue a multi-sensor localization approach.

9.5.1 *Sensors for Localization*

ANYmal in its configuration used for the ERL search and rescue challenge is equipped with an RTK GPS receiver, as well as a 2D LiDAR mounted on a rotating servo-motor to obtain full 3D point clouds. These two techniques have complimentary failure modes and are therefore well-suited to achieve robust localization in different and changing environments. GPS fails when communication to satellites is degraded which typically happens inside buildings or close to high vertical structures where ICP based localization excels.

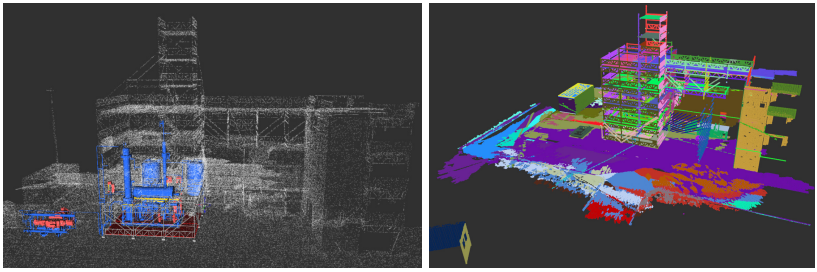
9.5.1.1 *RTK GPS*

GPS localization provides a globally consistent position and is per design drift-free. Using the RTK GPS technique its position measurement can reach centimeter-level accuracy relative to a stationary reference beacon which sends correction signals. However, since only a position measurement can be obtained, orientation drift of the proprioceptive odometry pose estimate cannot be corrected. Fig. 9.5 depicts the path taken by ANYmal during the ERL Challenge. The data was acquired using the onboard GPS receiver.

9.5.1.2 *LiDAR*

For localization using 3D point clouds acquired by a rotating line LiDAR sensor (Hokuyo UTM-30LX-EW), incoming point clouds are aligned to an existing map using the ICP algorithm [124]. By continuously scanning the environment, the robot is able to track its position and orientation. The individual scans (half rotation of the laser, 20'000 points) are dewarped using the local state estimation from Inertial Measurement Unit (IMU) and leg kinematics [125]. The ICP algorithm then searches for neighbor points between the 3D point cloud of the single scan and the map (Fig. 9.6(a)) and tries to minimize the sum of all their distances, which takes about 0.4 s. The estimated location is then fed back to the systems state estimator as an update measurement. The challenging part of this setup is that the duration of the ICP matching step is variable. To compensate for the potentially old position update, the pose was further propagated using the local state estimation [125]. In scenarios where an accurate point cloud map of the environment is available, this provides a drift-free 6-DOF pose estimate for the robot, given that an initial alignment estimate is available (e.g.,

FIGURE 9.5: An overlay of the path taken by ANYmal at the ERL Challenge with the map data provided by Google Maps. The path involved walking over sand, slopes and rough terrain.



(a) Point Cloud (b) Plane Segmentation

FIGURE 9.6: a) The ICP algorithm matches every scan acquired by the LiDAR sensor during one half rotation with the known reference map by minimizing the distance between both point clouds. b) The point cloud of the site is segmented into planes for global localization.

through odometry). Localization accuracy is typically centimeter-level or subcentimeter-level for position, and better than 1° for orientation.

In applications where a prior map of the environment cannot be obtained, the localization map is built incrementally by adding incoming point clouds to the map after aligning them through ICP. In these use-cases, the ICP estimate will exhibit drift according to the accumulated alignment error of incoming point clouds added to the map. The observed drift rate is mostly dependent on the incoming point clouds' overlap with the existing map, which is influenced by the LiDAR range and viewing distance.

To converge to the correct solution, ICP requires that the errors of the initial guess of the robot's pose be less than 1.5 m in position and 30° in orientation. In case of larger errors (i.e., at initialization or after loss of localization), the robot uses a plane matching algorithm. This algorithm searches for planes in the two 3D point clouds and groups them to all possible triples (see Fig. 9.6(b)). A similarity analysis of a scan and a reference plane triple can be done quickly by comparing the interplanar angles. If the plane triples are similar, their relative transformation can be directly computed from the plane parameters. Using this transformation, the scanned point cloud is expressed in the reference frame and a plausibility analysis is done by computing the nearest neighbor ratio. If a certain threshold is met, the associated relative transformation between the clouds is taken to derive the robot's pose. This approach has proven to work very reliably during all missions and typically took between 20 and 100 s.

One caveat of ICP-based localization is that some structure in the map is required, such that the incoming point cloud can be aligned to it. This is especially problematic in large open areas with little vertical structure where alignment in horizontal direction might fail.

9.5.2 *Sensor Fusion*

Sensor fusion is needed because pose and position measurements acquired from the sources mentioned above have varying accuracy and are measured in different coordinate systems. For robot control, a smooth state estimate without jumps is desired to prevent erratic behavior of the robot. Because GPS and ICP provide absolute measurements, fusing these in a tightly-coupled fashion with the leg-odometry estimator is not an option. Therefore, we derive a loosely coupled estimation framework whose only

task is to relate coordinate sensor frames to the odometry frame and to correct the drift accumulated by the odometry state estimator.

9.5.2.1 *Coordinate Frames*

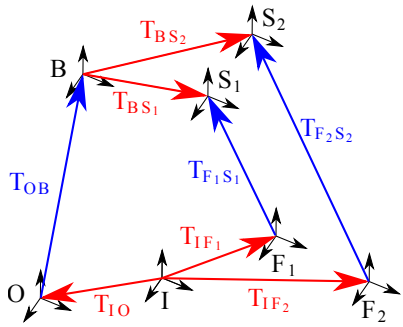
We define a virtual inertial fixed frame I which will be used for path planning. The odometry state estimator estimates the pose of the robot base frame T_B in the odometry frame O. The transform corresponding to the estimated odometry pose is denoted as T_{OB}. Each sensor i measures the pose or position of the coordinate frame S_i, which is attached to the sensor, in a reference coordinate frame F_i. The coordinate sensor frame might be offset from the robot base by a transform T_{BS_i}. The pose or position transform measured by the sensor is denoted as T_{FS_i}. The sensor index i will be omitted in some of the following explanations to achieve a more readable notation. The relation of all transforms described above is shown in Fig. 9.7(a).

The purpose of the virtual inertial frame I is to decouple the drift dynamics of the odometry estimator from the sensor reference frames F_i. If we estimated the transform T_{OF_i} between odometry and sensor reference frame directly, a drift of the state estimator would show in all filter states which estimate these transforms, in addition to the actual dynamics of the sensor reference frame. It would also mean that we need to update all other sensor transforms if we receive a measurement from a single sensor because we gain information on the odometry drift which is encoded in all transforms T_{OF_i}.

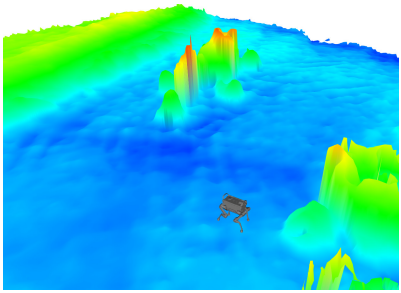
By decoupling all frames through the intermediate inertial frame I we only need to update the transforms T_{IF_i} and T_{IO} when we receive a measurement from sensor i. Additionally, if one sensor N drops out for a period of time, we can still keep the transform T_{OF_N} up-to-date for some time, because it is decomposed into T_{IF_N} and T_{IO}. T_{IF_N} is quasi-static and T_{IO} can be estimated through updates from other sensors.

9.5.2.2 *Implementation*

We implemented the coordinate frame estimation using a Two-State Information Filter (TSIF) [90] which offers great model flexibility because it does not require an explicit process-model. The state vector consists of the translational and rotational part of the odometry drift transform T_{IO}, the ICP related transforms $T_{BS_{ICP}}$ and $T_{IF_{ICP}}$ and GPS transforms $T_{IF_{GPS}}$ and $T_{BS_{GPS}}$. For the GPS sensor, the robot base to antenna transform $T_{BS_{GPS}}$



(a) Colocalization



(b) Colocalization

FIGURE 9.7: a) Coordinate frame transforms between (I)nertial, (O)dometry, (B)ase, (S)ensor-attached and sensor reference (F)rame. Transforms marked red are estimated, and blue ones are measured. b) ANYmal localized in an elevation map built by a MAV equipped with a GPS receiver.

does not need a rotational component because the position measurement is independent of the antenna orientation. Odometry, inertial, and GPS reference frame all have their z-axis aligned with the gravity vector so that we can fix roll and pitch and only estimate the yaw angle of T_{IO} and $T_{IF_{GPS}}$. The transformation between the robot base and GPS as well as LiDAR are fixed and could be measured with sufficient accuracy. They were therefore excluded from the recursive filter update.

The TSIF is based on formulating update residuals which correlate the previous state estimate to the current state estimate. To support a multitude of different localization techniques and sensors, we defined generic residuals for position and orientation measurements, which are processed to update the coordinate transform estimates. Measurements from a pose sensor can be treated as one position and one orientation measurement. The residuals for the position update $\boldsymbol{\Delta}_{\text{pos}}$ and the orientation update $\boldsymbol{\Delta}_{\text{rot}}$ are defined as

$$\begin{aligned} \boldsymbol{\Delta}_{\text{pos}} &= \boldsymbol{r}_{IO} + \boldsymbol{r}_{OB} + \boldsymbol{q}_{IO} \cdot \boldsymbol{q}_{OB} \cdot \boldsymbol{r}_{BS} - \boldsymbol{q}_{IF} \cdot \boldsymbol{r}_{FS} - \boldsymbol{r}_{IF} \\ \boldsymbol{\Delta}_{\text{rot}} &= \log(\boldsymbol{q}_{IO} \cdot \boldsymbol{q}_{OB} \cdot \boldsymbol{q}_{BS} \cdot \boldsymbol{q}_{FS}^{-1} \cdot \boldsymbol{q}_{IF}^{-1}) \end{aligned} \tag{9.8}$$

where $\boldsymbol{r}_{XY}$ and $\boldsymbol{q}_{XY}$ are the positional and rotational component of the transform $\boldsymbol{T}_{XY}$ between the generic frames X and Y. The log function [53] maps rotation matrices to their corresponding three-dimensional rotation vector.

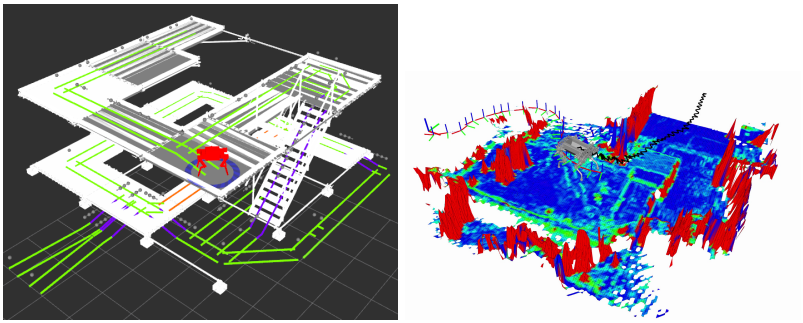
9.6 AUTONOMOUS AND SEMI-AUTONOMOUS NAVIGATION

The ability to autonomously move in challenging terrain requires that the robot can precisely (and globally) localize (Section 9.5), to map its environment (Section 9.4.2.1), to plan a navigation path, as well as to detect and overcome obstacles. The mission contains two consecutive states for *path planning* and *path following*. The *path planning* requires a start and a goal pose (whereby the start may be the robot's current pose) and outputs a path. The *path following* takes this path as input and computes desired forward, sideways and rotational velocities with respect to the robot's body frame for the locomotion controller described in Section 9.4.

9.6.1 *Autonomous Navigation for Industrial Inspection*

Industrial inspection tasks require a fully autonomous system that is capable of detecting unforeseen obstacles. In general, the environment is known, and therefore the mission including the path can be set up in advance. Nevertheless, the robot is expected to detect local changes in the environment. For path planning, the robot features two complementary algorithms, namely the *pose graph planner* and the *traversability planner*.

The *pose graph planner* is used for global path planning, when the robot needs to navigate from one pose to another on the site. The pose graph (Fig. 9.8(a)), which is created once for the entire site, is a representation of the accessible and safely traversable areas on the reference map. It consists of a tree of nodes with according connections (edges). Since the individual areas are mostly flat, the node is a degenerated pose containing position and yaw information. The type of motion that the robot can execute, i.e., the type of gait or climbing maneuver, is encoded in the edge. The *pose graph planner* computes a path by using an A* algorithm on the node tree. This method has several advantages: The planning is very fast (due to the limited dimension of the problem), and the result is entirely deterministic. Any two points on the site can be connected by the pose graph planner. If a point does not lie on the pose graph, the entry or exit nodes are determined by closest proximity evaluation, even if they do not lie on it. Thanks to the short planning time, the robot can continuously and in real-time re-plan its path independent on the event of blocked paths, changed missions, or emergency situations.



(a) Pose Graph (b) Local Planner

FIGURE 9.8: a) The pose graph consists of nodes (blue dots) and edges (green lines). b) While following the nominal trajectory, the traversability planner adapts the path around untraversable areas.

For local path planning, e.g. when an obstacle blocks the global path of the robot, the *traversability planner* [126][4] comes into play. It builds up a map containing the estimated traversability of the environment, which is computed by fusing various filters for slope, roughness, or step heights using the acquired LiDAR data (Fig. 9.8(b)). After planning the local path with a sampling-based RRT* planner, the mission replaces the blocked segment of the global path with the alternative local path. Additionally, the robot can overcome obstacles like steps and stairs by executing the map-based locomotion planner described in Section 9.4.2.

9.6.2 *Semi-Autonomous Navigation for Search and Rescue Missions*

Throughout inspection tasks, a finite set of events may be assumed, and therefore a robot may interact with such an environment in complete autonomy. In a search and rescue scenario, where several unforeseen events may happen and even be triggered by interaction with the environment, complete autonomy of an assisting robot is not yet feasible. Therefore, it is crucial to have an easy to handle and intuitive teleoperation solution, such that an operator may use the assisting robot to its full advantages. To facilitate this, a number of tools and an improved GUI (see Fig. 9.9) were introduced throughout the ERL challenge.

4 Traversability estimation online available: https://github.com/ethz-asl/traversability_estimation

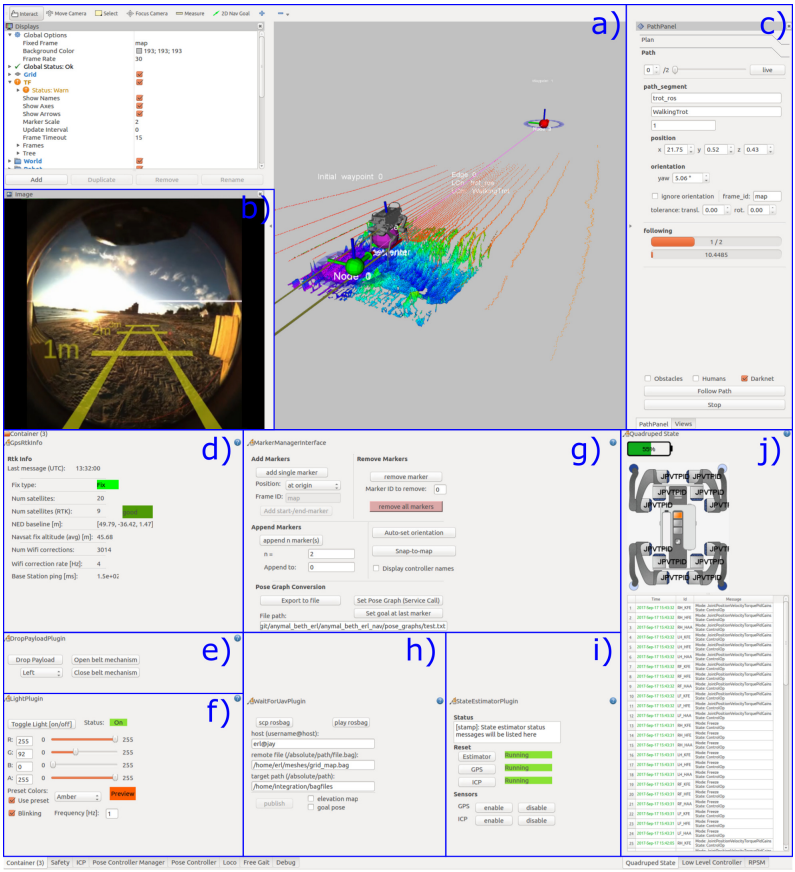


Figure 9.9: The GUI used throughout the ERL challenge shows the robot's status and allows the operator to interactively operate the robot through the rough environment. Thereby, a) shows the robot, the raw point cloud from the sensor unit, the elevation map, and the robot's global path. The operator can interact with the robot by adapting the global path in the three-dimensional space; b) shows the front camera's image of the robot. In addition, the yellow lines indicate the transformed robot's width and distances projected into the camera image; c) gives the operator a status of the path follower as well as the option to stop and start the path following; d) depicts an interface of the GPS RTK status; e) is an interface to drop the payload to the left or right side, respectively; f) is a tool to interact with the robot's status light; g) is the path creator tool interface the operator uses to plan and modify global paths online; h) is an interface for data exchange between multiple robots, e.g., flying and walking robot; i) allows enabling/disabling of GPS and/or ICP measurements to be considered by the frame estimation software as well as it's status and j) gives the operator a visual feedback of the actuator, battery, state estimator and contact states.

The various elements of the GUI are implemented to simplify access to different functionalities and options of the robot's software, such as streaming helper lines into a video feed, similar to the ones known in a car's backward facing parking camera. Apart from these auxiliary tools, the main improvement to the existing teleoperation framework is the introduction of a path creator tool. As described in Section 9.6.1 ANYmal uses a tree of graphs for efficient global planning. However, it is impractical to prepare such a graph in a search and rescue environment. Not all paths are known a priori, and new paths may appear at a later time due to the dynamic nature of such an environment. This path creator tool aims to facilitate intuitive and straightforward planning of a global path. An operator may do this by placement and subsequent interaction of markers into the robot's environmental representation. The interaction is handled via the tool's interface, a ROS compatible RQT[5] plugin. Thereby, each marker corresponds to a *node* and the path is converted online into a *pose graph* and is then used as a new global plan, on which the robot navigates.

While this tool was useful for various applications, it proved to be especially convenient in an exploration setting. While the robot is building up its map using LiDAR measurements, a human operator can interpret the camera's live video stream in parallel to the map being constructed. Since the camera looks much further ahead than the LiDAR, the operator can plan further into unknown terrain than the robot would have been able with only LiDAR data.

An additional feature of the graph creator tool is its ability to export and store planned paths and subsequently load them again. While performing different tasks, the robot may need to walk similar paths multiple times. This feature lets the operator plan a global path once, then reload and modify it as necessary for all subsequent tasks and thereby quickening the planning process substantially.

9.7 COLLABORATION WITH FLYING ROBOT

An MAV is a compact and agile platform that can fly either in indoor or outdoor environments while gathering data about the surrounding area. A collaboration between the RSL and the ASL of ETH Zurich was established to work towards the ERL challenge. The MAV focused on fusing vision-based odometry with RTK GPS to increase accuracy and robustness, and on creating a map of the environment and of the terrain from

5 Refer to http://wiki.ros.org/rqt for further information.

stereo vision. ANYmal could then use the terrain reconstruction to plan a traversable path and reach the desired goal: a collapsed building to inspect and explore. In order to coordinate between the robots, the same RTK system (Piksi Multi) was employed on the flying and ground robot, such that they could localize against a common coordinate frame while outdoors, and transfer maps created with the MAV to ANYmal during operation.

9.8 REAL-WORLD APPLICATIONS FOR LEGGED ROBOTS

Our goal is to provide a mobile platform that is versatile enough to cope with many real-world tasks. During the ERL and ARGOS challenges, we proved that the legged robot ANYmal could perform industrial inspection[6] and to be used as a payload delivery machine for search and rescue[7]. While most of the robotic solutions are specialized for one specific approach, we show that ANYmal is capable of performing different missions.

9.8.1 *Industrial Inspection*

The robot can perform visual, thermal, and acoustic inspection with the payload shown in Fig. 9.2. While a detailed description of all tools would go beyond the scope of this paper, we want to outline the specific approach and results for pressure gauge inspection at the ARGOS challenge.

At the competition, inspection targets were marked as checkpoints in the mission description. In order to robustly point the camera at a checkpoint, the robot moves to a precomputed optimal posture at the inspection point and then determines the required pan-tilt angles online depending on the position of the robot and the checkpoint, then the algorithm switches to a tracking mode. To track the checkpoint, we use a particle filter approach. The measurement updates for the particle filter come from analyzing the image with a Histogram Oriented Gradients (HOG) [127] descriptor. For a robust detector, a global HOG descriptor is trained with machine learning, namely the SVMlight library, using about 5000 positive and negative samples. This approach has proven to work very robustly since it does not matter if the pressure gauge has a different dial face. Once the checkpoint is tracked in the image, the camera zooms in until the checkpoint has the optimal size to read.

6 https://youtu.be/2RQDpoQ2vSo
7 https://youtu.be/qrJlMze_xhQ

Since it is not always possible to directly face the camera in front of the checkpoint, the image must be dewarped. Two complementary approaches have been implemented. First, a given front image of the checkpoint is matched with a given example image by looking for Scale-Invariant Feature Transform (SIFT) features [128] that appear in both images. From the matched SIFT features, the homography matrix is computed and used to dewarp the image. This approach may fail when the image of the checkpoint and the example image differ too much such that there are not enough features for image matching. In this case, the algorithm can "manually" dewarp the checkpoint image knowing both the camera orientation and the checkpoint nominal orientation.

For an accurate reading of the pressure gauge, it is important to identify the center of the pressure gauge. A Hough-based circle detection is first applied to estimate the gauge frame. In a second step, a Hough-based line detection is used to find the pointer of the pressure gauge. The algorithm uses the center to filter the lines and to discard the lines that do not belong to the pointer. The mean of the resulting lines gives the estimated pointer. Finally, the actual pressure value is computed from the known scale of the pressure gauge, the center, and the line angle.

9.8.2 *Payload Delivery for Search and Rescue*

The ERL competition required the search for victims in a simulated disaster scenario and the delivery of a medkit. To complete the task, a payload delivery system was designed to securely carry the medkit (Fig. 9.10).

The delivery system is made of a carbon fiber plate which is rigidly attached to the upper part of the main body of ANYmal. The servo installed in the belt mechanism described in Section 9.3.2 can be remotely triggered via software.

This design allows implementing a simple motion procedure to deploy the medkit once the robot arrives at a target location. As illustrated in Fig. 9.11, ANYmal can bend its knees to lower the main body and simultaneously roll to either the left or right side. After the mechanism is triggered, the robot straightens itself back up and the procedure is terminated. The Free Gait [66] Application Program Interface (API) was used to script the package drop behavior.

This mechanism worked reliably throughout all tests and is well suited for single packet delivery. However, if multiple objects need to be delivered, a different mechanism should be applied.

FIGURE 9.10: The payload delivery system, consisting of a carbon fiber plate, a belt mechanism, and a servo motor. A medical kit can be secured to the robot using the belt mechanism. This can be remotely unlatched by triggering the servo motor installed beneath the system.

9.9 CONCLUSIONS

The present paper illustrates the worldwide first attempt at using a versatile legged machine for autonomous inspection on industrial sites. It also outlines our approach in using such a machine for a realistic search and rescue scenario.

In three consecutive ARGOS competitions on a testing site in Pau, France, our team was able to demonstrate the high potential of the proposed solution. While all individual tasks regarding system mobility, navigation, and inspection could be entirely fulfilled, the high system complexity entails several potential sources for failure. Firstly, the fact that neither stopping (blocking) nor disabling the joint motors leads to an immediate stop of a legged robot complicates safety considerations. As a result, we encountered several (uncontrolled) falls during the three competitions, all of which the robot survived. In the future, it is required to further extend the work on smart emergency behaviors such that legged robots can safely operate even in case of critical software or hardware failure. Secondly, the realistic missions unveiled that robust, reliable, and fast terrain perception under harsh conditions is still challenging.

FIGURE 9.11: The payload delivery procedure. From its initial state (top left) the robot starts to tilt sideways (top right) until a certain roll angle is achieved. Afterward, the payload delivery mechanism releases its belt strip to drop the payload (lower left), and the robot straightens back up (lower right).

When the robot was moving fast, it was difficult to create terrain maps without any false positive obstacles due to the distorted scans from the motion, reflections on metal and wet surfaces, and water drops on the laser. Thirdly, despite the superiority in mobility compared to tracked or wheeled vehicles, there is still a lot of potential for improvement of the locomotion skills of legged robots. Once these deficiencies are overcome, we are convinced that legged robots such as ANYmal can find their way into applications like industrial inspection.

Similarly, during the ERL challenge, team ETH Zurich demonstrated the potential of the use of a legged robotic system for payload delivery. Using a GPS RTK receiver to globally localize itself, the robot was able to follow high-level target poses to navigate towards the competition goal in a semi-autonomous fashion. The advances in our control framework enabled ANYmal to locomote robustly in very rough terrain, including walking in shallow seawater, over wet and dry sand, sticks and steep slopes. A novel localization framework was implemented to fuse together different pose estimation sources, i.e., GPS measurements and pose estimates originating from LiDAR scan matching. A custom payload delivery system was developed which could be remotely triggered to carry and drop a medkit once a target pose was reached.

Acknowledgments

This work was supported in part by the Swiss National Science Foundation (SNF) through the National Centre of Competence (NCCR) in Research Robotics.
This work has been conducted as part of ANYmal Research, a community to advance legged robotics.
Thanks to Ludovica Bastianini for the pictures and videos produced during the ERL challenge.

BIBLIOGRAPHY

1. Čapek, K. *Rossum's Universal Robots* (Penguin Books, 2004).
2. Asmiov, I. *Robot Visions* (Ace, 1991).
3. Liston, R. A. & Mosher, R. S. *A Versatile Walking Truck* in *Transportation Engineering Conference* (1968).
4. Raibert, M. H. Hopping in legged systems — Modeling and simulation for the two-dimensional one-legged case. *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-14**, 451 (1984).
5. Raibert, M. H., Brown Jr, H. B. & Chepponis, M. Experiments in balance with a 3D one-legged hopping machine. *The International Journal of Robotics Research* **3**, 75 (1984).
6. Playter, R. R. *Control of a Biped Somersault in 3D* in *IFToMM-jc International Symposium on Theory of Machines and Mechanisms* **2** (1992), 669.
7. Raibert, M., Chepponis, M. & Brown, H. Running on four legs as though they were one. *IEEE Journal on Robotics and Automation* **2**, 70 (1986).
8. Takanishi, A., Ishida, M., Yamazaki, Y. & Kato, I. *Realization of dynamic walking by the biped walking robot WL-10RD* in *Unknown Host Publication Title* (Japan Industrial Robot Assoc, 1985), 459.
9. Vukobratovic, M. & Borovac, B. Zero-Moment Point — Thirty five years of its life. *International Journal of Humanoid Robotics* **01**, 157 (2004).
10. Hirose, M., Haikawa, Y., Takenaka, T. & Hirai, K. *Development of humanoid robot ASIMO* in *IEEE/RSJ Int. Conf. Intell. Robots Syst. – Workshop 2* (2001).
11. Semini, C., Tsagarakis, N. G., Guglielmino, E., Focchi, M., Cannella, F. & Caldwell, D. G. Design of HyQ - a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* **225**, 831 (2011).

12. Focchi, M., del Prete, A., Havoutis, I., Featherstone, R., Caldwell, D. G. & Semini, C. High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots* **41**, 259 (2017).

13. Hutter, M., Gehring, C., Hoepflinger, M., Bloesch, M. & Siegwart, R. Towards combining Speed, Efficiency, Versatility and Robustness in an Autonomous Quadruped. *IEEE Transactions on Robotics (in Press)* (2014).

14. Gehring, C., Bellicoso, C. D., Coros, S., Bloesch, M., Fankhauser, P., Hutter, M. & Siegwart, R. *Dynamic trotting on slopes for quadrupedal robots* in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015), 5129.

15. Fankhauser, P., Bloesch, M., Gehring, C., Hutter, M. & Siegwart, R. Robot-centric elevation mapping with uncertainty estimates (2014).

16. Fankhauser, P., Bjelonic, M., Bellicoso, D., Miki, T. & Hutter, M. Robust Rough-Terrain Locomotion with a Quadrupedal Robot. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 1 (2018).

17. Gehring, C., Coros, S., Hutter, M., Bloesch, M., Hoepflinger, M. A. & Siegwart, R. *Control of dynamic gaits for a quadrupedal robot* in *2013 IEEE International Conference on Robotics and Automation* (IEEE, 2013), 3287.

18. Gehring, C., Coros, S., Hutter, M., Bloesch, M., Fankhauser, P., Hoepflinger, M. A. & Siegwart, R. *Towards Automatic Discovery of Agile Gaits for Quadrupedal Robots* in *IEEE International Conference on Robotics and Automation 2014* (2014).

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

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

21. Gehring, C., Bellicoso, C. D., Fankhauser, P., Coros, S. & Hutter, M. *Quadrupedal locomotion using trajectory optimization and hierarchical whole body control* in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), 4788.

22. Gehring, C., Coros, S., Hutter, M., Bellicoso, C. D., Heijnen, H., Diethelm, R., Bloesch, M., Fankhauser, P., Hwangbo, J., Hoepflinger, M. & Siegwart, R. Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot. *IEEE Robotics Automation Magazine* **23**, 34 (2016).

23. Fankhauser, P. *Perceptive Locomotion for Legged Robots in Rough Terrain* PhD thesis (ETH Zurich, 2018).

24. Fankhauser, P., Bellicoso, C. D., Gehring, C., Dubé, R., Gawel, A. & Hutter, M. *Free gait—An architecture for the versatile control of legged robots* in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)* (2016), 1052.

25. Bjelonic, M., Bellicoso, D., de Viragh, Y., Sako, D. V., Tresoldi, F. D., Jenelten, F. & Hutter, M. Keep Rollin'—Whole-Body Motion Control and Planning for Wheeled Quadrupedal Robots. *IEEE Robotics and Automation Letters* **4**, 2116 (2019).

26. Bellicoso, C. D., Krämer, K., Stäuble, M., Sako, D., Jenelten, F., Bjelonic, M. & Hutter, M. *ALMA - Articulated Locomotion and Manipulation for a Torque-Controllable Robot* in *2019 IEEE International Conference onRobotics and Automation (ICRA)* (IEEE, 2019).

27. *Boston Dynamics* https : / / www . bostondynamics . com. Accessed: 2018-10-28.

28. Mordatch, I., Todorov, E. & Popović, Z. Discovery of Complex Behaviors Through Contact-invariant Optimization. *ACM Trans. Graph.* **31**, 43:1 (2012).

29. Winkler, A. W., Bellicoso, C. D., Hutter, M. & Buchli, J. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters* **3**, 1560 (2018).

30. Mehrotra, S. On the Implementation of a Primal-Dual Interior Point Method. *SIAM Journal on Optimization* **2**, 575 (1992).

31. Nocedal, J. & Wright, S. J. in *Numerical Optimization* chap. 18 (Springer, 2006).

32. Wächter, A. & Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* **106**, 25 (2006).

33. Gill, P., Murray, W. & Saunders, M. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Journal on Optimization* **12**, 979 (2002).

34. Hanafusa, H., Yoshikawa, T. & Nakamura, Y. Analysis and Control of Articulated Robot Arms with Redundancy. *IFAC Proceedings Volumes* **14**. 8th IFAC World Congress on Control Science and Technology for the Progress of Society, Kyoto, Japan, 24-28 August 1981, 1927 (1981).

35. Nakamura, Y., Hanafusa, H. & Yoshikawa, T. Task-Priority Based Redundancy Control of Robot Manipulators. *The International Journal of Robotics Research* **6**, 3 (1987).

36. Siciliano, B. & Slotine, J. J. E. *A general framework for managing multiple tasks in highly redundant robotic systems* in *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on* (1991), 1211.

37. Kanoun, O., Lamiraux, F. & Wieber, P. B. Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. *IEEE Trans. on Robotics* **27**, 785 (2011).

38. De Lasa, M., Mordatch, I. & Hertzmann, A. *Feature-based Locomotion Controllers* in *ACM SIGGRAPH 2010 Papers* (ACM, Los Angeles, California, 2010), 131:1.

39. Herzog, A., Rotella, N., Mason, S., Grimminger, F., Schaal, S. & Righetti, L. Momentum Control with Hierarchical Inverse Dynamics on a Torque-Controlled Humanoid. *arXiv* (2014).

40. Escande, a., Mansard, N. & Wieber, P.-B. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research* **33**, 1006 (2014).

41. McGhee, R. & Frank, A. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences* **3**, 331 (1968).

42. Vukobratovic, M. & Stepanenko, J. On the stability of anthropomorphic systems. *Mathematical Biosciences* **15**, 1 (1972).

43. Sardain, P. & Bessonnet, G. Forces acting on a biped robot. Center of pressure-zero moment point. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* **34**, 630 (2004).

44. Winkler, A. W., Farshidian, F., Pardo, D., Neunert, M. & Buchli, J. Fast Trajectory Optimization for Legged Robots using Vertex-based ZMP Constraints. *IEEE Robotics and Automation Letters (RA-L)* **2**, 2201 (2017).

45. Orin, D. E. & Goswami, A. *Centroidal Momentum Matrix of a humanoid robot: Structure and properties* in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2008), 653.

46. Mistry, M., Buchli, J. & Schaal, S. Inverse dynamics control of floating base systems using orthogonal decomposition. *2010 IEEE Int. Conf. on Robotics and Automation*, 3406 (2010).

47. Sentis, L. & Khatib, O. *A whole-body control framework for humanoids operating in human environments* in *Proceedings 2006 IEEE Int. Conf. on Robotics and Automation, 2006. ICRA 2006.* (2006), 2641.

48. Sentis, L. *Synthesis and Control of Whole-body Behaviors in Humanoid Systems* AAI3281945. PhD thesis (Stanford, CA, USA, 2007).

49. Kanoun, O., Lamiraux, F. & Wieber, P. B. Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task. *IEEE Trans. on Robotics* **27**, 785 (2011).

50. Herzog, A., Rotella, N., Mason, S., Grimminger, F., Schaal, S. & Righetti, L. *Multi-Contact Interaction with Hierarchical Inverse Dynamics and Momentum Trajectory Generation* in *Proceedings of Dynamic Walking* (2015).

51. Siciliano, B., Sciavicco, L., Villani, L. & Oriolo, G. *Robotics: Modelling, Planning and Control* 632 (2009).

52. Gehring, C., Bellicoso, C. D., Coros, S., Bloesch, M., Fankhauser, P., Hutter, M. & Siegwart, R. *Dynamic trotting on slopes for quadrupedal robots* in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2015), 5129.

53. Bloesch, M., Sommer, H., Laidlow, T., Burri, M., Nützi, G., Fankhauser, P., Bellicoso, D., Gehring, C., Leutenegger, S., Hutter, M. & Siegwart, R. A Primer on the Differential Calculus of 3D Orientations. *CoRR* **abs/1606.05285** (2016).

54. Hutter, M., Sommer, H., Gehring, C., Hoepflinger, M., Bloesch, M. & Siegwart, R. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research* **33**, 1047 (2014).

55. Iwamura, M. & Nagao, M. A method for computing the Hessian tensor of loop closing conditions in multibody systems. *Multibody System Dynamics* **30**, 173 (2013).

56. Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C. D., Tsounis, V., Hwangbo, J., Fankhauser, P., Bloesch, M., Diethelm, R. & Bachmann, S. *ANYmal - A Highly Mobile and Dynamic Quadrupedal Robot* in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016).

57. Bloesch, M., Gehring, C., Fankhauser, P., Hutter, M., Hoepflinger, M. A. & Siegwart, R. *State estimation for legged robots on unstable and slippery terrain* in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2013), 6058.

58. Martin Felis. *Rigid Body Dynamics Library* Available at http://rbdl.bitbucket.org/.

59. Featherstone, R. *Rigid Body Dynamics Algorithms* (Springer US, Boston, MA, 2008).

60. Luca Di Gaspero. *QuadProg++* Available at http://quadprog.sourceforge.net/. 1998.

61. Goldfarb, D. & Idnani, A. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming* **27**, 1 (1983).

62. Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M. & Schaal, S. *Fast, robust quadruped locomotion over challenging terrain* in *2010 IEEE International Conference on Robotics and Automation* (IEEE, 2010), 2665.

63. Winkler, A. W., Mastalli, C., Focchi, M., Caldwell, D. G. & Havoutis, I. Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain. *IEEE International Conference on Robotics and Automation*, 5148 (2015).

64. Park, H.-W., Wensing, P. M. & Kim, S. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. *Robotics: Science and Systems* (2015).

65. Bellicoso, C. D., Gehring, C., Hwangbo, J., Fankhauser, P. & Hutter, M. *Perception-less terrain adaptation through whole body control and hierarchical optimization* in *2016 IEEE-RAS 16th Int. Conf. on Humanoid Robots (Humanoids)* (2016), 558.

66. Fankhauser, P., Bellicoso, C. D., Gehring, C., Dubé, R., Gawel, A. & Hutter, M. *Free Gait - An architecture for the versatile control of legged robots* in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)* (2016), 1052.

67. McGhee, R. & Frank, A. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences* **3**, 331 (1968).

68. Herzog, A., Righetti, L., Grimminger, F., Pastor, P. & Schaal, S. *Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics* in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014), 981.

69. Gottschalk, S. *Separating axis theorem* tech. rep. (TR96-024, Department of Computer Science, UNC Chapel Hill, 1996).

70. Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M. & Schaal, S. Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research* **30**, 236 (2010).

71. Englsberger, J., Kozłowski, P., Ott, C. & Albu-Schäffer, A. Biologically Inspired Deadbeat Control for Running: From Human Analysis to Humanoid Control and Back. *IEEE Transactions on Robotics* **32**, 854 (2016).

72. Englsberger, J., Werner, A., Ott, C., Henze, B., Roa, M. A., Garofalo, G., Burger, R., Beyer, A., Eiberger, O., Schmid, K. & Albu-Schäffer, A. *Overview of the torque-controlled humanoid robot TORO* in *2014 IEEE-RAS International Conference on Humanoid Robots* (2014), 916.

73. Bellicoso, C. D., Jenelten, F., Fankhauser, P., Gehring, C., Hwangbo, J. & Hutter, M. *Dynamic Locomotion and Whole-Body Control for Quadrupedal Robots* in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017), Vancouver, Canada, September 24–28, 2017* (IEEE, 2017).

74. Nocedal, J. & Wright, S. J. *Numerical Optimization* 2nd (Springer, New York, 2006).

75. Caron, S., Pham, Q. C. & Nakamura, Y. ZMP Support Areas for Multicontact Mobility Under Frictional Constraints. *IEEE Transactions on Robotics* **33**, 67 (2017).

76. Bellicoso, C. D., Bjelonic, M., Wellhausen, L., Holtmann, K., Günther, F., Tranzatto, M., Fankhauser, P. & Hutter, M. Advances in Real-World Applications for Legged Robots. *accepted for Journal of Field Robotics* (2018).

77. Heppner, G., Buettner, T., Roennau, A. & Dillmann, R. *Versatile - High Power Griper for a Six Legged Walking Robot* in *Mobile Service Robotics* (WORLD SCIENTIFIC, 2014), 461.

78. Rehman, B. U., Calwell, D. G. & Semini, C. *CENTAUR ROBOTS - A SURVEY* in *Human-centric Robotics-Proceedings Of The 20th International Conference Clawar 2017* (2017), 247.

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

80. *Boston Dynamics: Spot* Accessed February 27, 2019. (2015).

81. *Boston Dynamics: SpotMini* Accessed February 27, 2019. (2018).

82. Abe, Y., Stephens, B., Murphy, M. P. & Rizzi, A. A. *Dynamic whole-body robotic manipulation* in *Proc. SPIE 8741, Unmanned Systems Technology* (2013).

83. Righetti, L., Buchli, J., Mistry, M., Kalakrishnan, M. & Schaal, S. Optimal distribution of contact forces with inverse-dynamics control. *The International Journal of Robotics Research* **32**, 280 (2013).

84. Hutter, M., Sommer, H., Gehring, C., Hoepflinger, M., Bloesch, M. & Siegwart, R. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research (IJRR)* **33**, 1062 (2014).

85. Herzog, A., Rotella, N., Mason, S., Grimminger, F., Schaal, S. & Righetti, L. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots* **40**, 473 (2016).

86. Bellicoso, C. D., Jenelten, F., Gehring, C. & Hutter, M. Dynamic Locomotion Through Online Nonlinear Motion Optimization for Quadrupedal Robots. *IEEE Robotics and Automation Letters* **3**, 2261 (2018).

87. Deo, A. S. & Walker, I. D. Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent and Robotic Systems* **14**, 43 (1995).

88. Dietrich, A., Wimbock, T., Albu-Schaffer, A. & Hirzinger, G. Integration of reactive, torque-based self-collision avoidance into a task hierarchy. *IEEE Transactions on Robotics* **28**, 1278 (2012).

89. Campeau-Lecours, A., Lamontagne, H., Latour, S., Fauteux, P., Maheu, V., Boucher, F., Deguire, C. & L'Ecuyer, L.-J. C. Kinova Modular Robot Arms for Service Robotics Applications. *Int. J. Robot. Appl. Technol.* **5**, 49 (2017).

90. Bloesch, M., Burri, M., Sommer, H., Siegwart, R. & Hutter, M. The Two-State Implicit Filter Recursive Estimation for Mobile Robots. *IEEE Robotics and Automation Letters* **3**, 573 (2018).

91. Focchi, M., del Prete, A., Havoutis, I., Featherstone, R., Caldwell, D. G. & Semini, C. High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots* **41**, 259 (2017).

92. Bledt, G., Powell, M. J., Katz, B., Di Carlo, J., Wensing, P. M. & Kim, S. *MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), 2245.

93. Di Carlo, J., Wensing, P. M., Katz, B., Bledt, G. & Kim, S. *Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), 7440.

94. Boston Dynamics. *Boston Dynamics' Atlas Robot Can Do Parkour* https://youtu.be/hSjKoEva5bg. [Online; accessed 20-December-2018]. 2018.

95. Boston Dynamics. *Hey Buddy, Can You Give Me a Hand?* https://youtu.be/fUyU3lKzoio. [Online; accessed 20-December-2018]. 2017.

96. Wieber, P.-B. *On the stability of walking systems* in *Proceedings of the International Workshop on Humanoid and Human Friendly Robotics* (Tsukuba, Japan, 2002).

97. Caron, S., Pham, Q.-C. & Nakamura, Y. ZMP Support Areas for Multi-contact Mobility Under Frictional Constraints. *IEEE Transactions on Robotics* **33**, 67 (2017).

98. Kolvenbach, H., Bellicoso, C. D., Jenelten, F., Wellhausen, L. & Hutter, M. *Efficient Gait Selection for Quadrupedal Robots on the Moon and Mars* in *14th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2018)* (2018).

99. Winkler, A. W. *Optimization-based motion planning for legged robots* PhD thesis (ETH Zurich, 2018).

100. Hutter, M., Diethelm, R., Bachmann, S., Fankhauser, P., Gehring, C., Tsounis, V., Lauber, A., Guenther, F., Bjelonic, M., Isler, L., *et al. Towards a generic solution for inspection of industrial sites* in *Field and Service Robotics* (2018), 575.

101. Kydd, K., Macrez, S. & Pourcel, P. *Autonomous Robot for Gas and Oil Sites* in *SPE Offshore Europe Conference and Exhibition* (Society of Petroleum Engineers, 2015).

102. Pfeiffer, K., Bengel, M. & Bubeck, A. *Offshore robotics - Survey, implementation, outlook* in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011), 241.

103. JPT Staff. Sensabot: A Safe and Cost-Effective Inspection Solution. *Journal of Petroleum Technology* **64**, 32 (2012).

104. Murphy, R. R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H. & Erkmen, A. M. in *Springer Handbook of Robotics* 1151 (Springer, 2008).

105. Davids, A. Urban search and rescue robots: from tragedy to technology. *IEEE Intelligent systems* **17**, 81 (2002).

106. Winfield, A. F., Franco, M. P., Brueggemann, B., Castro, A., Limon, M. C., Ferri, G., Ferreira, F., Liu, X., Petillot, Y., Roning, J., *et al. euRathlon 2015: A multi-domain multi-robot grand challenge for search and rescue robots* in *Conference Towards Autonomous Robotic Systems* (2016), 351.

107. Röning, J., Kauppinen, M., Pitkänen, V., Kemppainen, A. & Tikanmäki, A. The challenge of preparing teams for the European robotics league: Emergency. *Electronic Imaging* **2017**, 22 (2017).

108. Pratt, G. & Manzo, J. The darpa robotics challenge [competitions]. *IEEE Robotics & Automation Magazine* **20**, 10 (2013).

109. Johnson, M., Shrewsbury, B., Bertrand, S., Wu, T., Duran, D., Floyd, M., Abeles, P., Stephen, D., Mertins, N., Lesman, A., *et al.* Team IHMC's lessons learned from the DARPA robotics challenge trials. *Journal of Field Robotics* **32**, 192 (2015).

110. Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P. & Tedrake, R. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots* **40**, 429 (2016).

111. Feng, S., Whitman, E., Xinjilefu, X. & Atkeson, C. G. Optimization-based Full Body Control for the DARPA Robotics Challenge. *Journal of Field Robotics* **32**, 293 (2015).

112. Tsagarakis, N. G., Caldwell, D. G., Negrello, F., Choi, W., Baccelliere, L., Loc, V., Noorden, J., Muratore, L., Margan, A., Cardellino, A., *et al.* WALK-MAN: A High-Performance Humanoid Platform for Realistic Environments. *Journal of Field Robotics* **34**, 1225 (2017).

113. Kaneko, K., Morisawa, M., Kajita, S., Nakaoka, S., Sakaguchi, T., Cisneros, R. & Kanehiro, F. *Humanoid robot HRP-2Kai—Improvement of HRP-2 towards disaster response tasks* in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on* (2015), 132.

114. Zucker, M., Joo, S., Grey, M. X., Rasmussen, C., Huang, E., Stilman, M. & Bobick, A. A General-purpose System for Teleoperation of the DRC-HUBO Humanoid Robot. *Journal of Field Robotics* **32**, 336 (2015).

115. Semini, C., Barasuol, V., Goldsmith, J., Frigerio, M., Focchi, M., Gao, Y. & Caldwell, D. G. Design of the Hydraulically Actuated, Torque-Controlled Quadruped Robot HyQ2Max. *IEEE/ASME Transactions on Mechatronics* (2017).

116. Seok, S., Wang, A., Chuah, M. Y. (, Hyun, D. J., Lee, J., Otten, D. M., Lang, J. H. & Kim, S. Design Principles for Energy-Efficient Legged Locomotion and Implementation on the MIT Cheetah Robot. *IEEE/ASME Transactions on Mechatronics* **20**, 1117 (2015).

117. Park, H.-W., Wensing, P. M. & Kim, S. High-speed bounding with the MIT Cheetah 2: Control design and experiments. *The International Journal of Robotics Research* **36**, 167 (2017).

118. Hutter, M., Gehring, C., Bloesch, M., Hoepflinger, M., Remy, C. D. & Siegwart, R. *StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion* in *Proc. of the Int. Conf. on Climbing and Walking Robots* (2012).

119. Raibert, M., Blankespoor, K., Nelson, G. & Playter, R. Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes* (2008).

120. Hutter, M., Bodie, K., Lauber, A. & Hwangbo, J. *EP16181251 - Joint unit, joint system, robot for manipulation and/or transportation, robotic exoskeleton system and method for manipulation and/or transportation. European Patent Application 16181251.6, 26.07.2016* pat. 2016.

121. Kolvenbach, H. & Hutter, M. *Life Extension: An Autonomous Docking Station for Recharging Quadrupedal Robots* in *Field and Service Robotics* (eds Hutter, M. & Siegwart, R.) (Springer International Publishing, Cham, 2018), 545.

122. Keselman, L., Woodfill, J. I., Grunnet-Jepsen, A. & Bhowmik, A. Intel RealSense Stereoscopic Depth Cameras. *arXiv preprint arXiv:1705.05548* (2017).

123. Fankhauser, P. *Perceptive Locomotion for Legged Robots in Rough Terrain* Doctoral Thesis (ETH Zurich, 2018).

124. Pomerleau, F. *Applied registration for robotics* PhD thesis (ETH, 2013).

125. Bloesch, M., Hutter, M., Hoepflinger, M., Leutenegger, S., Gehring, C., Remy, C. D. & Siegwart, R. *State Estimation for Legged Robots - Consistent Fusion of Leg Kinematics and IMU* in *Robotics Science and Systems (RSS)* (2012), 17.

126. Wermelinger, M., Fankhauser, P., Diethelm, R., Krusi, P., Siegwart, R. & Hutter, M. *Navigation planning for legged robots in challenging terrain* in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2016), 1184.

127. Dalal, N. & Triggs, B. *Histograms of Oriented Gradients for Human Detection* in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition* (2005), 886.

128. Lowe, D. *Object recognition from local scale-invariant features* in *IEEE International Conference on Computer Vision* (IEEE, 1999), 1150.

CURRICULUM VITAE

Dario Bellicoso (born 1983 in Rome, Italy) is a senior PhD for the Robotic Systems Lab led by Prof. Marco Hutter at ETH Zurich, Switzerland. After studying Automation Engineering, he collaborated as a research assistant with Prof. Bruno Siciliano at the Università degli Sudi Federico II, Napoli, Italy, conducting research on modeling and control of a robotic manipulator equipped on an autonomous flying robot. During his doctoral studies at the Robotic Systems Lab, Dario has focused on optimization- and model-based algorithms applied to robotic locomotion and manipulation. He has developed a fast receding-horizon planner and a hierarchical whole-body controller that has been applied to multi-limbed robots.

LIST OF PUBLICATIONS

This section provides a list of publication that were authored and co-authored during this doctoral study. A full list can be found online at https://scholar.google.ch/citations?user=ScazqGMAAAAJ.

Articles in peer-reviewed journals

10. Hutter, M., Gehring, C., Lauber, A., Gunther, F., Bellicoso, C. D., Tsounis, V., Fankhauser, P., Diethelm, R., Bachmann, S., Blösch, M., *et al.* ANYmal-toward legged robots for harsh environments. *Advanced Robotics* **31**, 918 (2017).
11. Neunert, M., Stäuble, M., Giftthaler, M., Bellicoso, C. D., Carius, J., Gehring, C., Hutter, M. & Buchli, J. Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds. *IEEE Robotics and Automation Letters* **3**, 1458 (2018).
12. Bellicoso, C. D., Jenelten, F., Gehring, C. & Hutter, M. Dynamic Locomotion Through Online Nonlinear Motion Optimization for Quadrupedal Robots. *IEEE Robotics and Automation Letters* **3**, 2261 (2018).
13. Winkler, A. W., Bellicoso, C. D., Hutter, M. & Buchli, J. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters* **3**, 1560 (2018).
14. Bjelonic, M., Bellicoso, D., Tiryaki, M. E. & Hutter, M. Skating with a Force Controlled Quadrupedal Robot. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7555 (2018).
15. Bjelonic, M., Bellicoso, D., de Viragh, Y., Sako, D. V., Tresoldi, F. D., Jenelten, F. & Hutter, M. Keep Rollin'—Whole-Body Motion Control and Planning for Wheeled Quadrupedal Robots. *IEEE Robotics and Automation Letters* **4**, 2116 (2019).
16. Bellicoso, C. D., Bjelonic, M., Wellhausen, L., Holtmann, K., Günther, F., Tranzatto, M., Fankhauser, P. & Hutter, M. Advances in real-world applications for legged robots. *Journal of Field Robotics* **35**, 1311 (2018).

Conference contributions

1. Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C. D., Tsounis, V., Hwangbo, J., Bodie, K., Fankhauser, P., Bloesch, M., *et al. Anymal-a highly mobile and dynamic quadrupedal robot* in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), 38.
2. Bellicoso, C. D., Buonocore, L. R., Lippiello, V. & Siciliano, B. *Design, modeling and control of a 5-DoF light-weight robot arm for aerial manipulation* in *Control and Automation (MED), 2015 23th Mediterranean Conference on* (2015), 853.
3. Gehring, C., Bellicoso, C. D., Coros, S., Bloesch, M., Fankhauser, P., Hutter, M. & Siegwart, R. *Dynamic trotting on slopes for quadrupedal robots* in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015), 5129.
4. Hwangbo, J., Bellicoso, C. D., Fankhauser, P. & Hutter, M. *Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics* in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), 3872.
5. Bodie, K., Bellicoso, C. D. & Hutter, M. *ANYpulator: Design and control of a safe robotic arm* in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), 1119.
6. Fankhauser, P., Bellicoso, C. D., Gehring, C., Dubé, R., Gawel, A. & Hutter, M. *Free gait—An architecture for the versatile control of legged robots* in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)* (2016), 1052.
7. Bellicoso, C. D., Gehring, C., Hwangbo, J., Fankhauser, P. & Hutter, M. *Perception-less terrain adaptation through whole body control and hierarchical optimization* in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)* (2016), 558.
8. Gehring, C., Bellicoso, C. D., Fankhauser, P., Coros, S. & Hutter, M. *Quadrupedal locomotion using trajectory optimization and hierarchical whole body control* in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), 4788.
9. Bellicoso, C. D., Jenelten, F., Fankhauser, P., Gehring, C., Hwangbo, J. & Hutter, M. *Dynamic locomotion and whole-body control for quadrupedal robots* in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), 3359.

17. Bellicoso, C. D., Krämer, K., Stäuble, M., Sako, D., Jenelten, F., Bjelonic, M. & Hutter, M. *ALMA - Articulated Locomotion and Manipulation for a Torque-Controllable Robot* in *2019 IEEE International Conference onRobotics and Automation (ICRA)* (IEEE, 2019).