ISTITUTO ITALIANO
DI TECNOLOGIA

Planning and Execution of Dynamic Whole-Body Locomotion on Challenging Terrain

CARLOS MASTALLI

Università degli Studi di Genova
Istituto Italiano di Tecnologia

March 2017

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy (PhD)

Carlos Mastalli:
Planning and Execution of Dynamic Whole-Body Locomotion on Challenging Terrain
Doctor of Philosophy in Bioengineering and Robotics
Genoa, Italy - March 2017

TUTORS:

Prof. Darwin G. Caldwell
Advanced Robotics Department - Director
Istituto Italiano di Tecnologia (IIT)

Dr. Claudio Semini
Dynamic Legged Systems Lab - Head
Advanced Robotics Department
Istituto Italiano di Tecnologia (IIT)

CO-TUTOR:

Dr. Ioannis Havoutis
Robot Learning and Interaction Group - Senior Researcher
Idiap Research Institute

REVIEWERS:

Dr. Olivier Stasse
Laboratory for Analysis and Architecture of Systems - Researcher Director
Centre National de la Recherche Scientifique (CNRS)

Prof. Kris Hauser
Intelligent Motion Lab - Associate Professor
Pratt School of Engineering
Duke University

a Dios por bendecirme,
a mi familia por su amor incondicional,
y al espiritud que me impulsa a ser una mejor persona

Success consists of going from failure to failure without loss of enthusiasm.
Winston Churchill

Abstract

Legged vehicles present a potential advantage over traditional wheeled systems since they offer greater mobility in rough and challenging terrain. However, most legged robots are still confined to structured and flat terrain. One of the main reasons for this is the difficulty in planning complex whole-body motions while taking into account future terrain conditions. Previous research in locomotion focused either on generating reactive behaviors that tackle small terrain changes or planning kinematically foothold locations with relative terrain information. Alternatively legged motion planning approaches focus on synthesizing complex whole-body motions but these do not consider the terrain characteristics.

The goal of this thesis is to close the gap between locomotion approaches and legged motion planning methods. The problem of planning motions for navigating on rough terrain is high-dimensional. For instance, we need to consider the robot's dynamics and the terrain model in a suitable formulation of the planning problem. This thesis addresses these challenges by presenting three different motion planning methods. I initially present a locomotion framework that plans online, and kinematically, the foothold sequences from the terrain costmap and then generates dynamic whole-body motions. For that, I developed a method that builds online and onboard the terrain costmap. Next, I brought the foothold kinematic planning and the dynamic execution closer by proposing a novel trajectory and foothold optimization method. This second method jointly optimizes body motion, step duration and foothold selection while considering terrain topology. Finally, I propose a hierarchical trajectory optimization method that synthesizes dynamic maneuvers by considering the contact forces. This last method can generate a wider range of behaviors by discovering possible contact sequences.

My motion planner methods allow the legged robot to cross various terrains, and to plan highly dynamic motions for complex tasks. Fist, unlike previous work, the locomotion framework can plan online and onboard motions from perceived terrain conditions. It exploits a terrain-aware heuristic function for reducing the computation time. Second, the trajectory and foothold optimization method allows the robot to adapt its walking gait timing while considering terrain topology. To the best of my knowledge, this is the first approach that automatically adapts the walking gait timing for rough terrain locomotion. Finally, the hierarchical trajectory optimization plans behaviors without scheduling a contact sequence. This method ensures the joint torque limits of the robot. My method is the first to have been validated in a real-system.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Genoa, Italy, March 2017

Carlos Mastalli
March 6, 2017

Publications

Journals

- Mastalli, C., Havoutis, I., Focchi M., Caldwell, D. G. and Semini, C. (under-review). Motion planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control. *The International Journal of Robotics Research (IJRR)*.

Conferences

- Winkler, A., Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G. and Semini, C. (2015). Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain. *IEEE International Conference on Robotics and Automation (ICRA)*.
- Mastalli, C., Winkler, A., Havoutis, I., Caldwell, D. G. and Semini, C. (2015). On-line and On-board Planning and Perception for Quadrupedal Locomotion. *IEEE International Conference on Technologies for Practical Robot Applications (TEPRA)*.
- Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G. and Semini, C. (2016). Hierarchical Planning of Dynamic Movements without Scheduled Contact Sequences. *IEEE International Conference on Robotics and Automation (ICRA)*.
- Mastalli, C., Focchi, M., Havoutis, I., Radulescu, A., Calinon, S., Buchli, J., Caldwell, D. G. and Semini, C. (2017). Trajectory and Foothold Optimization using Low-Dimensional Models for Rough Terrain Locomotion. *IEEE International Conference on Robotics and Automation (ICRA)*.

Workshop Posters

- Mastalli, C., Havoutis, I., and Semini, C. (2014). Planning and Control of Whole-Body Motions for Robots with Legs and Arms. *Autonomous Learning Summer School*.
- Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G. and Semini, C. (2015). Dynamic Movements with Hierarchical Optimization. *Workshop on Perception and Planning for Legged Robot Locomotion in Challenging Domains (IROS-15)*.

- Mastalli, C., Havoutis, Radulescu, A., I., Focchi, M., Caldwell, D. G. and Semini, C. (2016). Preview Optimization for Learning Locomotion Policies on Rough Terrain. *Workshop on Dynamic Locomotion and Manipulation (DLMC 2016)*.

Acknowledgments

Before starting my PhD studies I could never imagine how many challenges, both from my professional and personal life, I would face during this period. Definitely without the truth support of many persons (family, friends and colleague) this works could not bring to the end. I learned many lesson from all of you. These fews words are for you.

I am happy with what I accomplished in this PhD thesis. In the first year, Ioannis Havoutis (my supervisor) defined clear goals and together with Alexander Winkler successfully developed a motion planning method. This was the first contribution which enjoyed to see successfully accomplished, it was amazing to make the HyQ robot navigate various challenging terrain conditions. I really thank Ioannis for having well defined goals from the very beginning of my PhD. It definitely helps to re-think about future works! Ioannis also held me to grow as a researcher, he trust in my potentiality and give that suitable degree of freedom to explore as much as wish but without losing the path.

Another important person during this studies was Michele Focchi. He was always ready to help. All the controller developed in this thesis were based on his work and experience in the HyQ robot. He is definitely pretty skilled researcher and engineer, it was a great pleasure to work together with a person with his intensity and curiosity. We always worked up to the point of "sono estasiato", it was not good enough to say that "tutto funziona". Furthermore, I enjoyed with him many other activities (hiking, climbing, barbecue, playing guitar, etc) that was crucial in this period.

Marco Frigerio definitely has to be in this list. He supported the entire team by developing the low-level software of HyQ, he was in the dark side of the moon! He is also a pretty good software developer which help me to make better software. Additionally, Felipe Polido was a key person for the progress of the entire software of the lab. He supported and extended all my ideas to all the different platforms. We worked together towards a "better" software infrastructure, which allowed me (and other colleagues) to move forward.

I will like to thank Claudio Semini and Darwin G. Caldwell, they gave me this wonderful opportunity as PhD student in one of the top lab in the world. In this lab, I had the opportunity to be in tough with worldwide media channels such as: Discovery channel, Reuters, Euro news, etc. I could also exchange ideas in different important events, and more important for me, to share my passion for the robotics with the littlest ones (i.e. kids).

As you might know a PhD is not only about research and papers, it's about the path!. In this path, I discovered the swing dance thanks to Sep Driessen; I got immediately attract to this amazing music. I would also want to thank the Swingin' Genova staff for sharing with me this passion: Marco Agote, Alida Ascari, Mattia

Russolino and Alberto Meucci. And of course, I had the amazing opportunity to share this passion with Romeo Orsolino and Andreea Radulescu (my colleagues and friends).

Last but not least, I would like to thank Eduardo Mastalli and Wilma Cadenas (my parents) for all the endless love, especially when I most needed, and Jose Mastalli and Samuel Mastalli (my brothers) who love and support. Despite the distance and time, they were closer to my heart from the very first second of this period. Finally, I would like to thank Ilaria Rizzuti, she joined me in the last period of my PhD. She brought a light in life, I hope to know always how to make another light in her life.

Contents

List of Figures

List of Tables

Acronyms

HyQ	Hydraulically actuated Quadruped
HyL	Hydraulically actuated Leg
LF	Left-Front
RF	Right-Front
LH	Left-Hind
RH	Right-Hind
HAA	Hip Abduction/Adduction
HFE	Hip Flexion/Extension
KFE	Knee Flexion/Extension
PTU	Pan and Tilt Unit
IMU	Inertial Measurement Unit
DoFs	Degree of Freedoms
CoM	Center of Mass
CoP	Center of Pressure
ZMP	Zero Moment Point
CMP	Centroidal Moment Pivot
GRFs	Ground Reaction Forces
CMM	Centroidal Momentum Matrix
RNEA	Recursive Newton-Euler Algorithm
A*	A star
ARA*	Anytime Repairing A*
MPC	Model Predictive Control
QP	Quadratic Programming
SQP	Sequential Quadratic Programming
FSQP	Feasible Sequential Quadratic Programming

DDP Differential Dynamic Programming

LBFGS Limited-memory Broyden–Fletcher–Goldfarb–Shanno

iLQR iterative Linear Quadratic Regulator

CMA-ES Covariance Matrix Adaptation Evolution Strategy

CHOMP Covariant Hamiltonian Optimization for Motion Planning

CIO Contact Invariant Optimization

LCP Linear Complementarity Programming

MPCC Mathematical Program with Complementarity Constraints

BFP Best First Planning

VM Virtual Model

RL Reinforcement Learning

PI^2 Policy Improvement with Path Integral

DMPs Dynamic Movement Primitives

HAL Hierarchical Apprenticeship Learning

PTSC Prioritized Task-Space Control

PCA Principal Component Analysis

ROS Robot Operating System

RT Real-Time

1

Introduction

Legged locomotion can deliver substantial advantages in unstructured real-world environments as it can offer mobility unmatched by traditional vehicles. Such environments are common in disaster relief, search and rescue, forestry and construction site scenarios. Nevertheless, most legged robots are still confined to structured and flat terrain despite the significant efforts of the research community. These approaches either use reactive behaviors to tackle small terrain changes or plan kinematically foothold locations with relative terrain information. In fact, navigating such environments requires taking into account future terrain conditions (i.e. motion planning). Planning of motion sequences can be split into separate subproblems (*decoupled planning*), i.e. motion and contact planning. Such approaches reduce the combinatorial search space at the expense of the richness of complex behaviors. In contrast, *coupled planning* approaches compute simultaneously contact interactions and body movements. These methods can be posed as hybrid or mode-invariant trajectory optimization problems. In theory, these approaches can synthesize general behaviors, but they suffer from local minima, non-convexity and/or discontinuity.

Motion planning for legged locomotion over rough terrain presents difficult challenges because of the high-dimensionality and complexity of the problem. The main questions that emerge are: (a) how should we model the terrain conditions? (b) what kind of robot model do we need to plan efficient movements? (c) which planning method is more suitable for rough terrain? and (d) how should we formulate these motion planning problems?

Throughout this thesis, I will address these questions. First I develop a decoupled planning approach. This planner computes kinematically, a foothold sequence given a terrain costmap, and then generates a body trajectory from a cart-table model that ensures dynamic stability. Second, I propose a novel coupled planning method that uses a sequence of preview models (i.e. parametrized cart-table models). Here motion, foothold locations and step duration are optimized given a terrain costmap. Finally, I propose a hierarchical trajectory optimization that computes movements with contact forces. This hierarchical trajectory optimization method computes a feasible trajectory using the robot's centroidal dynamics. It ensures that the joint torque limits are satisfied by applying the robot's full dynamics. All these approaches use different robot and terrain models, planning methods, and solvers (i.e. graph search, non-linear optimization, etc). In this opening chapter, I present the main motivations of this thesis in detail. The chapter finishes with a list of main contributions, and an overview of the thesis organization.

Figure 1.1: Dogs have been trained for search and rescue in natural disaster in part because of their ability to navigate such environments. Like their biological counterparts, legged robots can deliver substantial advantages that traditional vehicles cannot offer. Unlike animals and human beings, legged machines can potentially be deployed in dangerous environment, for example the Fukushima Daiichi nuclear power plant.

1.1 Motivation

Crossing rough terrain is the main motivation for the development of legged machines since they can deliver a substantial benefit compared with traditional vehicles. For instance legged machine can potentially navigate in challenging terrains like their dogs (see Fig. 1.1). Nowadays, legged machines are restricted to relatively easy rough terrain conditions. The main reason for this is the difficulty of generating complex dynamic motions that allow them to cross different terrain conditions. Many legged locomotion approaches have focused on the study of reactive behaviors for robot stability without considering the terrain conditions (i.e. "blind" locomotion). A reactive behavior is an instantaneous action that aims to immediately stabilize the robot, i.e. it does not consider a horizon of future events. These approaches can only tackle small changes in the terrain topology, and furthermore, they cannot guarantee the successful accomplishment of the task.

Recently, trajectory optimization with contacts gained much attention in the legged robotics community. It aims to overcome the previously mentioned drawbacks of reactive locomotion approaches by considering a horizon of future events (e.g. body movements and foothold locations). For example, it could potentially improve the robot stability along a specific planning horizon given a certain terrain. In spite of the promising benefits of trajectory optimization for rough terrain locomotion, most of the works focused on flat conditions. Conversely, in rough terrain locomotion, the foothold locations and movements have to be carefully planned.

To ensure locomotion stability, the robot needs to "understand" the environment through a perception system. The *terrain modeling* serves to quantify the terrain difficulty, so that, the robot can plan foothold locations and movements. Note that the terrain model can be used in two ways: for *foothold selection* and for *foothold interaction*. Next, the robot has to evaluate different possible body motions and foothold locations. The *robot modeling* helps to capture the fundamental dynamics, while reducing an unnecessary set of robot behaviors (i.e. the search space). Finally, different *motion planning* methods can be used for evaluating and making decision about the set of possible actions (i.e decoupled or coupled planning). All these elements are addressed in this thesis.

1.1.1 *Terrain modeling*

The terrain condition influences the foothold selection. For example, walking over stepping stones requires selecting footholds along a path of stones. In the legged robotics community few works have focused on terrain modeling for foothold planning [44][40][81][87]. As a common practice, all these works quantify how desirable it is to place a foot at a specific location given a specific cost value. The cost values are computed from a set of geometric features of the terrain, where each feature contributes to an individual cost value. On the other hand, the terrain can also be modeled through constraint functions. In fact, Deits and Tedrake [11] define a set of safe terrain regions using linear constraints. Other approaches describe a set of possible contact interactions by using signed distance constraint functions. However, this kind of terrain models do not provide enough information for foothold selection, just *foothold interaction*. Instead, the terrain costmap uses different geometric features (i.e. slope, curvature, height deviation, etc.) for a more accurate description of the terrain topology. Nonetheless, it cannot fully avoid dangerous terrain regions. In the case of terrain modeling as a costmap, there is a remarkable difficulty in quantifying how desirable a footstep location is. Nevertheless, expert demonstrations can be used to extract associated cost values. A set of high-level expert advice (i.e. body path and footsteps) can be used to infer the weights of the individual cost [44]. Additionally, a set of terrain templates can be learned along an associated set of weights from expert-demonstrated footholds [40].

In the above-mentioned methods, the terrain model has been used just for foothold planning, not for full planning of motions and footholds. The problem of motion and foothold planning can be formulated as a trajectory optimization method. Both terrain representations, either cost or constraint functions, have a strong impact on the solution of these problems. They reshape the landscape of the search space (i.e. non-convexity of the problem), and affect how we can solve it (i.e. gradient-based or stochastic-search optimization). Thus, the terrain model needs to provide sufficiently descriptive information, which also has to be suitable for trajectory optimization.

1.1.2 *Robot modeling*

The robot model allows us to evaluate different possible actions (e.g. body motion and foothold locations). The robot's action can either be evaluated through the geometry of motion or the influence of forces and torques acting on the bodies, i.e. kinematics or dynamics, respectively. In foothold planning, kinematics models have been extensively used for even terrain [34][11] and rough terrain [45][38][81][87]. These approaches use the robot's kinematics for ensuring the reachability of the foothold (e.g. joint position limits). Hence, we can only use the robot kinematics if the body motion and the foothold planning are decoupled. Compared with the dynamic model, the kinematics model has a search space with a lower dimension, but it limits the set of possible robot's actions, such as jumping over a gap. A generalized gait adaptation needs to dynamically balance the body between the gait transitions. In other words, it needs to consider the acting internal forces. Another issues is that the robot's kinematics cannot describe the joint torque limits. These drawbacks stimulated the study of dynamic models in the legged locomotion community.

Dynamic models can address these limitations by describing the system's evolution given the contact configuration and forces (i.e. full dynamic model). Nevertheless, these models are often high-dimensional and nonlinear, which significantly increases the complexity of the search space. Given that, reduced dynamic models have been proposed in the legged robotics community [64][21]. These dynamic models aim to reduce the complexity of the motion planning problem by capturing the legged locomotion dynamics (e.g. point-mass, inverted pendulum, cart-table, etc.) or mapping the full dynamics to the centroidal one (e.g. contact wrench). This dimensionality reduction often helps to describe the trajectory optimization problems in a way that is easier and faster to solve for a specific locomotion behavior. In fact, some of these models describe the trajectory optimization problem as a Quadratic Programming (QP) program [37][38][48], or relaxes the nonlinearities of the problem [9][8][69]. But, these models cannot be used to synthesize a wide variety of behaviors because they capture the dynamics for a specific set of movements. In these cases, the full dynamic model might be used for the motion generation [71].

1.1.3 *Motion planning*

Whole-body motion planning is the process of finding motion and contact sequences. It can be classified as decoupled or coupled planning, depending on the motions and contact sequences are independently or jointly computed, respectively. Decoupled planners are the most studied in rough terrain locomotion because they reduce the problem complexity and computation time [38][45]. The contact locations are found either from a predefined contact sequence or an undefined contact sequence. The planned motion might maintain static or dynamic balance. Nonetheless, decoupled planners cannot synthesize a wide range of dynamic movements such as rearing or gait transitions because we can only plan based on

the robot's kinematics. In contrast, planning simultaneously the contact and motion sequences can potentially synthesize a wide range of behaviors (i.e. coupled planning). But, these approaches often suffer from: (a) non-convexity on account of switching multiple possible dynamics, (b) dynamic discontinuities owing to contact forces. In practice this limits the discovery of new behaviors as we often need to search from a warm-start point (i.e. a suggested behavior). Dealing with these limitations is an open question. For instance, some works aim to overcome these issues by applying stochastic optimization to simple parametric dynamic models [56] or some others propose non-linear optimal control with smooth contact models [60][61][22][63][78]. It is hard to synthesize a wide range of behaviors in an online fashion. However, different approaches have been proposed to reduce the computation time, i.e. by (a) linearizing the non-linear optimal control problem [63][69], or (b) transferring the optimized trajectories to motor policies using machine learning techniques [50][59][57].

Traditionally, motion planners compute a trajectory that connects the actual robot state to a desired goal position. These approaches have been used extensively for computing a sequence of footholds due to the fact that we can easily connect the desired body position with a set of foothold locations, e.g. [45][34]. These planner techniques are often *one-shot* method, i.e. they compute the full plan just once. In contrast, for legged locomotion, it might be more convenient to plan motions in the velocity space. Defining the velocity as user input, it might improve the teleoperation of legged machines. Furthermore, velocity commands are more suitable for receding horizon planning, and as a result, this will increase the robustness of the overall task.

Considering the terrain conditions increases the complexity of the planning problem, as this might increase the number of local minima and/or infeasible regions. In fact, for coupled planning, the terrain model has only considered allowed contact regions (e.g. [60]). These methods can deal with uneven terrain topologies but they neglect important properties such as curvature and slope. In the same way the robot model affects the complexity of the planning problem, too. Simplified models often help to reduce the complexity of the trajectory optimization problem but they limit the range of possible movements. Combining both, terrain and robot model, in a suitable way in terms of computation time and solution results is an exciting new frontier of research. This is the main motivation of this thesis.

1.2 Contributions

To address the points described above, this thesis focused on the development of three different planning techniques for dynamic whole-body locomotion on challenging terrain: (a) decoupled motion and foothold planning, (b) coupled motion and foothold planning and (c) whole-body motion planning with contact forces. These methods are developed primarily for quadrupedal robots but can be extended to other legged systems such as bipeds or hexapods, etc.

1.2.1 *Decoupled motion and foothold planning*

The challenge of decoupled planners lies primarily in increasing the complexity of the generated motion while reducing the computation time. To the best of our knowledge, up to now, decoupled approaches are limited in the versatility of movements and computation time. For instance Kolter et al. [45] and Kalakrishnan et al. [38] reduce the computation time but are still limited to small changes of the robot's yaw (heading). All these previous substantial developments were driven by the DARPA Learning and Locomotion Program with a small quadruped robot called Littledog (a robot that weighs 3 Kg).

Based on those ideas we developed a framework for dynamic legged locomotion over challenging terrain, where the choice of appropriate footholds is crucial for the success of the behavior (see Fig. 1.2) [54][86][31]. First, a hierarchical *motion-before-contact* approach computes an approximated body-path and chooses appropriate footholds from a perception system. This perception system builds online and onboard the terrain costmap. Next, an optimal body trajectory is generated according to a dynamic stability metric to produce fast and natural dynamic whole-body motions. A *floating-base inverse dynamics* controller, in combination with a *virtual model* controller, generates feedback torques to account for tracking inaccuracies. Compared with preliminary foothold planning methods, my method increases the versatility of plans due to the fact that the foothold search regions and sequences are defined accordingly with the body action plan. Furthermore, it computes online and onboard plans (around 0.5 Hz) using the incoming perception information on commodity hardware. The whole-body motion generator can also compute trajectory for different dynamic walking gait patterns.

The main contributions of this work are online terrain costmap building and online foothold planning. The optimized dynamic whole-body motions – despite irregular swing-leg sequences – are generated given a planned foothold sequence. To execute the plan, the controller uses a combination of inverse-dynamics and virtual-model control formulation that exploits the natural partitioning of the robot's dynamic equations. Overall, this framework aims to increase the autonomy, compliance and flexibility of legged robot locomotion in unstructured and challenging environments. To the best of our knowledge, this framework is the first that computes online and onboard foothold sequences and whole-body motions. Note that it has has been published after this work other frameworks that compute online foothold sequences such as in [46]. The details of my perception system and our framework are described in Chapters 3 and 4, respectively. This work has been previously published in [54], and used it to cross various terrain in [86].

1.2.2 *Coupled motion and foothold planning*

Gait adaptability to the terrain requires us to optimize the step duration while considering appropriate foothold locations. To plan these movements, we need to be able to balance dynamically and choose the foothold locations of the robot. This

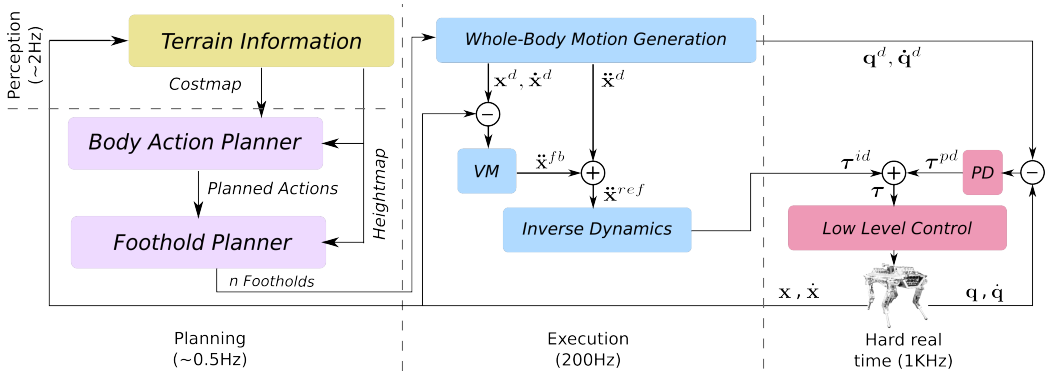


Figure 1.2: Overview of the decoupled motion and foothold planning framework for rough terrain locomotion. The perception and planning processes, on the left of the figure, generate foothold sequences according to the terrain information: terrain costmap and heightmap. The whole-body motion generator, in the middle of the figure, computes a dynamic motion give a sequence of planned footholds. Next, the desired movement is compliantly executed using a combination of feedforward and feedback terms. All the processes are computed online and onboard with their respective frequencies.

means we have to plan motions and footholds at the same time. Thus, I propose a novel trajectory optimization method for legged locomotion over rough terrain [52]. The problem is formulated as a coupled planning of Center of Mass (CoM) motions and footholds, where the foothold locations are selected using the incoming terrain costmap. The trunk height and attitude is planned, in a second step, to cope with different terrain elevations (see Fig. 1.3). First, a sequence of control parameters (the Center of Pressure (CoP) displacement, the phase duration and the foothold locations) is optimized given the terrain costmap. Then, the CoM trajectory and the swing-leg trajectory are jointly generated using a sequence of parametric preview models and the terrain elevation map. To realize the low-dimensional plan, the controller selects appropriate torque commands that are computed by the combination of a trunk controller with a joint-space torque controller.

My trajectory optimization method increases the locomotion capabilities of the previous framework compared with my decoupled planner. With this coupled planner, the robot is able to cross terrain with fewer stepping stone terrain and different elevations. My planning method is based on the following concepts: (a) the terrain costmap to quantify how desirable it is to place a foot at a specific location [54], (b) coupling the CoM planning with the foothold selection, (c) a trunk attitude planner for coping with terrain elevations, (d) CoM planning and foothold selection in the horizontal frame, which effectively decouples it from the trunk attitude planning, (e) robot-centric planning for scaling the preview control actions to different robot poses and (f) the trunk velocities are the high-level user commands for practical teleoperation of the system.

The main contribution of this work is a novel trajectory optimization approach for legged locomotion on rough terrain. In contrast to [56], my method considers terrain topologies (in the form of a terrain costmap) for foothold selection in

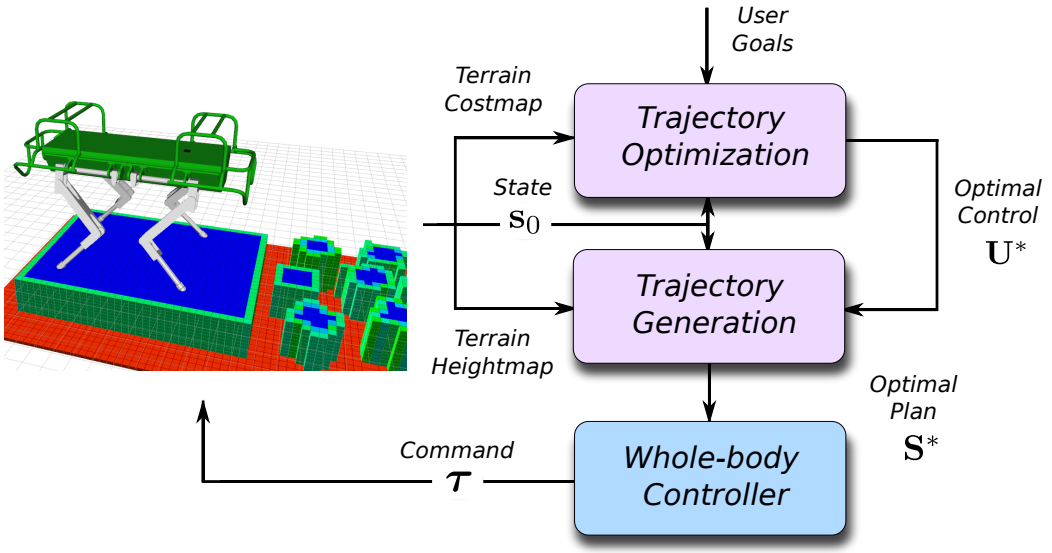


Figure 1.3: Overview of the trajectory optimization framework for locomotion on rough terrain. An optimal control sequence $\mathbf{U}^*$ given the user's goals, the actual state $\mathbf{s}_0$ and the terrain costmap is computed offline. Given this optimal control sequence, the optimal plan $\mathbf{S}^*$ is generated in order to cope with the changes in the terrain elevation through trunk attitude planning. Lastly, the whole-body controller calculates the joint torques τ^* that satisfy friction-cone constraints.

the trajectory optimization. My planner can produce a wide range of different locomotion behaviors despite the use of low-dimensional parametric models. The combination of these models with stochastic-based exploration and receding horizon planning helps us to automatically adapt the walking gait, that increases the versatility of legged robots in rough terrain locomotion compared with decoupled planners. Moreover, the various terrain elevations are tackled by modulating the trunk height and attitude, and planning in the horizontal frame. Additional contributions include trajectory optimization with different terrain costmaps, a suitable terrain description through a cost function and a method for guaranteeing the dynamic stability when the robot adjusts the attitude of its trunk. To the best of our knowledge, this approach is the first that jointly optimizes phase duration and foothold selection *while considering terrain topology*. The details of my coupled planning method are described in Chapter 5. This work has been previously published in [52].

1.2.3 *Whole-body motion planning with contact forces*

Whole-body motions can be planned using a set of simplified dynamic models that capture the relevant locomotion quantities such as CoM and CoP, as in the coupled motion and foothold planning approach. These models reduce significantly the dimensionality of the problem, but they are just representative for a subset of movements (e.g. walking or hopping). However, most animal and human locomotion

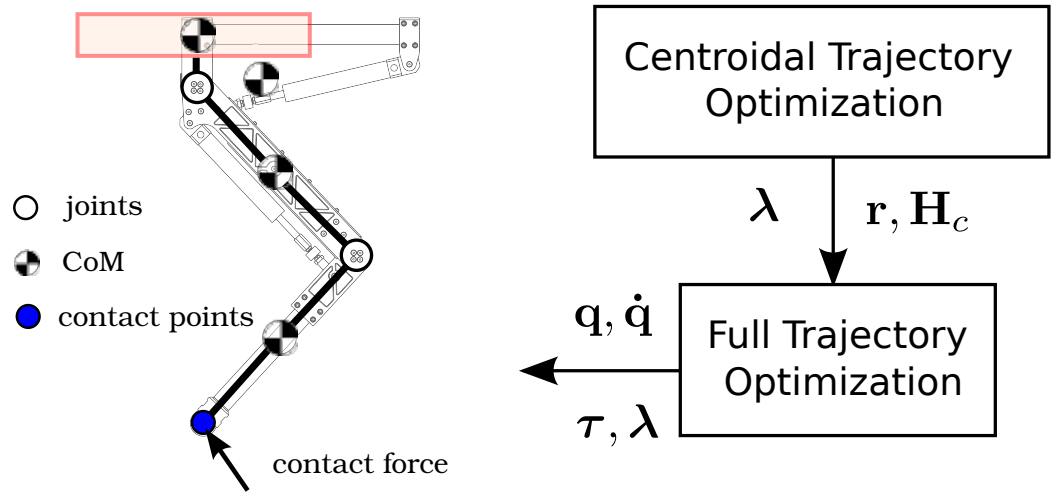


Figure 1.4: Overview of the proposed hierarchical trajectory optimization. This hierarchical trajectory optimization method reduces the complexity of the motion planning problem by considering two different optimization phases: centroidal and full trajectory optimization. First, the centroidal trajectory optimization phase produces a locally optimal CoM motion using the centroidal dynamics model [64], which does not consider joint dynamics (i.e. link's CoM). Second, the full trajectory optimization phase refines the CoM trajectory by applying the robot's full-dynamics and joint limits. Both optimization phases use complementarity constraints to model the contact interactions.

behaviors involve dynamic motions and rich contact interactions. In fact complex maneuvers need to consider dynamic movements and contact forces at the same time. A good example is the jumping task because the contact forces play an important role in predicting the ballistic trajectory of the robot. Nevertheless, these trajectory optimization problems are hard to solve due to the high-dimensional search spaces and very nonlinear constraint functions, such as the contact forces ones. Indeed contact forces abruptly change the behavior of the system, which create discontinuities in the optimization landscape [60][71]. To alleviate these issues, I propose a novel hierarchical trajectory optimization method [53] (see Fig. 1.4). This finds feasible trajectories for tasks that require the exploration of different contact sequences through highly-dynamic movements. We choose a set of jumping tasks as examples, as these highlight the ability to explore the dynamical capabilities of the robot, in order to reach goals that are unreachable in a kinematic manner. I validate my method in 2-DoFs leg called HyL.

The main contribution of this work is a novel hierarchical trajectory optimization approach based on the *divide and conquer* principle (see Fig. 1.4). The hierarchical trajectory optimization can produce a wide range of complex behaviors, without scheduling a contact sequence, by reducing the search space to a fixed sequence of commands. I use the fixed sequence of commands to generate a continuous motion plan. The trajectory optimization approach is posed as a Mathematical Program with Complementarity Constraints (MPCC). First, I prune the search space by finding a feasible trajectory according to the robot's centroidal dynamic. Second, I

impose the robot's joint limits by optimizing with the full dynamic model. The details of my hierarchical trajectory optimization method are described in Chapter 6. This work has been previously published in [53].

1.3 Thesis Outline

In this opening chapter, I introduced the main research topics, motivations and contributions of this work. Each main contribution is presented in an individual chapter (Chapters 4 to 6); these chapters build up the core of this thesis. The rest of this thesis is organized as follows:

Chapter 2 This chapter introduces the state of the art in legged robot motion planning for rough terrain locomotion, summarizing the main works in this field. An overview of the different modules for legged locomotion is described in Section 2.1. Section 2.2 introduces few relevant works focused on rough terrain locomotion. Later, Section 2.3 explains in details the different motion planning methods which are classified in two categories (*decoupled* and *coupled planning*). Finally, Section 2.4 explains how the presented related works are connected with this thesis.

Chapter 3 This chapter describes the common elements used along all the planning methods developed in this thesis, i.e. our robotic platform and perception module for online mapping. Section 3.1 describes the quadrupedal robot platform used for our experiments called HyQ, and its leg called HyL, and introduces the various nomenclature used in quadrupedal robots along this thesis. Section 3.2 explains the terrain perception system developed in this thesis. Our motion planners use the terrain costmap (Section 3.2.1) and the terrain heightmap (Section 3.2.2). Both maps are computed online and onboard from a RGBD (Asus Xtion) camera data given the estimated body position.

Chapter 4 This chapter presents a new framework for dynamic legged locomotion over challenging terrain. A decoupled motion and foothold planning method is proposed. Section 4.1 describes how the overall body-path and appropriate footholds are chosen. Section 4.2 explains how dynamically stable whole-body motions are generated based on an arbitrary footstep sequence. Section 4.3 describes how desired motions are accurately and compliantly executed. Section 4.4 evaluates the performance of our framework in real-world experimental trials and pinpoints the main reasons behind trial-failure that were encountered. Next, the possible factors that can limit the success of the framework are identified (Section 4.5). Section 4.6 summarizes this work and presents ideas and directions for future work.

Chapter 5 This chapter proposes a novel trajectory optimization method for dynamic legged locomotion over rough terrain. The problem is formulated as a trajectory and foothold optimization (i.e. coupled planning of CoM motions

and footholds), in which the terrain topology is considered. Section 5.1 describes how to generate the CoM trajectory from a sequence of parametric preview models. Next, the trajectory optimization algorithm for legged locomotion over rough terrain is described (Section 5.2). Section 5.3 describes how to guarantee friction-cone constraint when a desired motion is accurately and compliantly executed. Section 5.4 evaluates the performance of the trajectory optimization approach in experimental trials, which shows the improvements compared with a *decoupled planning* approach, and Section 5.5 discusses the improvement factors. Finally, Section 5.6 draws the conclusions and presents ideas for future work.

Chapter 6 This chapter proposes a novel hierarchical trajectory optimization of whole-body motions and contact forces. Our method describes the contact model through a set of complementary constraints. Section 6.1 describes the hierarchical trajectory optimization approach proposed for dynamic motion planning. Section 6.3 evaluates the performance of the trajectory optimization approach in real-world experimental trials with a robotic leg. Next, Section 6.4 discusses how the hierarchical trajectory optimization helps to discover different behaviors before Section 6.5 summarizes this work and presents ideas for future work.

Chapter 7 This chapter draws the conclusions and presents ideas for future work regarding of the research presented in this thesis.

2

Related Work

Legged locomotion might require to combine different elements such as planning of motion sequences, discovery and learning new behaviors and reactive control strategies. Thus, we present an overview of the different approaches for planning of motion sequences, learning and discovery of new behaviors and reactive control strategies. Later, we describe in detail a few relevant motion planning techniques and frameworks for rough terrain locomotion. There are different approaches in the literature such as methodical search algorithms, non-linear optimization and reinforcement learning methods.

2.1 Overview

Accident and natural disaster sites require robots that are able to navigate over rough and unstructured terrain, where legged robots, particularly quadruped robots, are the better solution compared to humanoids or wheeled vehicles. Nevertheless, the legged locomotion problem requires careful planning of motion sequences and accurate body control, sometime through highly dynamic phases (i.e. jumping, rearing, etc). The most recent developments in rough terrain locomotion were driven by the DARPA Learning Locomotion Program[1], where LittleDog, a small quadruped robot, was shown to traverse a rough terrain such as in Fig. 2.1. In the same line, the Dynamic Legged Systems (DLS) lab of the Istituto Italiano di Tecnologia has developed a torque-controlled quadruped robot called Hydraulically actuated Quadruped (HyQ), as is shown in Fig. 2.1, which was designed to move over rough terrain and perform highly dynamic tasks such as jumping and running with a variety of gaits.

Planning of Motion Sequences involves making decisions about the contact point sequences and behaviors. The problem could be posed as decoupled or coupled planning, i.e. by independently or jointly computing the CoM motions and contact sequences, respectively. Decoupled planning is the most established method for rough terrain locomotion [39][45][85] due to the fact that it reduces the complexity of the problem by splitting it into a list of sub-problems. These approaches might use optimization techniques [38][11] to achieve robust and fast locomotion, or even methodical search[2] [28][15]. In contrast, there is a new wave of developments focusing on motion synthesis frameworks that plan simultaneously sequences of contacts and behaviors [60][71], which enable more natural locomo-

1 For more information about the DARPA Learning Locomotion Program see [68].
2 Footstep transitions are methodically explored, by combining graph search and single-mode planning.

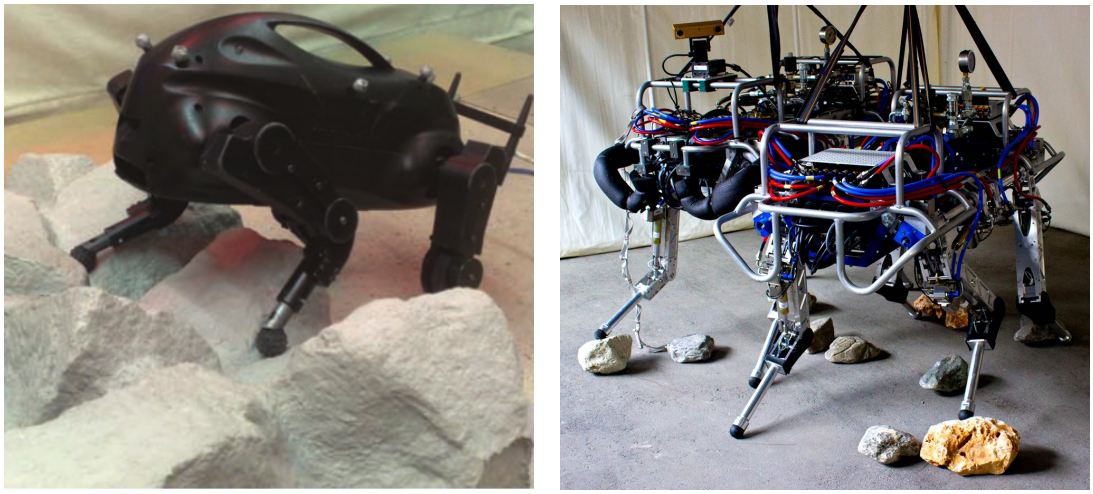


Figure 2.1: *Left*: the LittleDog quadruped robot traversing a rocky terrain [68]. *Right*: the two copies of HyQ robots, fully torque-controlled hydraulically actuated quadruped robots [76].

tion. In theses approaches, the locomotion is stated as complex non-linear optimization problems because of the discontinuous nature of contact forces. There are two trends for motion synthesis, the first ones state it as an optimization problem with switching dynamics and discontinuous state transitions[3] [62][28][15] or with a predefined sequence of locomotion phases [56][33], and the second ones use multi-contact dynamics methods in order to pose as a mode invariant trajectory optimization problem[4] [60][71][13]. Table 2.1 shows the different methods for planning of motion sequences.

Learning and Discovery of New Behaviors can be solved using trajectory optimization in order to guide the policy searching [60][13]. These approaches allow us to easily search for a policy in high-dimensional spaces. They are based on possible inaccurate models of the robot, i.e. they assume a perfect knowledge of the system. In contrast, traditional machine learning approaches such as Reinforcement Learning (RL) compute optimal policies directly on measured data and rewards from interactions with the environment [41][80]. Note that they are not plagued by model inaccuracies. For instance Theodorou et al. [80] propose a stochastic optimal control with path integral, called Policy Improvement with Path Integral (PI^2). This method learns parametrized policies that allows LittleDog to jump over a gap. They use Dynamic Movement Primitives (DMPs)[5] as parametrized policies. DMPs have been proved to be a successful method for policy-based RL algorithms [66][67][42]. Time-dependent Gaussian basis functions can also be used as parametrized policy. In fact, Fankhauser [16] proposed a method that learns

3 This type of system are the so called hybrid ones.
4 It allows simultaneous optimization of contact and behaviors.
5 DMPs are canonical policies or differential equations with well-defined attractor properties that encode discrete and rhythmic motor skills; theses canonical policies were proposed by [36].

Table 2.1: Brief overview of existing motion planning approaches, classified as decoupled and coupled. Both decoupled and coupled approaches are sub-classified in different families: multi-model and hierarchical planning, and hybrid system and mode-invariant planning, respectively.

	Multi-modal planning	**Hierarchical planning**
Decoupled	Hauser [26]	Kolter et al. [45]
	Hauser and Latombe [28]	Kolter et al. [43]
	Hauser et al. [30]	Kalakrishnan et al. [40]
	Escande et al. [14]	Kalakrishnan et al. [39]
	Escande et al. [15]	Winkler et al. [85]
	Hybrid system planning	**Mode-invariant planning**
Coupled	Mordatch et al. [56]	Mordatch et al. [60]
	El Khoury et al. [12]	Tassa et al. [78]
	Nakanishi et al. [62]	Erez et al. [13]
		Posa et al. [71]

long vertical jumps and periodic hopping on ScarlETH[6]. They created a simple policy library that allows to generate jumps towards a desired position by using a linear interpolation between two neighboring learned policies.

Whole-Body Control must be robust against uncertainties and unexpected terrain changes, and at the same time being able to generate accurate motions of the robot. There is a risk of unbalancing the robot with high-gain controllers because this cannot adapt to unexpected contacts. One possible solution consists of increasing the controller compliance by using feedforward torques. For instance they can be computed using the inverse dynamics and predicted contact forces [6][55], but the wrongly predicted contact forces will impact dramatically on the controller tracking performance. In fact, these errors might produce inconsistent desired whole-body trajectories $(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)$ which are required for computing the desired torque commands. Thus, the Prioritized Task-Space Control (PTSC) framework has been proposed to overcome these issues [10][84]. Such task-space controllers provide a more appropriate methodology for planning and learning of whole-body behaviors because they do not suffer from such inconsistencies. For instance, through considerations on the Ground Reaction Forces (GRFs) constraints, generated motions are able to easily adapt to uneven terrain scenarios. PTSC formulations pose it as a conic optimization problem. Other examples compute optimal

6 ScarlETH is a planar one-legged robotic system and consists of a main body/base, a thigh and a shank.

control commands by using a template model[7] [37][47], which might be implemented using the Model Predictive Control (MPC) framework such as [13].

2.2 Legged Locomotion on Rough Terrain

In the literature the problem of legged locomotion over rough terrain has been hierarchically decomposed in different modules. As a common ground, these approaches separate the problem basically into three components: 1) a high-level planner, which plans a sequence of footsteps 2) a low-level planner, which plans a trajectory for the body and 3) a low-level controller, which achieves the desired body and feet trajectories in the face of disturbances. An important advantage of a hierarchical decomposition is that it speeds up computation time. These approaches have proved to work successfully in practice.

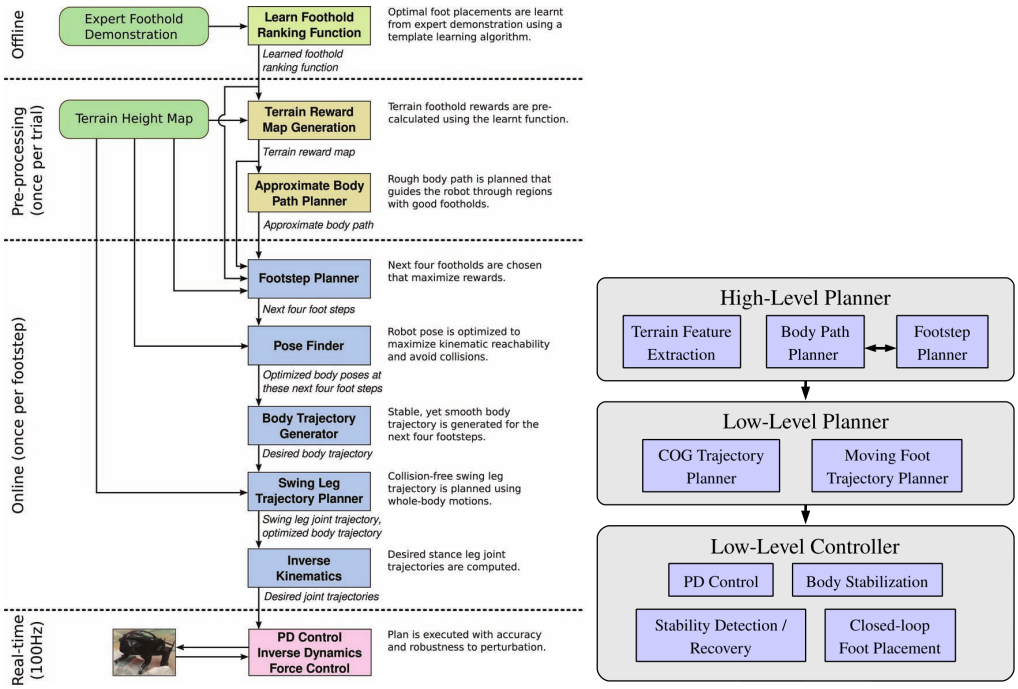
2.2.1 *Overview*

Several works have proposed a general planning and control architecture for fast and robust locomotion on rough terrain. Kolter et al. [45] propose three levels: high-level planner, low-level planner and low-level controller. In the same vein, Kalakrishnan et al. [39] use three main subsystems: 1) body path planner 2) footstep planner and 3) floating-base inverse dynamics controller as is shown in Fig. 2.2. Furthermore, for exploiting the active compliance of the HyQ robot, Winkler et al. [85] developed an adaptive controller that combines a virtual model controller with a floating-base inverse dynamics controller.

In work [45], the high-level planner finds a set of feasible footsteps across the terrain. They use a *foot costmap* that indicates the desirability of stepping at any location in the terrain. For that, it is necessary to build a *body costmap* which averages the foot costs around the default foot locations. An approximated body path is found from the body costmap. This path is computed in a pre-processing step using dynamic programming. Footholds are selected along the approximated body path through a receding horizon branching search. At the low-level planner, the desired trajectory of the robot's CoM is computed for static stability. The static stability projects the actual CoM position into the support triangle. Then, the desired trajectory is calculated by moving the feet in a box pattern (i.e. from a lateral viewpoint). Finally, joint-space PD controllers are used for tracking the desired trajectories.

Kalakrishnan et al. [39] proposed an approach that learns offline a foothold ranking function used for computing the terrain reward map. The robot motion and footholds are computed online from an offline map. The approximated body path is computed from the terrain reward map, which guides the robot through regions with good footholds. Later, the footstep planner computes a set of optimized footholds, for the next four steps, by using greedy search. A fixed pattern of locomotion is used to reduced the search space. For every candidate foothold,

7 A template model captures the fundamental dynamics of the system



(a) (b)

Figure 2.2: An overview of the planning and control architecture for quadruped locomotion over rough terrain: (a) Kalakrishnan et al. [39] and (b) Kolter et al. [45] are two examples of architectures developed for the DARPA Learning and Locomotion Challenge. These architectures decompose the problem in three components: i.e. high-level planner, low-level planner and controller. The images are copied from the aforementioned works.

the body pose generator maximizes the kinematic reachability and to avoid collision with the terrain. The body trajectory generator is formulated as a QP, which uses a desired foothold sequence and the CoP stability condition for computing the CoM trajectory. Finally Covariant Hamiltonian Optimization for Motion Planning (CHOMP) generates a foot trajectory which avoids leg collisions (e.g. shin collision) [88]. CHOMP uses an initial trajectory as seed.

2.2.2 *Terrain modeling*

In rough terrain locomotion, the footholds have to be selected carefully in order to improve the stability and robustness of the task. A terrain model influences the foothold selection process. In fact, the terrain model quantifies how desirable it is to place a foot at a specific location. A set of geometric feature can be used to quantify, e.g. through cost values, the *cross-ability* of a terrain. Furthermore, allowed terrain regions can be described through a set of constraint or high-penalty cost values.

In work [45], the high-level planner computes a set of feasible footsteps from a terrain costmap. The planned footstep sequence aims to be robust against terrain

difficulties, e.g. it prevents slipping and other undesired behaviors. As mentioned above, the terrain costmap is used for computing the *body* and *foot costmaps*. The associated cost values are computed from different geometric features such as slope and curvature. The total cost value is a linear combination of feature cost values, i.e. $g(\mathbf{r}_{ij}) = \mathbf{w}^T \phi(\mathbf{r}_{ij})$ where $\phi(\mathbf{r}_{ij})$ maps the feature to cost values. Nevertheless, one of the major challenges of this approach is to properly tune the set of weights $\mathbf{w}$ that allows the robot to successful cross the terrain. Kolter et al. [43] tackled this problem by inferring the terrain model (or weights $\mathbf{w}$) given a set of high-level expert advices (i.e. body path and footsteps). The authors proposed to use the Hierarchical Apprenticeship Learning (HAL) framework for solving this problem. An important remark about the approach is that the performance depends on the careful design of cost-value functions.

In contrast, Kalakrishnan et al. [40] use expert-demonstrated footholds for learning a set of *terrain templates* along associated set of weights. A *terrain template* is a discretized height map of the terrain around the foothold, and it has an associated feature value[8]. Manual building of a set of templates is time-consuming, and it might be nearly impossible to attain a good generalization performance. Thus, the authors propose a method that can simultaneously learn, from expert-demonstrated footholds, a small set of templates and a foothold ranking function (i.e. the set of weights) per each template. Given a set of reachable footholds $\mathcal{F} = \{f_1 \cdots f_n\}$, an expert foothold demonstration f_c implicitly means that it provides the maximum reward (or minimum cost):

$$\mathbf{w}^T \mathbf{x}_c > \mathbf{w}^T \mathbf{x}_i \quad \forall i \in \mathcal{F}; i \neq n \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the feature vector computed from a selected subset of templates, $\mathbf{w} \in \mathbb{R}^d$ is the weight vector (d number of template in the library), and $R(\mathbf{f}_i) = \mathbf{w}^T \mathbf{x}_i$ is the reward value of a determined foothold $\mathbf{f}_i$. The foothold ranking function learning (i.e. learning of the weight vector $\mathbf{w}$) can be posed as linear classification problem. They compute the weight vector $\mathbf{w}$, together with a selection of the template subset, by promoting the sparse solution in the linear binary classifier (i.e. a l_1-regularized logistic regression) that minimizes the following cost function:

$$J = \sum_{i=1}^{m} -\log\left(\frac{1}{1+\exp(\mathbf{w}^T \mathbf{y})}\right) + \lambda \|\mathbf{w}\|_1 \tag{2}$$

where m is the number of expert demonstrations, λ is the regularization parameter, and $\mathbf{y} \in \mathbb{R}^d; \forall \mathbf{y} \in \{+1, -1\}$ is the classified demonstration. Note that all the demonstrated footholds map to the target class $+1$ of the pairwise difference feature vectors $(\mathbf{x}_c - \mathbf{x}_i)$ (see Eq. (1)). Once the sparse weight vector $\mathbf{w}$ is learned, the templates with zero weight are discarded. The reward values are computed from the feature value of the templates of the library. The feature value represents the similarity between the template and the candidate foothold.

8 The feature vector $\mathbf{x}_i$ is composed of two groups: terrain features, which encode information about the terrain at the foothold, and pose features, which quantify all other relevant information like progress towards the goal, stability margins, and collision margins

2.3 Planning of Motion Sequences

In legged systems, planning of motion sequences can be stated as the process of finding optimal contact interactions and motions. A feasible set of motions depend on the selected contact interactions, and vice-versa; the motion and contact sequences are coupled variables. In fact, a *coupled planning* approach jointly computes the motion and contact sequences. One of the main problems with such approaches is that the search space grows quickly, and the problem might become intractable. There are several efforts to develop computationally efficient algorithms by describing the system as hybrid [56][62][12] (Section 2.3.2.1) or considering the contact forces [60][71][13][78] (Section 2.3.2.2). The main motivation of these techniques is to synthesize more general motions, but they might suffer of local minima problems. On the other hand, decoupling into a set of subproblems (i.e. contact and motion planning) reduces the search space, and as consequence, the computation time. These approaches can be classified as *multi-modal planning* [28][29][15] (Section 2.3.1.1) or *hierarchical planning* [45][39][89] (Section 2.3.1.2). There is two classes of multi-modal planning: *contact-before-motion* and *motion-before-contact*. In addition the hierarchical planning approach assumes a predefined contact sequence (e.g. walking or trotting gaits). So it does not require to explore different contact configuration transitions. This section reviews the state of the art of these methods, and proposes the aforementioned classification.

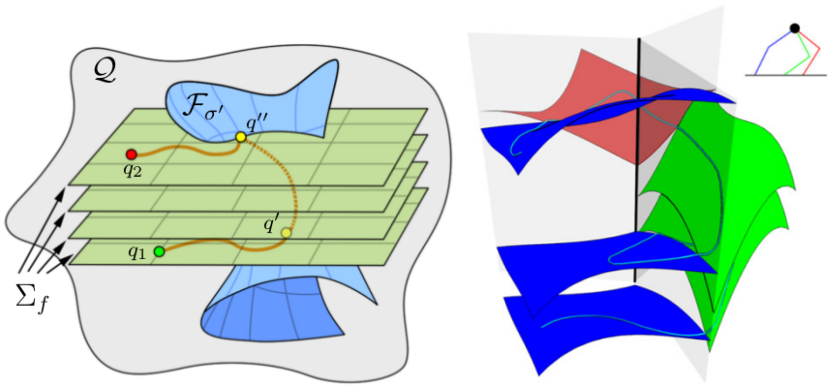
2.3.1 *Decoupled motion and contact planning*

Contacts or impacts comprise of discrete state transitions through continuous motions. Each continuous motion is called a *mode*, where a mode defines a configuration space[9] given a determined contact interaction. The task of finding a sequence of modes towards a specified goal is called multi-modal planning [26]. When there is not a restriction about the choices of possible contacts is called *multi-contact planning* [15]. Different methodical search methods (i.e. search-based planning algorithms) can be used for finding the path between modes. On the other hand, we can exploit the knowledge of a task such as walking by predefining a fixed contact sequence (i.e. foothold sequence). Along this thesis I describe it as hierarchical planning. Because the mode switches are predefined, it allows us to decompose into contact and motion planning. For instance, Kolter et al. [45], Kalakrishnan et al. [39] and Winkler et al. [85] decompose into a high-level planning (i.e. body-path and foothold planning) and a low-level planning (i.e. CoM and foot trajectory generation).

2.3.1.1 *Multi-modal planning*

In multi-modal planning, the configuration space $\mathcal{Q}$ consists of a set of n modes $\Sigma = \{\sigma_1 \cdots \sigma_n\}$. Every mode $\sigma \in \Sigma$ is defined by a set of motion constraints such as the active contact set. A configuration space can be partitioned into a set of

9 In motion planning, the configuration space defines a set of all possible robot configurations [49].



(a) (b)

Figure 2.3: An overview of a multi-modal planning over contact sub-manifold (a) Moving within the same family Σ_f from q_1 to q_2 requires transitioning to a different mode σ' (figure copied from [28]) and (b) These transitions in any contact sequence, or locomotion pattern of a 2D tripod robot (figure copied from [15]).

f disjoint modes $\Sigma_1 \cdots \Sigma_f$ (see Fig. 2.3). Given a goal state q_2, the task of the multi-modal planner is to plan a path through different lower-dimensional sub-manifolds (i.e. a mode manifold $\mathcal{Q}_\sigma$) that connect to the initial state q_1. For moving through different mode sub-manifolds it is required to find the set of *transition configurations*[10] q'. The set of transition configurations define the contact sequence. There are two mode transitioning approaches, *implicit* and *explicit*. Implicit methods generate a path in $\mathcal{Q}_\sigma$, and then check if any transition configuration exist (i.e. *motion-before-contact* planning). Explicit methods sample one or more transition configurations from two adjunct mode sub-manifolds $\mathcal{Q}_\sigma \cap \mathcal{Q}_{\sigma'}$, and plans a single-mode path that connect them (i.e. *contact-before-motion* planning).

Different transition regions are computed given a set of adjacent modes (see Fig. 2.4a,2.4b). Due to the large number of modes, it might be randomly selected a subset of mode transitions for computing the single-mode paths [30]. Moreover, a heuristic function might produce a smaller subset of modes that are likely to contain a path to the goal [28]. Motion primitives can be used as domain knowledge in order to guide the exploration toward promising choices [27]. Note that this selection distributes a sparse number of modes across the configuration space. Once the transition configurations are computed, probabilistic roadmaps can be used to connect them [26] (see Fig. 2.4c). Finally, a set of roadmaps are selected in order to compute a path from the start to the goal states (see Fig. 2.4d).

Escande et al. [15] propose a *multi-contact planning* algorithm that can plan a sequence of movements without restricting the choices of possible contacts, i.e. any part of the robot[11] could be a contact point. This planner belongs to the class

10 If there is a transition configuration between the modes σ and σ', then it is said that σ' is an adjacent mode of σ

11 Regions defined as possible contact areas.

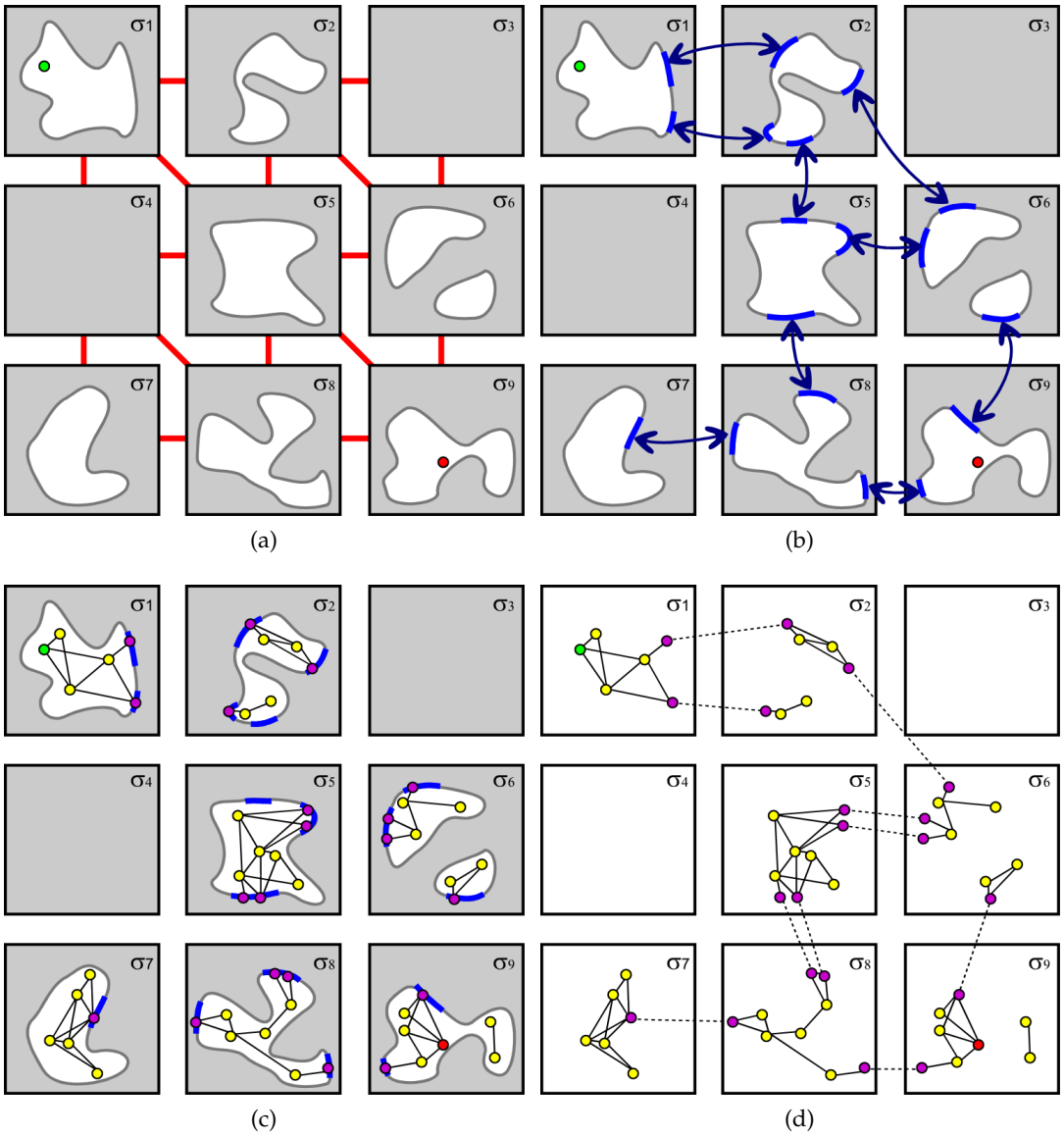


Figure 2.4: Multi-modal planning (copied from [26]); (a) there are 9 modes, where the feasible spaces are white and the start and goal configuration are green and red points, respectively; (b) the transition regions between the adjacent modes are computed; (c) the roadmaps are built in each mode; and (d) the set of roadmaps are connected.

of contact-before-motion strategies. The contact planner spans a tree of possible contacts using a potential-field, this is an adapted version of the Best First Planning (BFP) algorithm. Then, a posture generator checks the feasibility of a certain contact configuration. This problem is posed as a nonlinear optimization program, and solved using the Feasible Sequential Quadratic Programming (FSQP) solver [14].

2.3.1.2 *Hierarchical planning*

Under the assumption of a predefined contact sequence, the hierarchical planning decomposes the problem into two subproblems: motion and contact planning. These assumptions are common in locomotion because we can impose a predefined foothold sequence (or gait), as in works [45][39][85]. This decomposition reduces the search space, and as a consequence the computation time.

HIGH-LEVEL PLANNING The high-level planner computes offline an approximated body path from the terrain costmap. Given the terrain costmap, the body cost-to-go is calculated using the n best foothold locations, and the body attitude cost (i.e. roll and pitch angles). The body cost reflects how desirable is a certain CoM path in terms of the terrain conditions. The cheapest body path can be computed using Dijkstra's algorithm [85], Dynamic Programming algorithm [45] or tree search approaches [73][34].

Once the body path is generated, a set of footsteps is planned along the approximated body path from a greedy search [39][85] or a receding horizon branching search [45]. Both approaches generate an online footstep sequence given a predefined locomotion pattern. For quadrupedal walking gaits, two different pattern of locomotion are successfully used: Left-Hind (LH), Left-Front (LF), Right-Hind (RH), Right-Front (RF) [39][85] or RH, RF, LH, and LF [45]. The foothold selection minimizes the terrain cost [45] or might even minimize the cached terrain cost, body attitude and static stability [39][85]. In addition, Kalakrishnan et al. [39] proposed a pose finder system that optimizes the 6D pose of the robot body $(x, y, z, \mathtt{roll}, \mathtt{pitch}, \mathtt{yaw})$, where the x and y positions are defined according to the stability criterion. The pose finder maximizes the kinematic reachability and avoids feet collisions. Two main approaches exist: the first one is the cyclic coordinate descent search technique, and the second one is a gradient-based optimizer that uses floating-base iterative inverse kinematic algorithm.

LOW-LEVEL PLANNING The low-level planner computes a desired body trajectory and foot trajectories given a foothold sequence. The desired body trajectory generator satisfies the static stability [45][85] or the dynamic one [39]. Kolter et al. [45] propose to use the double support triangle as static criterion, i.e. moving the CoM to the intersection point between the two support triangles and then to the robot's effective center[12]. In work [85], the CoM is moved from the center of the support triangle to the feet average position of the next stance. Both approaches generate the vertical body position based on the height of the feet raised by a

12 The robot's effective center is calculated as the average positions of its four feet, i.e. stance posture.

desired body height. The body pitch is calculated from the support polygon orientation. In contrast to traditional CoM-based approaches (i.e. static stability), Kalakrishnan et al. [39] use the CoP condition[13], as dynamic stability criterion, and a cart-table model. In this work, the step durations are predefined which ends up in QP problem.

The foot trajectory generator has to consider leg collisions, e.g. shin collision. For instance, Kolter et al. [45] and Winkler et al. [85] implemented a trivial approach that generates a swing-movement using a box pattern shape (i.e. from the lateral viewpoint). In contrast, Kalakrishnan et al. [39] compute the trajectory in two phases. First, an initial trajectory is generated between the start and goal position using piecewise quintic splines. This trajectory only guaranteed to avoid collisions in the end-effector. Second, the trajectory is refined using a trajectory optimizer called CHOMP [88].

2.3.2 *Coupled motion and contact planning*

From another standpoint, the motion planning could be solved by simultaneously computing the motion and contact interactions, i.e. coupled planning. It might be described through a hybrid system, i.e. by predefining a sequence of contact switching. There exist two methods for describing the contact switching, i.e. time- or state-based. Nakanishi et al. [62] demonstrated that a time-based hybrid system can be used to tackle the problem of dynamic motion planning in robots with passive compliant actuation. Furthermore, hybrid systems have been also used to synthesize different bipedal locomotion gaits [56]. On the other hand, the contact discontinuities can be described through smooth approximation of the contact dynamics (i.e. contact forces) or complementary constraints [77]. These models are used to describe the robot dynamics in a trajectory optimization problem. For instance, they might be used to stabilize complex behaviors using a real-time MPC in a humanoid robot [13][78], and for motion planning through contact-invariant trajectory optimization [60][22] or contact models as complementary constraints [71][9][22]. Note that the contact forces are part of the decision variables, whereas in hybrid models, contact events are modeled as instantaneous discrete transitions.

2.3.2.1 *Hybrid system planning*

Motion planning with contacts can be posed as a hybrid optimization problem, i.e. by describing the contact switching as state-based or time-based. The contact switching can be modeled as a state-based hybrid system, but it is a nontrivial optimization problem because it has to be posed as multi-point boundary value problem with several interior-point constraints. In fact, Nakanishi et al. [62] suggested to approximate it using a nonlinear time-based switching hybrid dynamic representation, where the multiple phases of the movement are known. The authors achieved highly dynamic motions for a realistic brachiating robot and a hopper. This approach computes an optimal feedback control law $\mathbf{u} = \mathbf{u}(\mathbf{x},t)$ which

13 It also knows as the Zero Moment Point (ZMP) criterion [82].

minimizes the composite cost, and simultaneously optimizes switching instances $t_1, \cdots, t_k$ and the final time t_f as well:

$$J = \phi(\mathbf{x}(t_f)) + \sum_{j=1}^{k} \psi^j(\mathbf{x}(t_j^-)) + \int_{t_0}^{t_f} h(\mathbf{x}, \mathbf{u}) dt \tag{3}$$

where $\phi(\mathbf{x}(t_f))$ is the terminal cost, $\psi^j(\mathbf{x}(t_j^-))$ is the via-point cost at j-th switching instance and $h(\mathbf{x}, \mathbf{u})$ is the running cost. They solve it using an iterative Linear Quadratic Regulator (iLQR) by approximating it with local quadratic model. This optimization problem is subject to a time-based hybrid dynamic model. A time-based hybrid dynamic model can be written as:

$$\begin{aligned} &\dot{\mathbf{x}} = \mathbf{f}_i(\mathbf{x}, \mathbf{u}), && i \in \mathcal{I} = \{1, 2, \cdots, m\} \\ &\mathbf{x}(t_j^+) = \boldsymbol{\Delta}^{i_{j-1}, i_j}(\mathbf{x}(t_j^-)) \end{aligned} \tag{4}$$

where $\mathbf{f}_i : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the i-th subsystem, $\mathbf{x} \in \mathbb{R}^n$ is a state vector, $\mathbf{u} \in \mathbb{R}^m$ is a control input vector and $\mathcal{I}$ is the set of indices for subsystems. They assume an instantaneous discrete (discontinuous) state transition $\boldsymbol{\Delta}^{i_{j-1}, i_j}$ for $j = 1, \cdots, k$, where $\mathbf{x}(t_j^+)$ and $\mathbf{x}(t_j^-)$ denote the post- and pre- transition states, respectively.

2.3.2.2 *Mode-invariant planning*

Above-mentioned techniques treat the discontinuous nature of the problem, which results from contact forces or impacts, as discrete sequence of modes, e.g. by restricting the search for a complete trajectory to a specified sequence of modes of hybrid behaviors. In contrast, mode-invariant planning considers the contact dynamics by smoothing them [60][58][22] and/or describing them as complementary constraints [60][71]. For instance, Mordatch et al. [60] introduced a Contact Invariant Optimization (CIO) method that simultaneously optimizes the contacts and the motions. CIO uses a scalar variable that indicates whether a potential contact should be active in a given phase (mode) of the movement. CIO relaxes the contact constraints which allows contact forces to act at a distance. On the other hand, the contact forces can be posed as complementary constraints such as in the Linear Complementarity Programming (LCP) formulation for multi-contact forward simulation [77]. Posa et al. [71] described the contacts as complementary constraints, i.e. as a MPCC problem[14].

SMOOTHING OF THE CONTACT DYNAMICS Smoothing of the contact dynamics allows us to reduce the complexity and computation time of the optimization problem [78], and even to include a continuation procedure that helps to discover different behaviors [60]. For instance, Erez et al. [13] and Tassa et al. [78] implemented an online MPC machinery based on iLQR[15]. Note that due to the smoothing

14 A MPCC is a class of non-linear optimization problems in which complementary constraints are imposed.

15 iLQR is a variant of the well-known Differential Dynamic Programming (DDP) algorithm with the difference that it only uses the first derivatives of the dynamics.

contact dynamics, they can evaluate very quickly the dynamics and its derivatives, and still predict accurately the robot behavior. On the other hand, Mordatch et al. [60] propose a set of auxiliary decision variables $c_{i,\phi(s)}$ that indicate whether a potential contact should be active in a given phase (i.e CIO), which helps to discover different behaviors. Given a particular initial position of the robot and a goal, CIO computes a sequence of movements and contact interactions for a predefined number of phases k and n end-effector. The optimal solution $\mathbf{s}^* \in \mathbb{R}^{(12(n+1)+n)k}$ (i.e. $\mathbf{s}^* = \begin{bmatrix}\mathbf{x}_{1\cdots k} & \dot{\mathbf{x}}_{1\cdots k} & \mathbf{c}_{1\cdots k}\end{bmatrix}^T$) is computed by minimizing a composite cost function $g(\mathbf{s})$:

$$g(\mathbf{s}) = g_{ci}(\mathbf{s}) + g_{physics}(\mathbf{s}) + g_{task}(\mathbf{s}) + g_{hint}(\mathbf{s}) \tag{5}$$

where g_{ci} is the contact-invariant cost, $g_{physics}$ physics-consistency cost, g_{task} the task cost and g_{hint} a hint cost that shapes the exploration in the early iterations (e.g. dynamic stability hint). As an important mark of the CIO is that it transforms, due to the introduction of the auxiliary variable, a highly discontinuous and local-minima-prone search space into a slightly larger but much better-behaved and continuous search space. The auxiliary variables do not only affect the dynamics (i.e. $g_{physics}$ by enabling and disabling contact forces) but also the contact-invariant cost function:

$$g_{ci}(\mathbf{s}) = \sum_t c_{i,\phi(t)}(\mathbf{s})(\|\mathbf{e}_{i,t}(\mathbf{s})\|^2 + \|\dot{\mathbf{e}}_{i,t}(\mathbf{s})\|^2) \tag{6}$$

where $\mathbf{e}_{i,t}(\mathbf{s})$ is a 4D contact-violation vector for end-effector i at time t. This cost defines that if we have an active contact, a large value of $c_{i,\phi(t)}(\mathbf{s})$, the optimizer has to reduce the contact-violation vector, i.e. the i^{th} end-effector must be touching in the phase $\phi(t)$ at time t. On the other hand, the optimizer could find more convenient that the i^{th} end-effector does not make a contact in the phase $\phi(t)$, thus, it will find large values for $\|\mathbf{e}_{i,t}(\mathbf{s})\|$. But if the auxiliary variable $c_{i,\phi(t)}$ affects also the dynamics (i.e. $g_{physics}$) then the optimizer avoids to set all c's to zero. The physics-violation cost $g_{physics}$ is defined as:

$$J_{physics}(\mathbf{s}) = \sum_t \|\mathbf{J}_t(\mathbf{s})^T\mathbf{f}_t(\mathbf{s}) + \mathbf{B}\mathbf{u}_t(\mathbf{s}) - \boldsymbol{\tau}_t(\mathbf{s})\|^2 \tag{7}$$

where $\mathbf{J}_t(\mathbf{s}) \in \mathbb{R}^{6n\times d}$ is the Jacobian matrix, $\mathbf{f}_t(\mathbf{s}) \in \mathbb{R}^{6n}$ is the vector of the contact forces acting on all n end-effectors, $\mathbf{B} \in \mathbb{R}^{6n\times d}$ is the mapping matrix of applied forces/controls in the actuated space to the full space, $\mathbf{u}_t(\mathbf{s}) \in \mathbb{R}^{6n}$ are the applied forces in the n end-effector and $\boldsymbol{\tau}_t(\mathbf{s})$ denotes the inverse of the smooth dynamics.

The CIO is an unconstrained non-linear optimization problem, which is solved using the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) algorithm. LBFGS is an iterative method for solving unconstrained non-linear optimization problems using quasi-Newton methods.

CONTACT FORCES AS COMPLEMENTARY CONSTRAINTS The robot interaction with the environment causes discontinuities in the dynamics due to the contact forces. The contact forces can be modeled as complementary constraints [77]. For

instance, some multi-contact engines solve the contact constraints using numerical solutions of linear or non-linear complementary problems (LCPs and NCPs). The complementary-based contact model imposes a hard-constraint in the optimization problem, instead of relaxation of contact forces that allow them to act at a distance (e.g. CIO approach). Note that it might generate dynamically infeasible trajectories. These type of constraints represent the combinatorial nature by describing them as orthogonal and positive. For example a contact model can be described as:

$$0 \leqslant \lambda_i^{\hat{n}} \perp \phi_i(\mathbf{q}) \geqslant 0 \tag{8}$$

where $\lambda_i^{\hat{n}}$ is the contact force acting along the surface normal at the i^{th} end-effector (i.e. contact point) and $\phi_i(\mathbf{q})$ is the signed distance between the i^{th} contact point $\mathbf{x}_i$ and the surface $\mathcal{S}_i$ given a joint position $\mathbf{q}$.

The trajectory optimization problem could be solved using direct and shooting methods. A shooting method can be plagued by poorly conditioned gradients, which is why direct methods are widely used for planning in robotic systems. Posa et al. [71] posed it as direct transcription method and included complementary constraints, i.e. the contact forces are part of the decision variables. In the literature, this kind of problems are called MPCC.

$$\min_{h,\mathbf{q}[k],\mathbf{u}[k],\boldsymbol{\lambda}[k]} \phi(\mathbf{q}[k]) + h \sum_{k=1}^{N} \mathcal{L}(\mathbf{q}[k-1], \boldsymbol{\tau}[k], \boldsymbol{\lambda}[k]) \tag{9}$$

which is subject to full dynamic constraints and contact model constraints. They transcribe the full dynamic constraint by applying an Euler-backward integration rule.

2.4 Summary

In this chapter, we present an overview of the state of the art in motion planning for legged systems. We classify the motion planners as decoupled and coupled ones. At the same time, we sub-classify the decoupled and coupled planning in terms of how they are formulated. The decoupled planners use kinematic models for selecting contact interactions, but we can plan dynamic motions in the hierarchical planning family. On the other hand, the coupled planners compute contact and motion using dynamic models. This allows us to increase the capabilities of the robot by planning dynamic motions. The hybrid system planners reduce the computation time, and complexity of the problem, but they are still limited to a certain kind of movements (e.g. walking gait). Additionally, we present an overview of the different rough terrain locomotion frameworks. In contrast to general planning methods, these frameworks consider the terrain topology for foothold planning. Finally, different robot models have been studied along all these planning techniques.

Based on these preliminary works, in this thesis we develop three different planning methods: one decoupled and two coupled ones. We use different robot dynamic models: cart-table, contact wrench and full dynamics; in addition to the

kinematic models. Furthermore, we formulate the motion planning problem in different ways as: graph search, quadratic optimization, non-linear optimization and stochastic optimization. Finally, we developed a perception system that builds an online terrain costmap. We use this terrain costmap in different planning techniques.

3

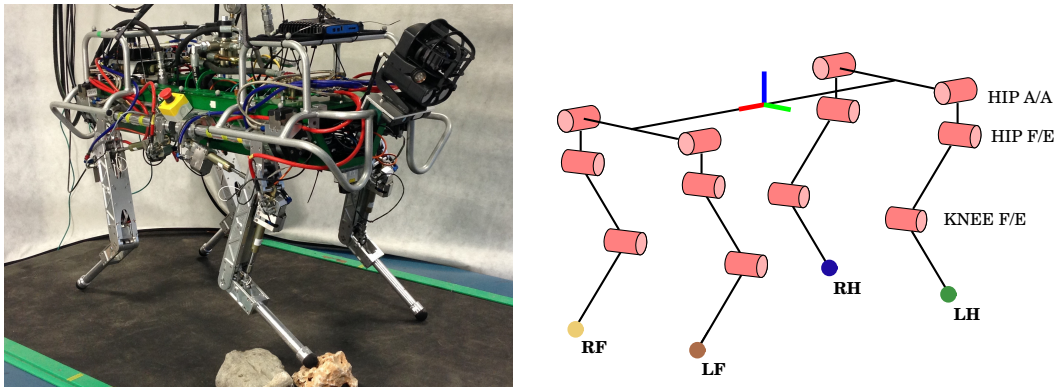
Robotic System and Perception Module

This chapter describes the HyQ and the HyL robots; the platforms used in the experiments of this thesis. Furthermore, it describes the perception system that we developed for rough terrain locomotion. This system builds a terrain *costmap* and *heightmap* from the incoming point-cloud data on commodity sensor. In this thesis, we use the RGBD (Asus Xtion) camera. The terrain costmap and heightmap are common components between the different motion planning methods developed in this thesis.

3.1 HyQ

HyQ is a hydraulically actuated quadruped robot developed by the Dynamic Legged Systems (DLS) Lab at the Istituto Italiano di Tecnologia (IIT) [76][75]. HyQ has been designed to perform highly-dynamic motions such as trotting and jumping, as well as, to carefully navigate over rough terrain. Different potential applications are targeted by this robot such as search and rescue, forestry technology, and construction. The robot is fully torque-controlled which enables to actively control the compliance [5][4]. It roughly has the dimensions of a goat, i.e. 1.0 m×0.5 m×0.98 m (length × width × height). The leg length ranges from 0.339-0.789 m and the hip-to-hip distance is 0.75 m. HyQ's weight is approximately 85 kg, it varies depending on the number of onboard computers and exteroceptive sensors such as cameras and lasers. The Left-Front (LF), Right-Front (RF), Left-Hind (LH) and Right-Hind (RH) legs are represented as brown, yellow, green and blue, respectively (see Fig. 3.1b).

The robot is equipped with 12 active Degree of Freedoms (DoFs). Each leg has three joints: Hip Abduction/Adduction (HAA), Hip Flexion/Extension (HFE) and Knee Flexion/Extension (KFE). The HyQ robot is actuated by 8 hydraulic cylinders (HFE and KFE) and 4 hydraulic rotary motors (HAA), which are driven by high performance servo-valves (bandwidth around 250 Hz). At every piston rod ends, there are load-cells that measure the forces of the pistons, and as result the joint torques can be computed through the mechanism kinematics (Fig. 3.1a). Similarly, a custom torque sensor provides direct measurement of the HAA torques. All the joints are equipped with high-precision encoders: relative and absolute encoders. For all the experiments presented in this thesis, HyQ has two onboard computers: a Pentium i5 with Real-Time (RT) Linux (Xenomai) patch, and a Pentium i5 with Linux. The PC with Xenomai processes the low-level controller (hydraulic-actuator controller) at 1 KHz which communicates with the proprioceptive sensors through EtherCAD boards. Additionally, this PC runs the high-level controller (robot controller) at 250 Hz. Both RT threads (i.e. low- and high- level controllers) communicate through shared memory. The non-RT PC processes the extereoceptive sensors



(a) (b)

Figure 3.1: HyQ: the Hydraulic actuated Quadruped robot. The HyQ robot is actuated by 8 hydraulic cylinders (hip flexion/extension and knee flexion/extension) and 4 hydraulic rotary motors (hip abduction/adduction). (a) A picture of HyQ with the Asus Xtion camera and the MultiSense SL sensor developed at Carnegie Robotics. (b) Kinematic structure of the HyQ robot. The color of the legs are brown, yellow, green, blue for the Left-Front (LF), Right-Front (RF), Left-Hind (LH) and Right-Hind (RH) legs, respectively. Each leg has three joints Hip Abduction/Adduction (HAA), Hip Flexion/Extension (HFE) and Knee Flexion/Extension (KFE). Note that we use the RGB color convention for drawing the base frame.

for generating the map and then computing the plans. Later, the motion plan is sent to the high-level controller through a RT-friendly communication (i.e. using Robot Operating System (ROS) as interprocess communication). The characteristics and features of the HyQ robot are summarized in table 3.1.

3.1.1 *HyL*

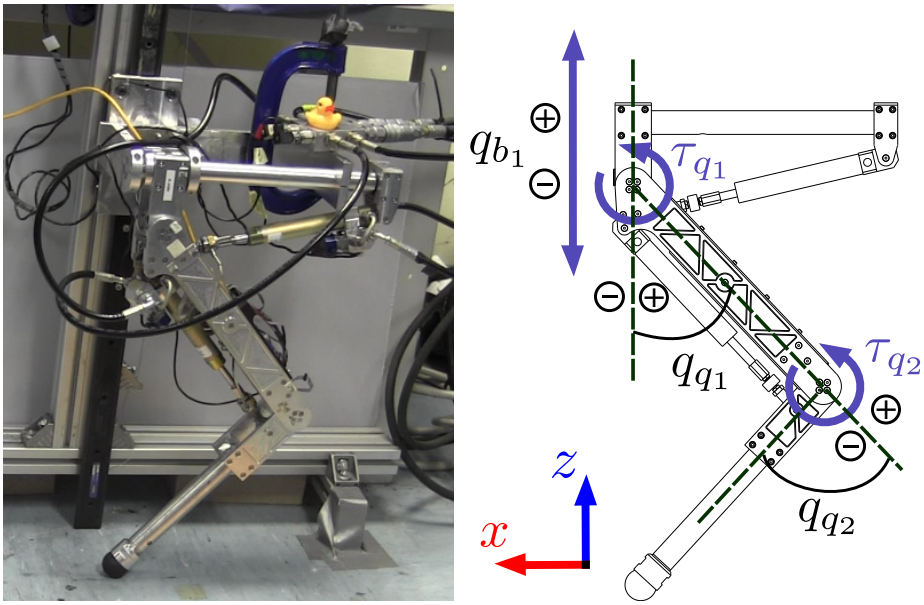
The Hydraulically actuated Leg (HyL) weighs approximately 11 kg. HyL is a 1D floating-base system with 2 actuated joints. The HFE and KFE joints are the two actuated ones, and the base is constrained to move on a vertical slider, as is shown in Fig. 3.2b. Similarly to HyQ, the actuated joints are equipped with precision encoders and load-cells. Moreover, the base is equipped with a relative encoder that allows us to estimated the base position. The leg is controlled with two controllers that run in a real-time PC as HyQ.

3.2 Perception

This section describes the pipeline of acquisition and evaluation of terrain information. We implemented an onboard terrain information server that holds and continuously updates the state of the environment. We show how this information is processed and transformed to a qualitative metric of the terrain geometry.

Table 3.1: System overview of the HyQ robot

dimensions	1.0 m×0.5 m×0.98 m (LxWxH)
link lengths & weights	hip (HAA-HFE): 0.08 m, 2.9 kg
	upper leg (HFE-KFE): 0.35 m, 2.6 kg
	lower leg (KFE foot): 0.35 m, 0.8 kg
weight	85 kg
active DoFs	12
HAA actuators	double-vane rotary hydraulic actuators
HFE/KFE actuators	asymmetric hyd. cylinders with hinge joint
joint motion range	90° (HAA), 120° (HFE, KFE)
max. torque [HAA]	120 Nm (peak torque at 20 MPa)
max. torque [HFE/KFE]	181 Nm (peak torque at 20 MPa)
position sensors	position 80000 cpr in all joints
torque sensors	custom torque (HAA), loadcell (HFE, KFE)
onboard computer	Pentium i5 with real-time Linux (Xenomai)
controller rate	1 kHz



(a) (b)

Figure 3.2: HyL: a single hydraulically-actuated and fully torque-controlled leg of the quadruped robot HyQ [76]. The HyL robot has a total number of 3 DoFs: 1D floating-base system vertically along the slider q_{b_1} with 2 actuated joints (q_{q_1}, q_{q_2}).

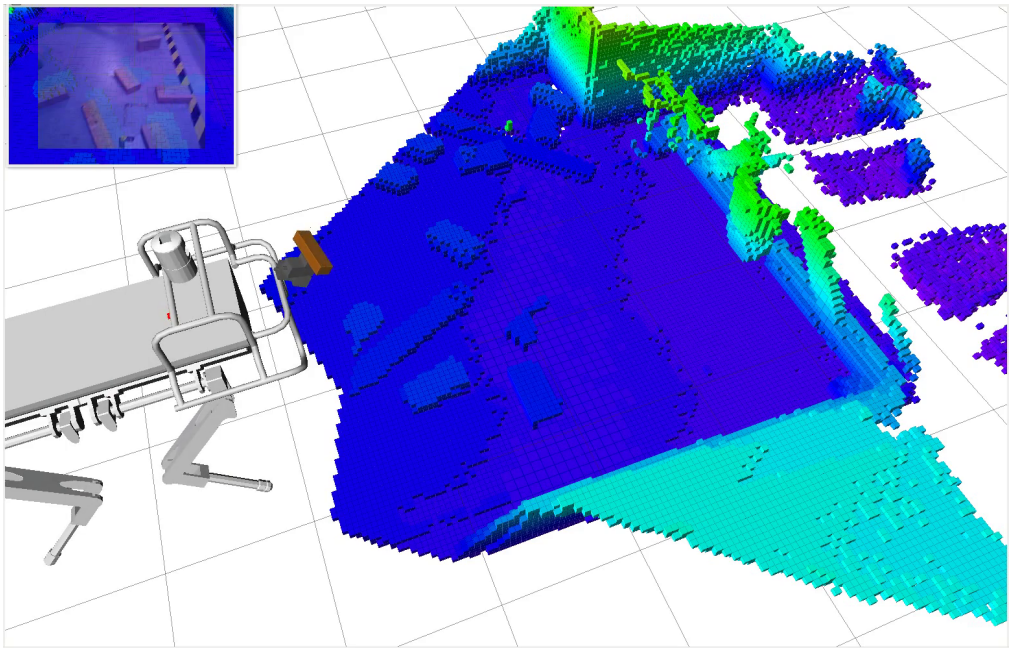


Figure 3.3: The HyQ robot mapping the terrain using Octomap [35]. The voxel map is generated from RGBD (Asus Xtion) camera data given the estimated body position. The RBGD sensor is mounted on a PTU that scans the terrain. The body position is estimated with an Extended Kalman Filter that integrates the leg kinematics with an IMU sensor [2]. The occupancy map is built with a 2 cm resolution.

We use a terrain information server to hold the geometric perception data and simplify the access to terrain information queries. The required information for the motion planning, e.g. the *terrain costmap* and the *terrain heightmap* of the environment around the robot, is computed based on the information that this server holds. We use an RGBD sensor (Asus Xtion) mounted on a scanning Pan and Tilt Unit (PTU) at the front of the robot to produce raw point cloud data. This data is subsequently voxelized and stored in an efficient tree-based data structure, as an Octomap occupancy grid [35]. This provides a probabilistic representation that handles sensor noise and represents both free and occupied space. Octomap uses a hierarchical data structure, for spatial subdivision in 3D, called *octrees*. An octree-based 3D map representation is designed to allow for efficient map updates and for copying (for more details see [35]). This multi-resolution volumetric representation is used for speeding up the computation time of the geometric features. Fig. 3.3 shows the HyQ robot mapping the terrain using Octomap. We use a 2 cm resolution occupancy map because we found out that it is sufficient for the estimation of the features values, and it allows us to build online terrain costmaps (Section 3.2.1).

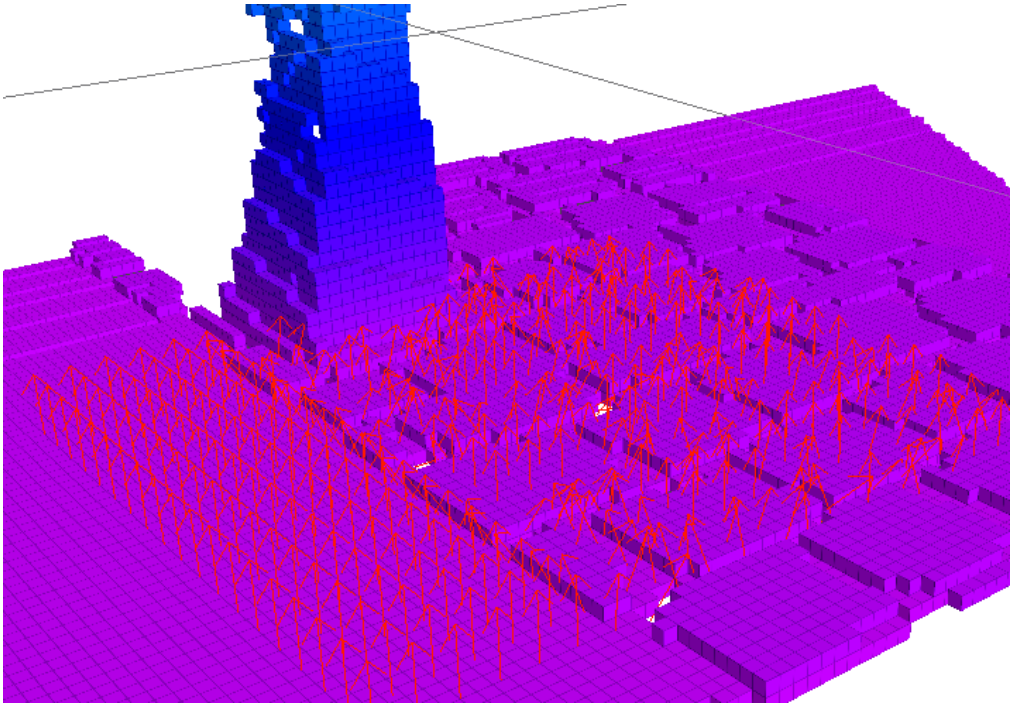


Figure 3.4: A set of surface normals are extracted from the RBGD sensor. The surface normals are estimated from the eigenvalues and eigenvectors computed from the nearest neighbors of a query point.

3.2.1 *Terrain costmap*

The terrain costmap quantifies how desirable it is to place a foot at a specific location. The cost value for each voxel in the map is computed using geometric terrain features as in [85]. Namely we use the *standard deviation of height values*, the *slope* and the *curvature* as computed through regression in a 6 cm×6 cm window around the cell in question; the feature are computed from a voxel model (2 cm resolution) of the terrain. For instance, the estimating surface normals and curvatures are computed from a set of neighboring occupied voxels. Though many different normal estimation methods exist, the surface normal can be estimated through an analysis of the eigenvector and eigenvalues, also known as Principal Component Analysis (PCA), of the set of nearest neighbors (for more information, including the mathematical equations of the least-squares problem, see [74]). Note that the surface normal is computed from the eigenvector that has the smallest eigenvalue λ_0, and the surface curvature σ as follows:

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{10}$$

Fig. 3.4 shows a set of estimated surface normals from the occupancy map of a cobblestone terrain. We compute the normals, and other geometric features, with 4 cm of resolution (i.e. costmap resolution). Note that the terrain costmap uses a different discretization resolution of the occupancy map. The terrain costmap

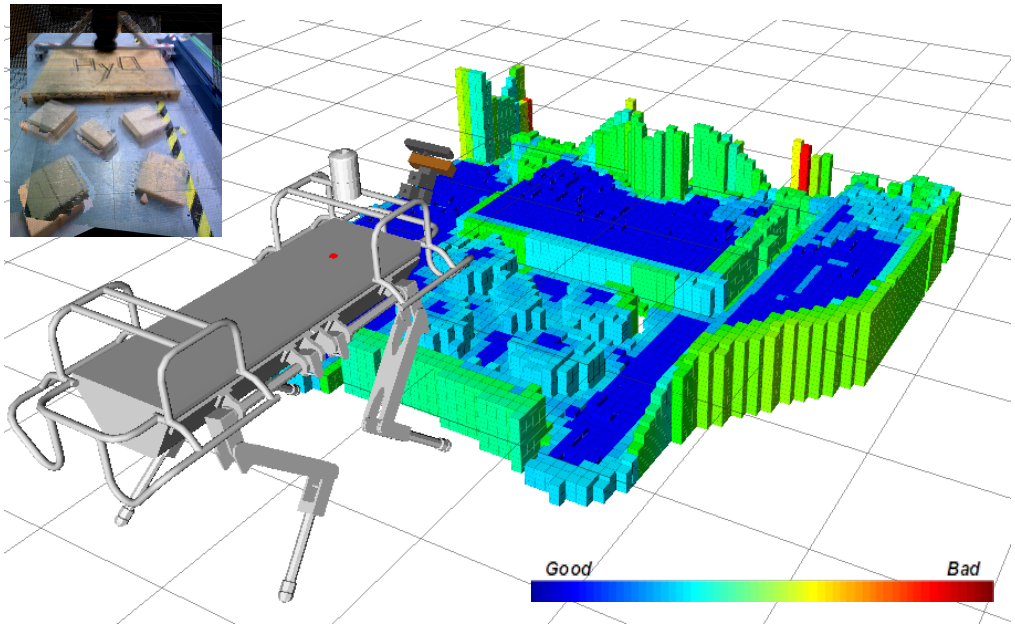


Figure 3.5: The costmap generation from RGBD (Asus Xtion) camera data. The RBGD sensor is mounted on a PTU that scans the terrain. An occupancy map is built with a 2 cm resolution. Then the set of features is computed and the total cost value per pixel is calculated. In addition, a heightmap is created with a resolution of 2 cm in z. The cost values are represented using a color scale, where blue is the minimum cost and red is the maximum one.

is incrementally built based on the aforementioned features and updated locally whenever a change in the map is detected. For computing the terrain costmap, we define an area of interest around the robot of 2.5 m×5.5 m that uses a cell grid resolution of 4 cm. The cost value for each pixel of the map is computed as a weighted linear combination of the individual feature $T(x, y) = \mathbf{w}^T \mathbf{T}(x, y)$, where $\mathbf{w}$ and $\mathbf{T}(x, y)$ are the weights and feature values, respectively. Fig. 3.5 shows the generation of the costmap from the onboard RGBD sensor. The cost values are represented using a color scale, where blue is the minimum cost and red is the maximum one.

3.2.2 *Terrain heightmap*

The terrain heightmap allows the robot to estimate the vertical component (i.e. z-direction) of the footstep. Indeed the body/swing trajectory can be approximately planned to place the foot at the correct height. A lower resolution of the heightmap helps the robot to properly establish a footstep, which improves the overall execution. This feature is particularly important when the contact forces are estimated through the joint torque sensors, as the HyQ robot does. The heightmap is computed using the same resolution (4 cm) as the costmap in the (x, y)-plane. Along the z-direction we desire higher accuracy to step safely onto obstacles, so we use a resolution of 2 cm. In essence, the heightmap is a $2^1/_2$-dimensional *projection* of

the costmap that can be more efficiently handled by the subsequent steps of the motion and foothold planning. For instance, the terrain costmap associates the cost value to the higher occupied voxel.

4

Decoupled Motion and Foothold Planning

Natural locomotion over rough terrain requires simultaneous computation of footstep sequences, body movements and locomotion behaviors, i.e. coupled planning. In this chapter, we split the planning and control problem into a set of subproblems, following a decoupled planning strategy. First, we plan a sequence of footholds by planning an approximate body path (Section 4.1). The approximate body path is computed from a sequence of planned body actions (Section 4.1.1). Then we choose locally the footholds locations (Section 4.1.2). Second, we generate a body trajectory that ensures dynamic stability and achieve the planned foothold sequence (Section 4.2). We achieve a compliance execution by combining a virtual model with a floating-base inverse dynamic controllers (Section 4.3).

In the literature there are few coupled planner methods [79][60][71][9]. One of the main problems with such approaches is that the search space grows quickly and searching becomes unfeasible, especially for systems that need solutions in real-time. In contrast, we use a lattice representation (i.e. body movement primitives) for planning foothold sequence. Our hierarchical foothold planner increases the versatility of the body movements (e.g. discretized yaw–changing movements instead of continuous changes) compared to state-of-the-art approaches [45][81][40]. We reduce the computation time using a terrain-aware heuristic function which estimates the cost-to-go. We generate a CoM trajectory that ensures dynamic stability and needs no knowledge of a predefined gait. For that, it checks if the next swing leg is diagonally opposite of the current swing leg. If so, the disjoint support triangles require a four-leg support phase for the optimization to find a solution. Both, the foothold and motion planners, increase the versatility of the planned motions compared to previous works. Furthermore, our framework builds online the terrain costmap and computes online the foothold sequence. Fig. 4.1 shows an overview of our perception, planning and control framework for dynamic legged locomotion over rough terrain.

All the material presented in this chapter has been previously published in different works. The main contribution of this thesis is a hierarchical foothold planning method [53], which is used in [86]. We explained our motion planning and execution in [86]. Finally, we presented our full framework for dynamic legged locomotion over challenging terrain in work under-review [31].

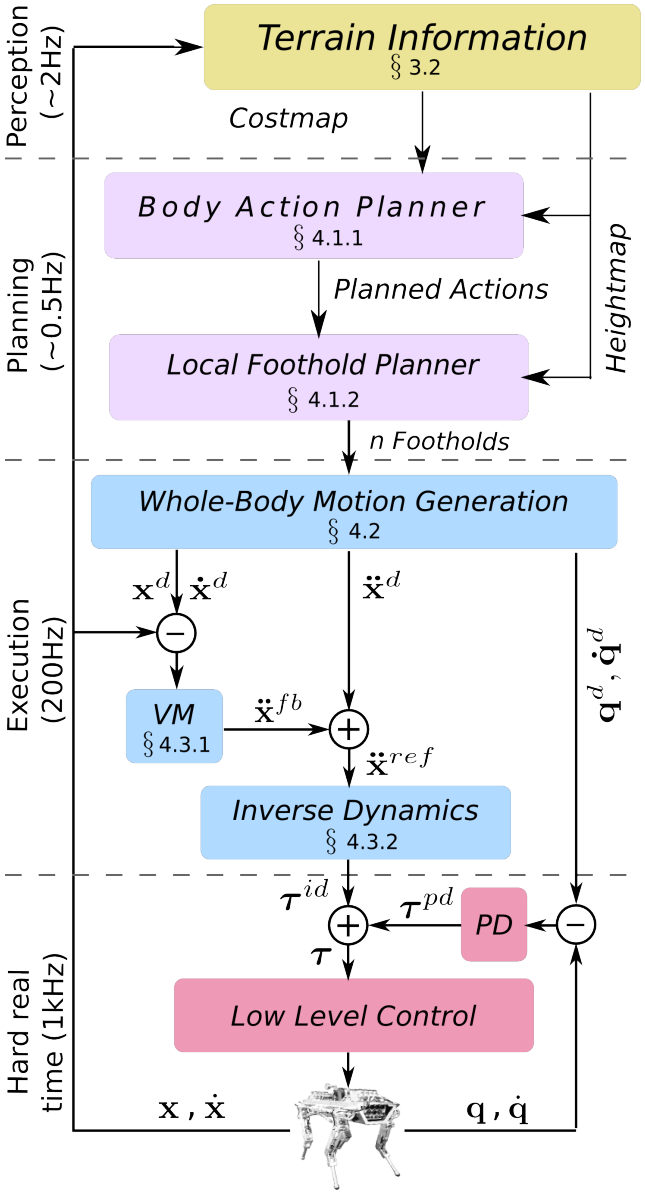


Figure 4.1: An overview of our perception, planning and control framework. The perception and planning processes, on top of the figure, generate foothold sequences according to the terrain information [31]. The next level uses planned footholds to generate dynamic whole-body motions and compliantly executes them using a combination of feedforward and feedback terms.

4.1 Foothold Planning

This section describes the motion planner, consisting of two main elements: the *body action planner* that decides the general direction of movement, and the *foothold sequence planner* that chooses specific footholds. Finally, we describe the (re-)planning and costmap updating process during execution.

The overall task is to plan online an appropriate sequence of footholds $\mathbf{F}$ that allows a quadruped robot to traverse a challenging terrain toward a body goal state (x, y, θ). To accomplish this, the motion planning problem is decoupled into: body action and foothold sequence planning. The body action planner searches a bounded sub-optimal solution around a growing *body-state graph*. Then, the foothold sequence planner selects a foothold around an action-specific foothold region of each planned body action.

Decoupled approaches reduce the combinatorial search space at the expense of locomotion capabilities. State-of-the-art approaches produce plans that are limited by the "richness" of the action space. In contrast, my approach employs a lattice representation that aims to increase the number of feasible body movements compared with previous works. For that, the foothold search regions depend on the given body action, i.e. forward/backward, diagonal, etc.

4.1.1 *Body action planning*

Body action planning over challenging terrain needs to consider the varying difficulty of the terrain areas, obstacles, types of actions, potential leg collisions, potential body orientations and kinematic reachability. Thus, given a terrain costmap (see Section 3.2.1), the body action planner computes a sequence of body actions that maximize the *cross-ability* of the terrain. The cross-ability describes how desirable is a determined body path in terms of the terrain conditions. It is quantified by computing a body cost. The body action plans are computed by searching over a *body-state graph* that is built using a set of predefined body movement primitives (see Fig. 4.2). Anytime Repairing A* (ARA*) is used to find the optimal action sequence [51]. A terrain-aware heuristic function is used to expand the node inside the OPEN list; for more details about the different lists (i.e. OPEN, INCONS and CLOSED used in ARA* see [51]). The body-action-based planning is described in pseudo-code in Algorithm 1.

4.1.1.1 *Graph construction*

The *body-state graph* is constructed using a lattice-based adjacency model. The lattice representation is a discretization of the action space into a set of feasible body movements. The body-state graph represents the transition between different body-states (nodes) and it is defined as a tuple, $\mathcal{G} = (\mathcal{S}, \mathcal{E})$, where each node $\mathbf{s} \in \mathcal{S}$ represents a *body-state* and each edge $\mathbf{e} \in \mathcal{E} \subseteq \mathcal{S} \times \mathcal{S}$ defines a potential feasible transition from $\mathbf{s}$ to $\mathbf{s'}$. A sequence of body-states (or body poses (x, y, θ) where it represents the 2D position and yaw angle, respectively) approximates the body trajectory that the controller will execute. An edge defines a feasible transition (body action) according to a set of body movement primitives. The body movement primitives are defined as body displacements (or body actions), which ensure feasibility together with a feasible footstep region. A feasible footstep region is defined according to the body action.

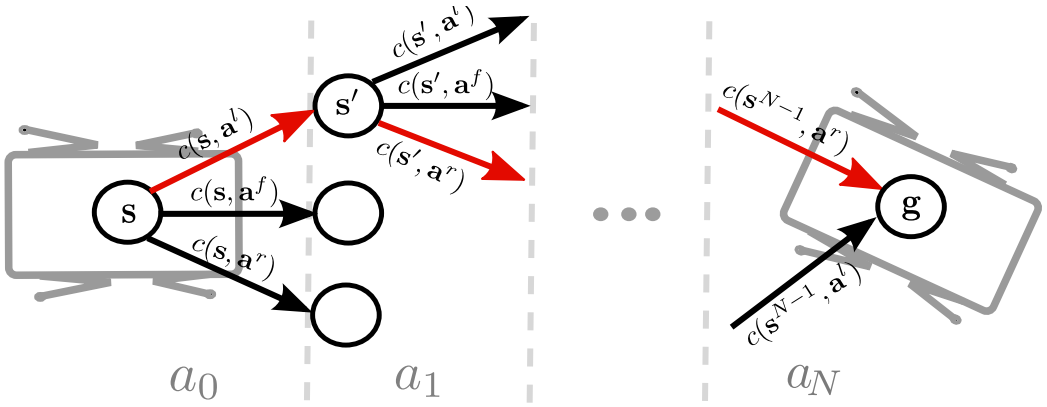


Figure 4.2: A sketch of the body action graph. The objective is to find a sequence of actions $\mathbf{a}$ from the current body state $\mathbf{s} = (x, y, \theta)$ to the goal state $\mathbf{g}$, that minimizes the accumulated action costs $c(\mathbf{s}, \mathbf{a})$. For simplicity only three possible actions are shown, namely move left ($\mathbf{a}^l$), right ($\mathbf{a}^r$) and forward ($\mathbf{a}^f$). The optimal action sequence $\{\mathbf{a}^l, \mathbf{a}^r, \ldots, \mathbf{a}^r\}$ found through ARA* is shown in red.

Given a body-state query, a set of successor states are computed using a set of predefined body movement primitives (line 15). A predefined body movement primitive connects the current body state $\mathbf{s} = (x, y, \theta)$ with the successor body state $\mathbf{s}' = (x', y', \theta')$. The graph $\mathcal{G}$ is dynamically constructed since the associated cost of transition $c_{body}(\mathbf{s}, \mathbf{s}')$ depends on the current and next states (or current state and action) (line 17). In fact, the feasible foothold regions change according to each body action, which affect the value of the transition cost. Moreover, an entire graph construction could require a greater memory pre-allocation and computation time than available (onboard computation). Fig. 4.3 shows the graph construction for the body action planner. The associated cost of every transition $c_{body}(\mathbf{s}, \mathbf{s}')$ is computed using the footstep regions (line 19). These footstep regions depend on the body action in such a way that they ensure feasibility of the plan, as is explained in Section 4.1.1.4. For every expansion, the footstep regions of LF (brown squares), RF (yellow squares), LH (green squares) and RH (blue squares) legs are computed given a body action.

The resulting states $\mathbf{s}'$ are checked for body collision with obstacles, using a predefined area of the robot, and invalid states are discarded. For legged locomotion over challenging terrain, obstacles are defined as unfeasible regions to cross, e.g. a wall or tree. In fact, the obstacles are detected when the height deviation w.r.t. the estimated plane of the ground is larger than the kinematic feasibility of the system in question HyQ. The obstacle map is built with 8 cm resolution since the amount of time for collision checking would increase significantly for higher resolutions.

4.1.1.2 *Body cost*

The body cost describes how *desirable* it is for the robot to be at a specific area of the terrain, and by evaluating a sequence of such areas and costs we understand how difficult/desirable it would be to cross a terrain with a given body path. This cost is designed to maximize the *cross-ability* of the terrain while minimizing the

Algorithm 1 Computes a set of body action over a challenging terrain using ARA* search over a growing body-state graph.

Data: Inflation parameter ϵ, time of computation t_c
Result: a body action plan $\mathbf{Q} = [\mathbf{q}^0, \mathbf{q}^1, \cdots, \mathbf{q}^L]$
function COMPUTEBODYACTIONPLAN($\mathbf{s}_{start}, \mathbf{s}_{goal}$)
    set ϵ and computation time t_c
    $g(\mathbf{s}_{start}) = 0$; $g(\mathbf{s}_{goal}) = \infty$
    OPEN $= \{\mathbf{s}_{start}\}$; INCONS = CLOSED$= \emptyset$
    while ($\epsilon \geqslant 1$ and $t_c < t$) **do**
        decrease ϵ
        OPEN = OPEN $\cup$ INCONS; CLOSED $= \emptyset$
        // fval($\mathbf{s}$) $= g(\mathbf{s}) + \epsilon h(\mathbf{s})$
        while (fval($\mathbf{s}_{goal}$) $>$ minimum fval in OPEN) **do**
            remove $\mathbf{s}$ with minimum fval from OPEN
            insert $\mathbf{s}$ into CLOSED
            generate action($\mathbf{s}$) from body primitives
            for all $\mathbf{u} \in$ action($\mathbf{s}$) **do**
                compute $\mathbf{s}'$ given $\mathbf{u}$
                compute the footstep regions given $\mathbf{s}'$
                compute $c_{body}(\mathbf{s}, \mathbf{s}')$ from footstep regions
                if $g(\mathbf{s}') > g(\mathbf{s}) + c_{body}(\mathbf{s}, \mathbf{s}')$ **then**
                    $g(\mathbf{s}') = g(\mathbf{s}) + c_{body}(\mathbf{s}, \mathbf{s}')$
                    add $\mathbf{s} \mapsto \mathbf{s}'$ to policy $\mathbf{s}' = \pi(\mathbf{s})$
                    if $\mathbf{s}'$ is not in CLOSED **then**
                        insert ($\mathbf{s}'$,fval($\mathbf{s}'$)) into OPEN
                    else
                        insert ($\mathbf{s}'$, fval($\mathbf{s}'$)) into INCONS
                    end if
                end if
            end for
        end while
    end while
    reconstruct body action plan $\mathbf{Q}$ from $\pi(\cdot)$
    return $\mathbf{Q}$
end function

path length. The body cost function c_{body} is a linear combination of: terrain cost c_t, action cost c_a, potential leg collision cost c_{pc}, and potential body orientation cost c_{po}. The cost of the transition from the body state $\mathbf{s}$ to $\mathbf{s}'$ is defined as follows:

$$\begin{aligned} c_{body}(\mathbf{s}, \mathbf{s}') = w_t^b c_t(\mathbf{s}) + w_a c_a(\mathbf{s}, \mathbf{s}') \\ + w_{pc} c_{pc}(\mathbf{s}) + w_{po} c_{po}(\mathbf{s}), \end{aligned} \tag{11}$$

where w_t^b, w_a, w_{pc} and w_{po} are the weights of terrain, action, potential leg and collision costs, respectively.

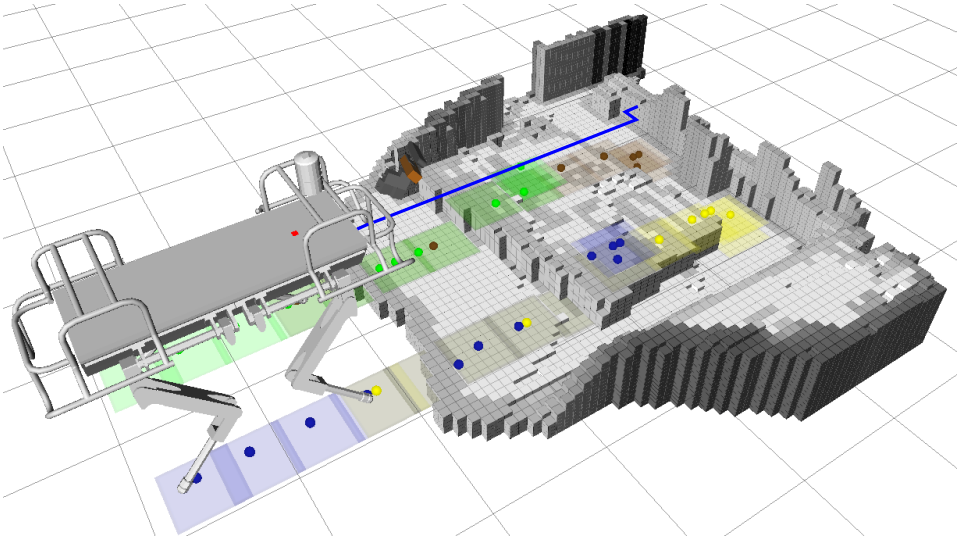


Figure 4.3: Illustration of graph construction of the least-cost path. The associated cost of every transition $c_{body}(\mathbf{s}, \mathbf{s'})$ is computed using the footstep regions. For every expansion, the footstep regions of LF (brown squares), RF (yellow squares), LH (green squares) and RH (blue squares) are computed according to a certain body action.

For a given current body state $\mathbf{s}$, the terrain cost c_t is evaluated by averaging the best n-footsteps terrain cost values around the foothold regions of a nominal stance (nominal foothold positions). The action cost c_a is defined by the user according to the desirable actions of the robot (e.g. it is preferable to make diagonal body movements than lateral ones). The potential leg collision cost c_{pc} is computed by searching potential obstacles in the predefined workspace region of the foothold, e.g. near to the shin of the robot. In fact, a potential shin collision is detected around a predefined region, which depends on the configuration of each leg as shown in Fig. 4.4. This cost is proportional to the height defined around the footstep plane, where red bars represent collision elements. Finally, the potential body orientation c_{po} is estimated by fitting a plane in the possible footsteps around the nominal stance for each leg.

4.1.1.3 *Reducing the search space*

Decoupling the planning problem into body action and foothold planning avoids the combinatorial search space explosion. This allows us to compute plans quickly (~1.5 sec on average for our benchmark trials which around 40 steps), and develop a closed-loop foothold planning approach that can deal with changes in the environment. Nevertheless the reduced motion space might not explore solutions that are better cost-wise but are not part of the solution set imposed by the action set. In contrast to previous approaches [45][81][39][87][85], our foothold planner uses a lattice-based representation of the configuration space, i.e. we represent the space using an abstract graph which is embedded in the Euclidean space. Our lattice

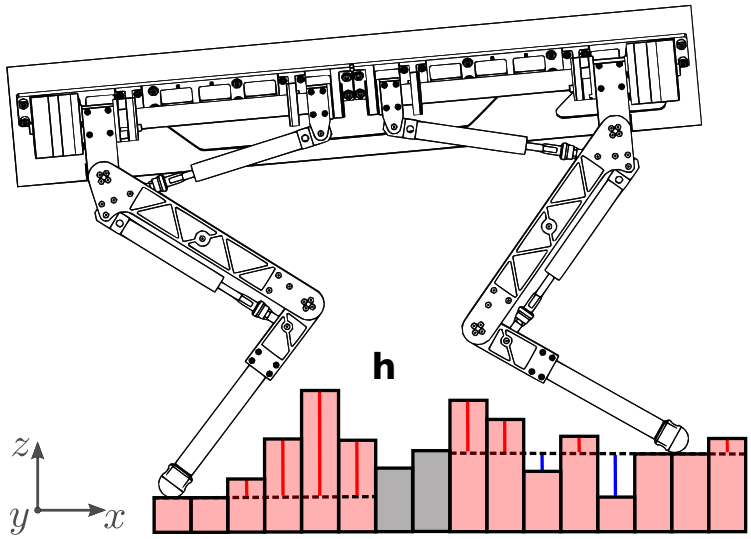


Figure 4.4: Computation of the potential shin collision. The potential shin collisions are detected around a predefined region (red bars). The computed cost is proportional to the height defined (red lines) around the footstep plane (dashed black lines), where blue height differences do not contribute to the cost.

representation uses a set of predefined body movement primitives that allows us to apply a set of rules that ensure feasibility and reduce the search space.

The body movement primitives are defined as 3D *actions*, $a \in (\Delta x, \Delta y, \Delta\theta)$, of the body that discretize the search space. We define a set of different movements such as: forward and backward, diagonal, lateral and yaw-changing motions.

4.1.1.4 *Ensuring feasibility*

Our lattice representation allows the robot to search a body-action plan according to a set of predefined body movement primitives. The body movement primitives also define nominal feet positions and accordingly transitions. This way we chose nominal positions that undertake to increase the quadruped's supporting-triangle area, and as a consequence, to improve the conditions of the subsequent quadratic optimization problem for the whole-body motion generation phase that follows in Section 4.2.

We classify the actions into six types of movement primitives: forward/backward, right/left, forward-left/backward-right, forward-right/backward-left, clockwise and counter-clockwise. Fig. 4.5 illustrates the footstep regions according to the type of body movement primitives. These footstep regions are defined to increase the triangular support areas, improving the execution of the whole-body plan.

4.1.1.5 *Heuristic function*

The heuristic function guides the search toward promising body actions, improving the efficiency of the search. Heuristic-based search algorithms require that

the heuristic function is admissible and consistent. Often, when planning within Cartesian spaces, heuristic functions estimate the cost-to-go by computing the Euclidean distance to the goal. Such heuristic functions do not consider the terrain difficulty and geometry, and often under or over-estimate the cost-to-go considerably. In contrast, we present a terrain-aware heuristic function that considers the terrain geometry and produces an estimated cost-to-go according to the already acquired knowledge of the environment.

The terrain-aware heuristic function estimates the cost-to-go (i.e. body actions) by averaging the terrain cost locally and estimating the number of footsteps to the goal:

$$h(\mathbf{s}) = -\bar{R}\mathcal{F}(\|\mathbf{g} - \mathbf{s}\|), \tag{12}$$

where $\bar{R}$ is the average cost and $\mathcal{F}(\|\mathbf{g} - \mathbf{s}\|)$ is the function that estimates the number of footholds from the current state $\mathbf{s}$ to the goal state $\mathbf{g}$.

4.1.1.6 *Ensuring online planning*

Open-loop, or offline, planning approaches fail to deal adequately with changes in the environment. In real scenarios, the robot has a limited range of perception that makes open-loop planning approaches unreliable. Closed-loop planning considers the changes in the terrain conditions, and uses predictive terrain conditions for non-perceived regions, improving the robustness of the plan. Dealing with re-

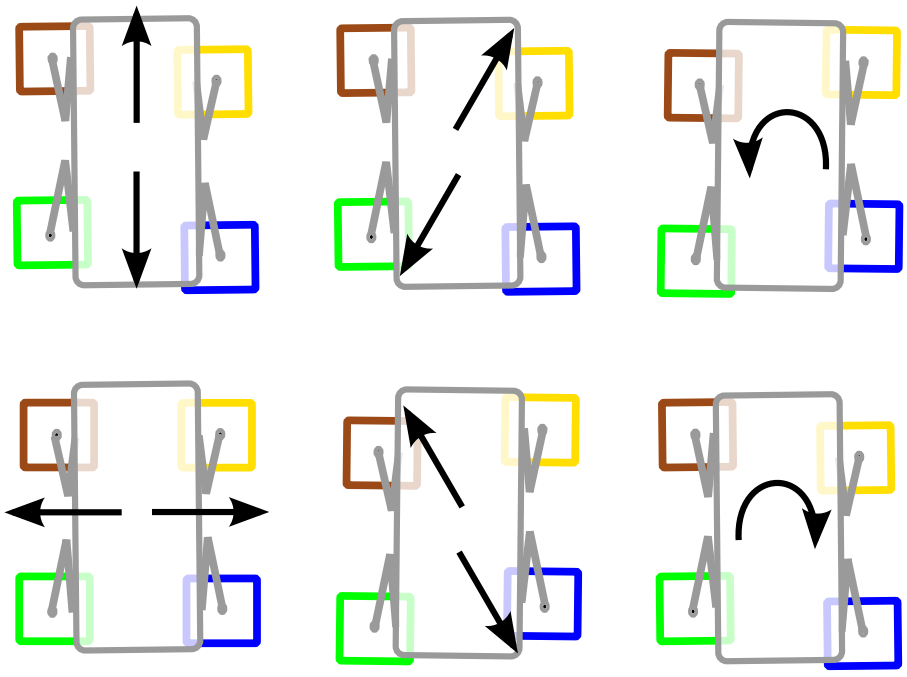


Figure 4.5: Different footstep search regions according to the body action. These footstep regions ensure the feasibility of the plan. Moreover, these predefined regions increase the triangular support areas, improving the execution of the whole-body plan. The brown, yellow, green and blue squares represent the footstep search regions for LF, RF, LH and RH feet, respectively.

planning and updating of the environment information requires that the information is managed in an efficient search exploration of the state space. We manage to reduce the computation time of building a costmap by computing the cost values from a voxelized map[1] of the environment. Additionally, we (re-)plan and update the information using ARA* [51]. ARA* ensures provable bounds on sub-optimality, depending on the definition of the heuristic function. Note that the terrain-aware heuristic function guides the search according to terrain conditions.

4.1.2 *Local foothold planning*

Given the desirable body action plan, $\mathbf{Q} = (\mathbf{x}_d, \mathbf{y}_d, \boldsymbol{\theta}_d)$, the foothold sequence planner computes the sequence of footholds that reflects the intention of the body action. A local greedy search procedure selects the optimal footstep target. Given a planned body action, it is defined a footstep search region where the foothold is locally searched. For every planned body action, the foothold planner finds the 4-next footholds, where the foothold sequence is predefined given the body action. Algorithm 2 describes how the foothold sequence is selected given a body action plan.

Algorithm 2 Computes a foothold sequence given a body action plan.

```
1:
2:  Data: foothold horizon F
3:  Result: a foothold sequence F
4:  function computeFootholdSequence(Q)
5:      set horizon F
6:      for q_{i=0:F} ∈ Q do
7:          for i=0:3 do
8:              compute the sequence of foothold e given q_i
9:              generate possible FOOTHOLD(e, q_i)
10:             compute the greedy solution f_min
11:             add f_min to F
12:         end for
13:     end for
14:     return F
15: end function
```

4.1.2.1 *Footstep cost*

The footstep cost describes how desirable is a foothold target, given a body state $\mathbf{s}$. The purpose of this cost function is to maximize the locomotion stability given a candidate set of footsteps. The footstep cost $c_{footstep}$ is a linear combination of:

1 A voxelized map is a volumetric discretization of the environment, i.e. a grid discretization in 3D space

terrain cost c_t, support triangle cost c_{st}, shin collision cost c_c and body orientation cost c_o. The cost of certain footstep $\mathbf{f}^e$ is defined as follows:

$$\begin{aligned} c_{footstep}(\mathbf{f}^e) = w_t^f c_t(\mathbf{f}^e) + w_{st} c_{st}(\mathbf{f}^e) \\ + w_c c_c(\mathbf{f}^e) + w_o c_o(\mathbf{f}^e), \end{aligned} \tag{13}$$

where $\mathbf{f}^e$ defines the Cartesian position of the foothold target e (foot index). We consider as end-effectors: the LF, RF, LH and RH feet of HyQ.

The terrain cost c_t is computed from the terrain cost value of the candidate foothold, i.e. using the terrain costmap (see Section 3.2.1). The support triangle cost c_{st} depends on the inradii of the triangle formed by the current footholds and the candidate one. As in the body cost computation, we use the same predefined collision region around the candidate foothold. Finally, the body orientation cost c_o is computed using the plane formed by the current footsteps and the candidate one. We calculate the orientation of the robot from this plane.

4.2 Motion Planning

The planned body trajectory ensures that the robot is *dynamically stable* at every time step. As in the approach [39], the CoM trajectory respects dynamic constraints without explicitly generating a CoP trajectory. We extend this approach by enabling *completely unconstrained*[2] swing-leg sequences and adaptation of four-leg support times according to the robot's motion.

For a CoM trajectory to be feasible it must be continuous and double differentiable. This way we avoid steps in accelerations that produce discontinuous torques. It is crucial for the inverse dynamics computation, and furthermore can damage the hardware and affect stability. Ensuring that the robot's CoM follows a fifth-order polynomial as:

$$x(t) = a_x t^5 + b_x t^4 + c_x t^3 + d_x t^2 + e_x t + f_x \tag{14}$$

satisfies both requirements. Note that $x(t)$ refers to the position of the CoM at time t, and the lateral component (i.e. $y(t)$) is equally described. Generating an optimal CoM trajectory represented by one spline can thus be reduced to find a set of optimal polynomial coefficients:

$$\mathbf{q} = \{a_x \dots f_x, a_y \dots f_y\} \in \mathbb{R}^{12}. \tag{15}$$

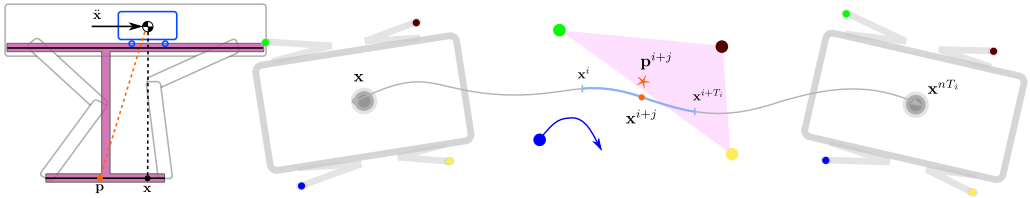
Describing the body trajectory for multiple steps by one polynomial restricts the movement. Hence, we describe the trajectory as a spline composed of multiple quintic polynomials. At each spline junction we require the last state ($t = T_i$) of spline i to be equal to the first state ($t = 0$) of the next spline $i+1$ as:

$$(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}})_{t=T_i}^{i} = (\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}})_{t=0}^{i+1}, \tag{16}$$

where $\mathbf{x} = [x, y, z]$ is the CoM position and $\mathbf{p} = [p_x, p_y, p_z]$ is the CoP position.

This ensures double differentiability and continuity of the trajectory, required by the floating-base inverse dynamics.

2 Not constrained to any specific gait sequence (e.g. the McGhee gait), the legs can swing in any order.



(a) (b)

Figure 4.6: (*a*) The cart-table model; the base of the table is formed by the feet that are in stance (in contact) while this model assumes that all the robot's mass is point-concentrated on the accelerating cart on top of the table. (*b*) The CoM trajectory is represented by a series of 5^{th} order polynomials that are constrained to have sequentially equal boundary values. At each (evaluated) point in time, we constrain the CoP to lie inside the support polygon that the stance feet form, thus the system is in a dynamically stable state.

4.2.1 *Dynamic stability*

To execute the planned footholds, a body trajectory must be found that ensures a stable stance at all time instances. In case of slow and static movements the CoM must be inside the current support polygon, i.e. the polygon formed by the legs in stance. For *dynamic* movements with large body accelerations we use the cart-table model (Fig. 4.6) [37] described by:

$$p_x = x - \frac{z\ddot{x}}{\ddot{z} + g_0}, \tag{17}$$

where p_x and x are the position of the CoP and the CoM respectively, z describes the height of the robot with respect to the feet, $\ddot{z}$ is the vertical acceleration of the body and g_0 represents the gravitational acceleration.

Dynamic stability is achieved by keeping the CoP inside the current support triangle instead of the CoM. This constraint is modeled by expressing the support triangle by three lines $\mathbf{l}$ of the form $px + qy + d = 0$. The CoP is considered to be *inside* a support triangle, if the following conditions[3] are met at every sampling interval:

$$p_l p_x + q_l p_y + r_l > 0 \quad \text{for } l = 1, 2, 3. \tag{18}$$

In reality there exist discrepancies between the CoP position from the model and the real robot. Additionally, desired body trajectories cannot be perfectly tracked as modeling, sensing and actuation inaccuracies are hard to avoid. Therefore, it is best to avoid the border of stable configurations by shrinking the support triangles by a stability margin r (see Fig. 4.7). Because of this, there is no continuous CoP trajectory when switching between diagonally opposite swing legs, as the support triangles are disjoint. We allow a transition period (four-leg support phase) during which the CoP is only restricted to the shrunk support *polygon* created by the four stance feet.

3 The direction of the inequality sign depends on the line convention.

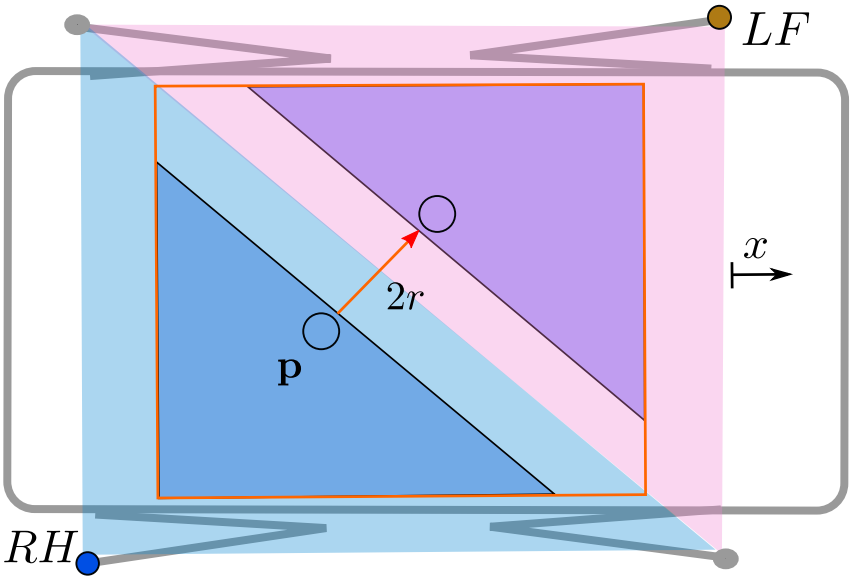


Figure 4.7: Disjoint support triangles due to the added stability margin r. When switching between swinging the LF to RH the CoP must move from the blue to the pink support triangle. Since all four feet are in stance during this phase, the CoP is only restricted by the red support polygon.

4.2.1.1 *Minimizing accelerations (Cost function)*

In addition to moving in a dynamically stable way, the trajectory should accelerate as little as possible during the execution period T. This increases possible execution speed and reduces required joint torques. As in [38] this is achieved by adding a cost function of the form:

$$g = w_x \int_0^T \ddot{x}^2(t)\,dt + w_y \int_0^T \ddot{y}^2(t)\,dt \tag{19}$$

to the quadratic program.

The directional weights **w** penalize sideways accelerations ($w_y = 1.5w_x$) more than forward-backward motions, since sideways motions are more likely to cause instability. This torque-minimization function can also be viewed as a regularization term. We solve the resulting convex QP using a freely available QP solver, namely *QuadProg++* [23].

4.2.1.2 *Irregular swing-leg sequences*

We allow a *completely irregular sequence* of steps for the CoP optimization. Our trajectory generator needs no knowledge of a predefined gait. For *every* step it checks if the next swing leg is diagonally opposite of the current swing leg. If so, the disjoint support triangles require a four-leg support phase in the optimization. This allows a greater decoupling from the foothold planner, which can generate swing leg sequences in any order useful for the success of the behavior.

4.2.1.3 *Feasible four-leg support time*

In addition to inserting a four-leg support phase, the *duration* t_{4ls} of this phase affects the quality of locomotion. A *too short* time requires large accelerations that might cause slippage in the feet when executed. A *too long* time on the other hand reduces locomotion speed and can oppose the natural dynamics of the system.

We adapt the four-leg support times according to the robot's motion using $t_{4ls} = r \cdot t_{swing}$, where r is the stability margin of the support triangles, and t_{swing} is the duration of a leg swing. This formulation respects the movement of the robot to overcome the stability margins of both support triangles. Furthermore, it complies to the body velocity imposed by t_{swing}. The more accurate the CoP describes the dynamics of the system, the smaller the margin r can be chosen, effectively reducing the four leg-support time and increasing locomotion speed. In case of perfect correspondence between model and reality and perfect trajectory tracking, a stability margin is not necessary. The four-leg support phase can be eliminated ($t_{4ls} = 0$), since the CoP can move smoothly between any support triangle of the walking gait.

4.2.2 *Trunk attitude and swing-leg trajectory*

To avoid kinematic limits the height of the robot follows the average height of the stance legs, raised by a desired ground clearance. The body pitch adapts to the difference between the height of the front and the hind legs, whereas the body roll depends on the difference between the height of the left and the right legs.

We adapt the swing leg trajectory based on the start and the goal foothold. If the goal foothold lies higher than the start, we raise the lift height as in [85] to avoid collisions. Furthermore, we swing outward to avoid potential collisions, e.g. with a stair step, while swinging up as in [85]. For steps on flat ground the lift height is low with no outward motion to increase speed.

4.3 Control and Execution

Dynamic whole-body motions require orchestrated and precise actuation of all the joints. Simple PD joint position controllers do not suffice for such motions, especially when considering uncertainties in the environment and/or model inaccuracies. We use a control scheme (Fig. 4.1) that combines a virtual model with a floating-base inverse dynamics controller.

After receiving an arbitrary sequence of footholds from the footstep planner, the whole-body motion generator calculates desired (feedforward) accelerations $\ddot{\mathbf{x}}^{d}$ for the body and a virtual model (VM) control loop adds feedback accelerations $\ddot{\mathbf{x}}^{fb}$ should the robot deviate from the desired trajectory. The inverse dynamics produce the majority of the joint torques which are combined with a low-gain joint-space PD controller to compensate for possible model inaccuracies. The computed

reference torques are then tracked by the low-level torque controller [5][3]. Note that $\mathbf{x}$ describes the linear and rotational coordinates of the body as:

$$\mathbf{x} = (\mathbf{x}_{cog}, \mathbf{R}_b), \dot{\mathbf{x}} = (\dot{\mathbf{x}}_{com}, \boldsymbol{\omega}_b), \ddot{\mathbf{x}} = (\ddot{\mathbf{x}}_{com}, \dot{\boldsymbol{\omega}}_b), \tag{20}$$

where $\mathbf{R}_b \in \mathbb{R}^{3\times3}$ is a coordinate rotation matrix representing the orientation of the base w.r.t. the world frame and $\boldsymbol{\omega}_b \in \mathbb{R}^3$ is the angular velocity of the base.

4.3.1 *Virtual Model*

The feedback action which compensates for possible inaccurate execution and drift is created by a virtual spring and damper system as in [32]. The linear/rotational springs map an error in position/orientation into a force $\mathbf{F}_{vm}$ and torque $\mathbf{T}_{vm}$ acting on the robot body (Fig. 4.8) as:

$$\begin{aligned} \mathbf{F}_{vm} &= \mathbf{P}_x(\mathbf{x}^d_{com} - \mathbf{x}_{com}) + \mathbf{D}_x(\dot{\mathbf{x}}^d_{com} - \dot{\mathbf{x}}_{com}) \\ \mathbf{T}_{vm} &= \mathbf{P}_\theta e(\mathbf{R}^d_b \mathbf{R}_b^\top) + \mathbf{D}_\theta(\boldsymbol{\omega}^d_b - \boldsymbol{\omega}_b), \end{aligned} \tag{21}$$

where the superscript d refers to the desired values dictated by the whole-body motion generator and non-superscript values describe the state of the robot as estimated by the on-board state estimator [2]. We define $e(.) : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ as a mapping from a rotation matrix to the associated rotation vector [7], and $\mathbf{P}_x, \mathbf{D}_x, \mathbf{P}_\theta, \mathbf{D}_\theta \in \mathbb{R}^{3\times3}$ as the positive-definite diagonal matrices of proportional and derivative gains, respectively. Expressing the body feedback action in terms of forces and moments allows us to give the virtual model gains a physical meaning of stiffness and damping and thus can be intuitively set and used.

Since the inverse dynamics computation requires reference *accelerations*, we multiply the forces/moments (*wrench*) $\mathcal{W}_{vm} = (\mathbf{F}_{vm}, \mathbf{T}_{vm})$ with the inverse of the composite rigid body inertia, $\mathbf{I}_c$, of the robot at its current configuration. Adding this body feedback acceleration to the desired body acceleration produced by the whole-body motion generator creates the 6D reference acceleration (linear/rotational) for the inverse dynamics computation as:

$$\ddot{\mathbf{x}}^{ref} = \ddot{\mathbf{x}}^d + \mathrm{I}_c^{-1}\mathcal{W}_{vm}. \tag{22}$$

By combining a feedforward acceleration $\ddot{\mathbf{x}}^d$ with a body-feedback acceleration, we achieve accurate tracking while maintaining a compliant behavior. This is crucial for robots that physically interact with their environment, in real-world scenarios.

4.3.2 *Floating- Base Inverse Dynamics*

The floating-base inverse dynamics algorithm calculates the joint torques required to execute the reference body accelerations. We can partition the dynamics equa-

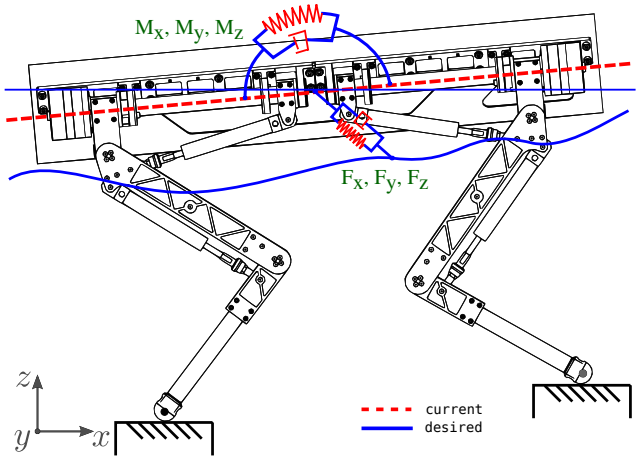


Figure 4.8: A virtual model control scheme is used to close a feedback loop at the robot-body level [32]. Effectively, this is a PD tracking controller that tries to minimize the error between the desired and current body state (position and orientation). This produces the feedback wrench ($\mathcal{W}_{vm}$) that is then transformed to the feedback body acceleration, $\ddot{\mathbf{x}}^{fb}$.

tion of the robot into the unactuated base coordinates $\mathbf{q}_b$ ($n_b = 6$ equations for our case) and the active joints' coordinates $\mathbf{q}_q$ ($n_q = 12$ equations), e.g. [20], as:

$$\underbrace{\mathbf{H}(\mathbf{R},\mathbf{q})\begin{bmatrix}\ddot{\mathbf{q}}_b\\ \ddot{\mathbf{q}}_q\end{bmatrix} + \begin{bmatrix}\mathbf{h}_b\\ \mathbf{h}_q\end{bmatrix}(\mathbf{R},\mathbf{q},\omega,\dot{\mathbf{q}})}_{\mathbf{b}} = \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\tau}\end{bmatrix} + \begin{bmatrix}\mathbf{J}_{cb}^T\\ \mathbf{J}_{cq}^T\end{bmatrix}\boldsymbol{\lambda}, \tag{23}$$

where $\mathbf{H}$ is the floating-base mass matrix, $\mathbf{h} = (\mathbf{h}_b, \mathbf{h}_q)$ is the force vector that accounts for Coriolis, centrifugal, and gravitational forces, $\boldsymbol{\lambda}$ are the ground contact forces, and their corresponding Jacobian $\mathbf{J}_c = \begin{bmatrix}\mathbf{J}_{cb} & \mathbf{J}_{cq}\end{bmatrix}$. $\boldsymbol{\tau}$ is the vector that holds the inverse-dynamics torques that are calculated.

The left hand term, $\mathbf{b} = (\mathbf{b}_b, \mathbf{b}_q)$, can be computed efficiently using the Featherstone implementation of the Recursive Newton-Euler Algorithm (RNEA) [17]. Since the CoM acceleration $\ddot{\mathbf{x}}_{com}^{ref}$ is defined in a frame aligned with the base frame but with the origin in the CoM, a translational coordinate transform ${}_b\mathbf{X}_{com}$ is performed to get the 6D base spatial acceleration: $\ddot{\mathbf{q}}_b = {}_b\mathbf{X}_{com}\ddot{\mathbf{x}}^{ref}$, as in [17].

By partitioning the dynamics equation, as in (23), and given that the base is not actuated, we can directly compute, in a least-squares way, the vector of ground reaction forces $\boldsymbol{\lambda}$ from the first n_b equations, $\boldsymbol{\lambda} = \mathbf{J}_{cb}^{+}\mathbf{b}_b$, where $()^+$ denotes the Moore-Penrose generalized inverse. The last n equations are used to produce the reference joint torques, $\boldsymbol{\tau}^{id} = \mathbf{b}_q - \mathbf{J}_{cq}^T\boldsymbol{\lambda}$.

4.4 Results

The following section describes the experiments conducted to validate the effectiveness and quantify the performance of our framework. The first set of experiments is designed to evaluate the capabilities of our motion planner. We use a

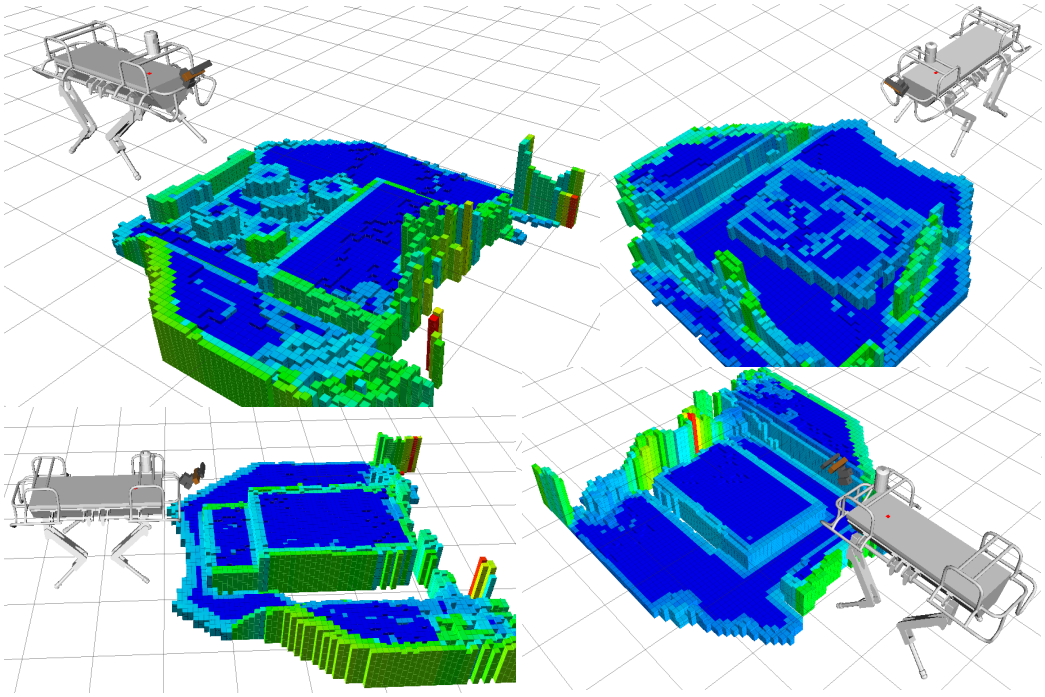


Figure 4.9: The planning benchmarks used to analyze the quality of the produced plans. *Top left*: stepping stones; *Top right*: pallet; *Bottom left*: stair; *Bottom right*: gap.

set of benchmarks of realistic locomotion scenarios (see Fig. 4.9): stepping stones (top right), pallet (top left), stair (bottom right) and gap (bottom left). In these experiments, we compared the cost, number of expansions and computation time of ARA* against A star (A*) using our lattice representation. The results are based on 9 predefined goal locations. In the next experiment the robot must plan online with dynamic changes in the terrain. In the experiment that follows after, we show the online planning and perception results. Finally, we validate the performance of our locomotion framework with the HyQ robot.

4.4.1 *Evaluation of path and foothold planning*

4.4.1.1 *Initial plan results*

The stepping stones, pallet, stair and gap experiments evaluate the initial plan quality (see Table 4.1) of our approach using A* and ARA*. To this end, we plan a set of body actions and foothold sequences between 9 predefined goal locations, approximately 2 m away from the starting position, and compare the cost and number of expansions of the body action path, and the planning time of ARA* against A*. Three main factors contribute to the decreased planning time while maintaining the quality of the plan:

First, the lattice-based representation (using body movement primitives) considers versatile movements in the sense that it allows us to reduce the search space around feasible regions (feasible motions) according to a certain body action, in

contrast to grid-based approaches that ensure the feasibility by applying rules that are unaware to body actions. Second, our terrain-aware heuristic function guides the tree expansion according to terrain conditions in contrast to a simple Euclidean heuristic. Finally, the ARA* algorithm implements a search procedure that guarantees bounded sub-optimality in the solution given a proper heuristic function [51]. Fig. 4.10 shows the initial plan of A* and ARA*.

The plans that ARA* produces within the given time budget are on average twice as costly as the plans of A*. Nevertheless, this allows for a dramatic decrease of the time it takes to compute a path, 333.9 sec for A* versus 1.7225 sec for ARA*, while any extra time budget can be used to optimize (*repair*) the computed path. These results were obtained in the onboard PC on HyQ, i.e a Pentium i5 with Linux kernel. Note that ARA* can be let to run until exhaustion, inc which it would match the characteristics in final path cost and computational time of A*.

4.4.1.2 *Online planning and perception*

Using a movement primitive-based lattice search reduces the size of the search space significantly, leading to responsive planning and replanning the next set of steps. In our experimental trials, we chose a lattice graph resolution (discretization) of 4 cm for x/y and 1.8° for θ and goal state is never more that 5 m away from the robot. In these trials, the plan/replan frequency is approximately 0.5 Hz.

On the other hand, the efficient occupancy grid-based mapping allows us to incrementally build up the model of the environment and focus our computations to the area of interest around the robot body, generating plans quickly. This allows us to locally update the computed costmap and incrementally build the costmap as the robot moves with an average response frequency of 2 Hz.

We generate swift and natural dynamic whole-body motions from an n-step lookahead optimization of the body trajectory that uses a dynamic stability metric, the CoP. A combination of floating-base inverse dynamics and virtual model control accurately executes such dynamic whole-body motions with an actively compliant system [86]. Our locomotion system is robust due to a combination of online planning and compliance control, as presented in Section 4.3.

Table 4.1: Cost of the plan (Cost), number of expansions (Exp.) and computation time (Time, in seconds) averaged over 9 trials of A* and ARA*.

	A*			ARA*		
Terrain	Cost	Exp.	Time	Cost	Exp.	Time
S. Stones	364.7	3191	428.7	597.4	12.1	1.59
Pallet	269.2	270	271.2	587.7	12.7	1.74
Stair	306.1	306	308.1	646.4	12.7	1.55
Gap	325.6	326	327.6	647.0	13.0	2.01

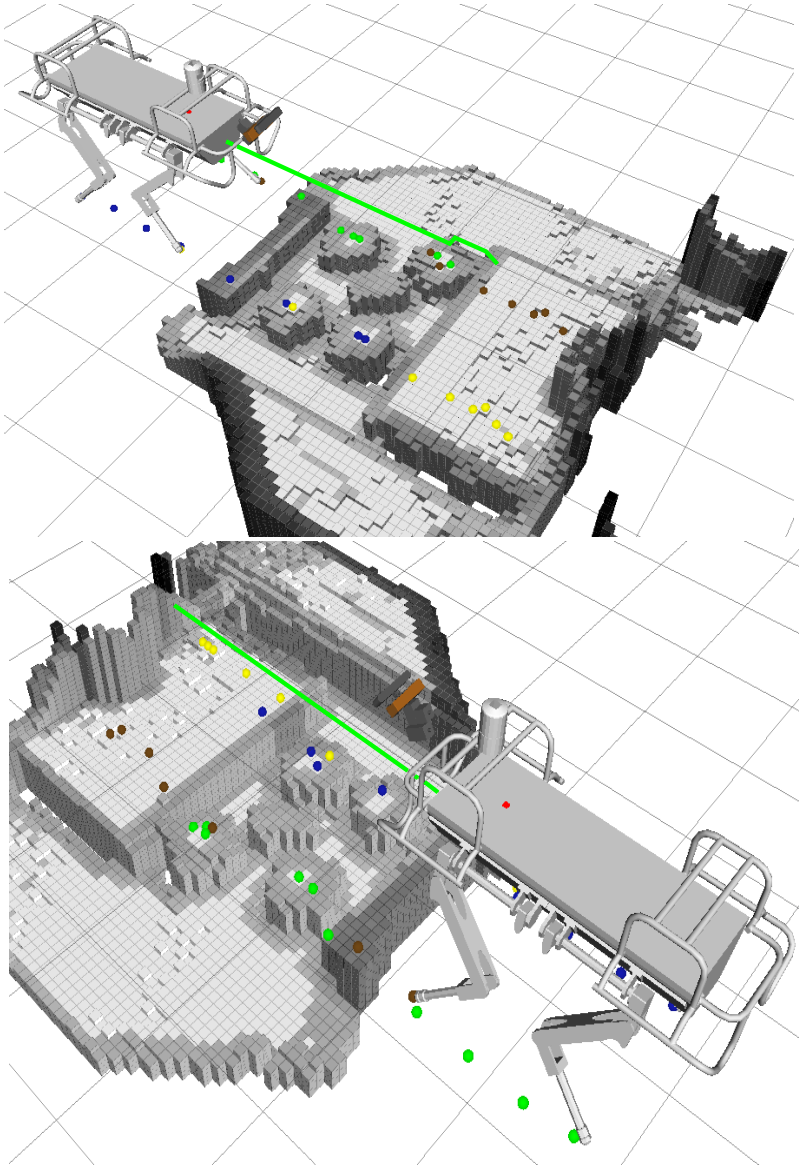


Figure 4.10: The body action (green line) and foothold sequence plan of A* (top) and ARA* (bottom) given the costmap (grey scale). The brown, yellow, green and blue points represent the planned footholds of the LF, RF, LH and RH feet, respectively.

4.4.2 *Trials*

The first experiment starts with a flat, obstacle-free terrain. After the robot has planned initial footholds, a pallet is placed into the terrain (Fig. 4.11). Later we show the terrain costmap, the planned footholds, and the execution of an initial plan in Fig. 4.12. In the next experiments the robot must climb one and two pallets of dimensions 1.2 m × 0.8 m × 0.15 m. The height of one pallet is equal to 20% of the leg length. Furthermore we show that the robot traverses a gap of 35 cm, which is approximately half the length between the front and hind hips. The final

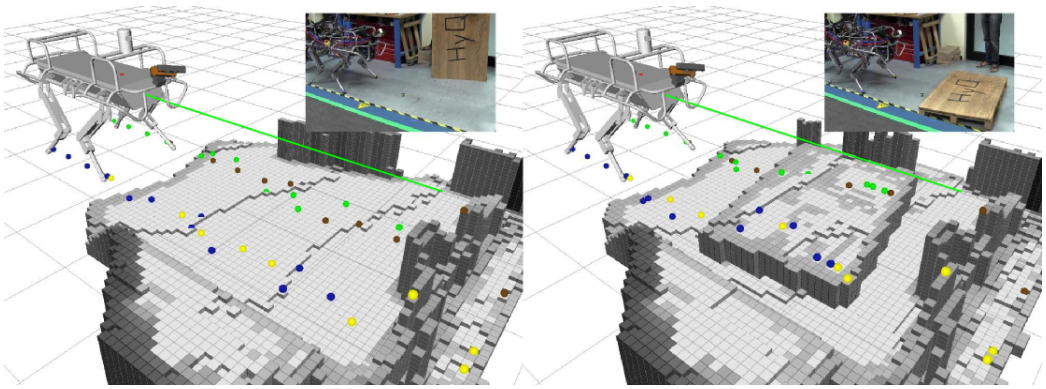


Figure 4.11: (Re-)planning and perception on-board. The left image presents the plan for a flat terrain, then, the right image reflects the re-planning and updating of the costmap (grey scale map). The green, brown, blue and yellow points represent the planned footholds of the LH, LF, RH, RF legs, respectively. The green line represents the body path according to action plan.

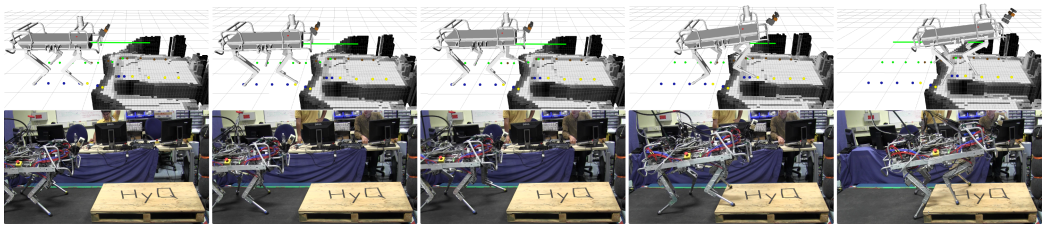


Figure 4.12: Snapshots of pallet trial used to evaluate the performance of our planning approach. From top to bottom: planning and terrain costmap; execution of the plan in HyQ.

experiment consist of two pallets connected by a sparse path of stepping stones. The pallets are 1.2 m apart and the stepping stones lie 0.08 m lower than the pallets.

For each experiment, we specify the $\mathbf{s} = (x, y, \theta)$ goal state. The foothold planner finds a sequence of footsteps of an arbitrary order, which the controller then executes dynamically. We validate the performance of our framework in 4 scenarios as seen in Fig. 4.13 and compare it to our previously achieved results (Table 4.2) on the same benchmark tasks. Additionally, Fig. 4.14 shows an overview of the benchmark trials along with generated footholds and body motion plans.

4.4.3 *Evaluation of whole-body motion generation and execution*

We evaluate the performance of the our locomotion framework in few different point of view: perception and (re-)planning, speed while dynamically stable, model accuracy, avoiding kinematic limits and stability despite irregular swing-leg sequences. All these aspects evaluate individually the performance of the foothold and motion planner.

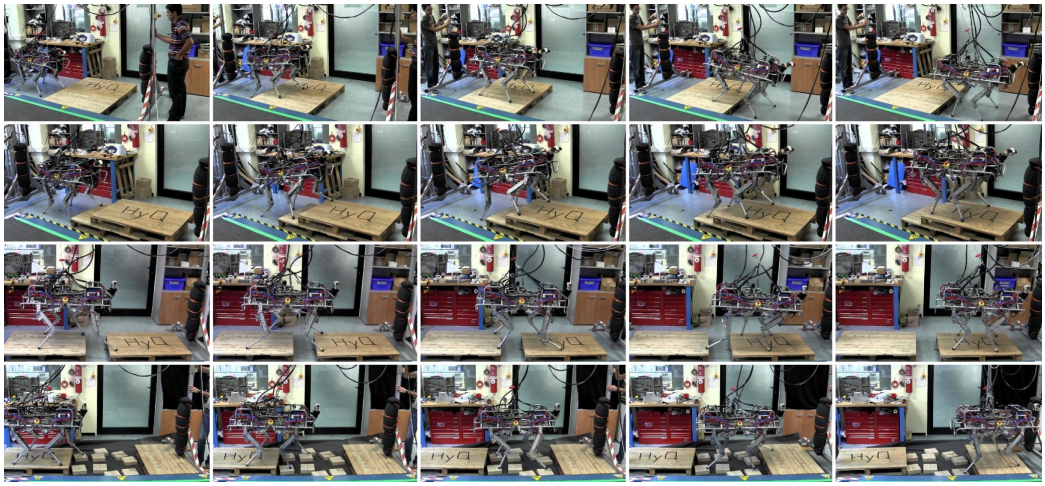


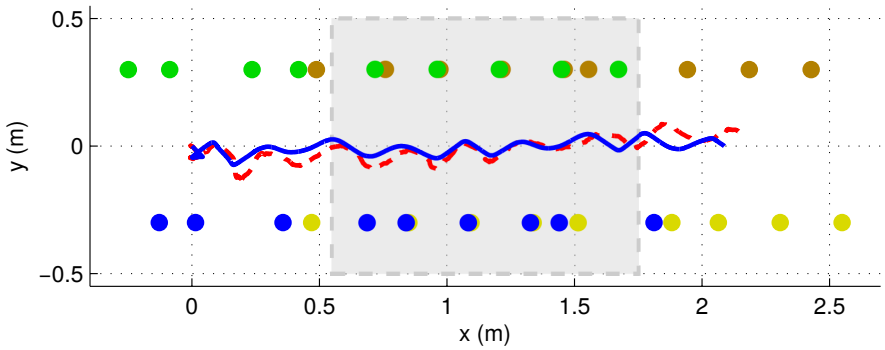
Figure 4.13: Snapshots of 4 experimental trials used to evaluate the performance of our framework. From top to bottom row: crossing a 15 cm hight pallet; climbing a stair-like structure consisting of two stacked pallets; traversing a 35 cm gap and crossing a sparse set of stepping stones. In addition, Fig. 4.14 gives and overview of the chosen footholds, planned body paths and executions on this set of benchmark trials.

Table 4.2: Forward speed and success rate of experiments averaged over 10 trials and compared to previous results from [85].

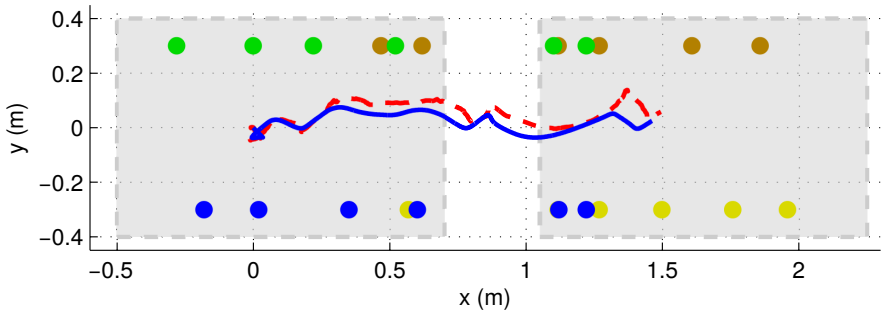
	Speed [cm/s]			Success Rate [%]		
Terrain	Curr.	Prev.	Ratio	Curr.	Prev.	Ratio
Step. Stones	7.3	1.7	4.2	60	70	0.9
Pallet	9.5	2.1	4.5	100	90	1.1
Two Pallets	10.2	1.8	5.8	90	80	1.1
Gap	12.7	-	-	90	0	-

4.4.3.1 *Perception and (re-)planning*

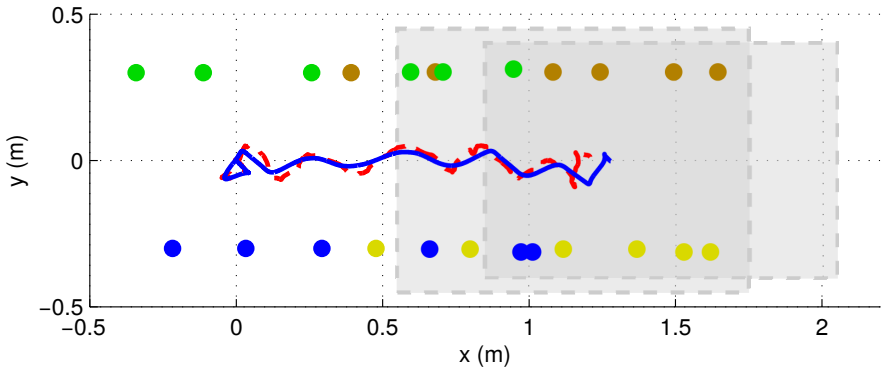
Efficient occupancy grid-based mapping and focusing our computations to an area of interest around the robot body greatly increase computation speed. This allows us to incrementally build a model of the environment and update the terrain costmap at a frequency of 2 Hz. Using the action-based search graph together with ARA* allows us to replan footholds at a frequency of approximately 0.5 Hz for goal states up to 5 m.



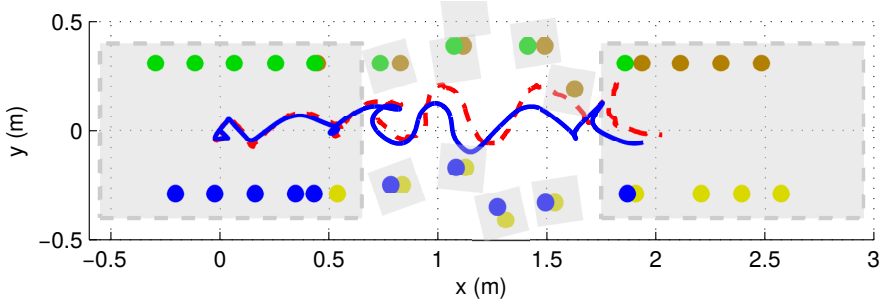
(a) Stepping over a pallet.



(b) Crossing a 35cm gap.



(c) Stair-climbing.



(d) Crossing a sparse set of stepping-stones.

Figure 4.14: Overview of experimental trials. Filled circles represent color-coded footholds, i.e. the brown, yellow, green and blue represent the LF, RF, LH and RH feet, respectively. The gray squares represent pallet and stepping-stone geometries. The solid blue line is the desired dynamically-stable body trajectory, while the red dashed line is the actual trajectory that the robot achieves. Possible reasons behind this difference are discussed in Section 4.5.

4.4.3.2 *Speed while dynamically stable*

The pallet climbing and gap experiments show the *speed* (Table 4.2) that our framework can achieve: This is due to the fact, that the body can move faster while still being stable, since we are using a *dynamic* stability criterion. All accelerations and decelerations are optimized, so that the CoM never leaves the support polygon. In addition, since we are not directly producing torques with the virtual model feedback controller, but only accelerations for the inverse dynamics controller our feedback actions also respect the dynamics of the system. Furthermore, the duration of the four-leg-support phase is significantly reduced: It is much faster to move the CoM from one support triangle to another one.

4.4.3.3 *Model accuracy*

Walking over a 35 cm gap (approximately half the length between front and hind hips) shows the stability of the robot despite of dynamic motions. When crossing the gap the robot accelerates up to a body velocity of 0.5 m/s and is able to decelerate again without losing balance. This shows, that the simple cart-table model is a sufficient approximation for large quadrupeds performing locomotion tasks.

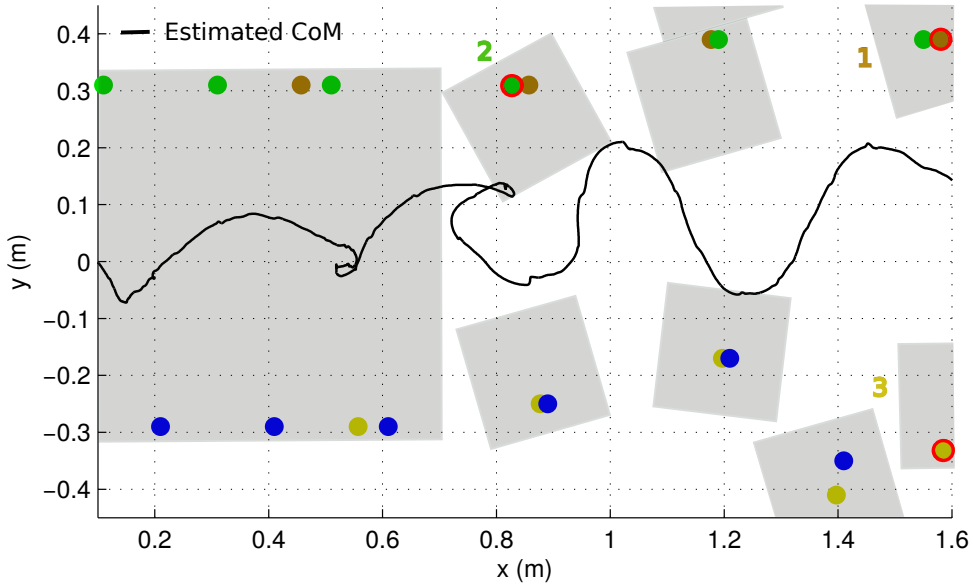
4.4.3.4 *Avoiding kinematic limits*

Attempting to cross the gap with a statically stable gait would overextend the legs, since large body motions are required to move the robot into statically stable positions. Dynamic motions allow us to keep the CoM closer to the center of all four feet, since stability can be achieved by appropriate accelerations, avoiding kinematic limits.

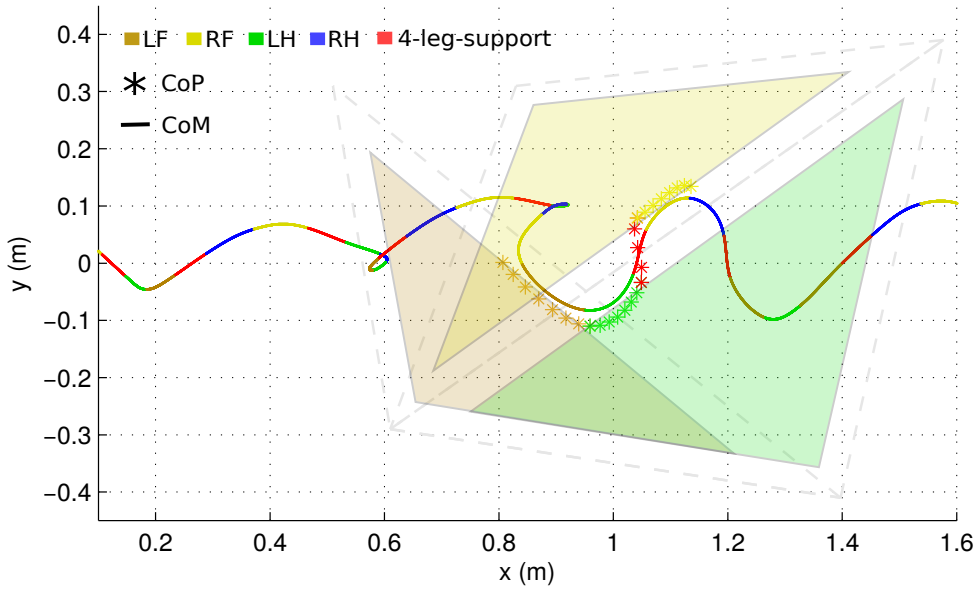
4.4.3.5 *Stability despite irregular swing-leg sequences*

Walking over the stepping stones demonstrates the ability of the controller to execute irregular swing-leg sequences in a dynamically stable manner. Fig. 4.15 shows how starting from a lateral sequence gait (LH-LF-RH-RF) the foothold sequence changes to traverse these irregularly placed stepping stones. Despite this, the generated CoM trajectory (colored solid line, bottom) is dynamically stable, since the CoP (asterisk) is always kept inside the current support triangle. When comparing the actual (top) and desired (bottom) CoM trajectories, a tracking error is evident. By keeping the CoP e.g. $r = 6\,\text{cm}$ away from the stability borders, we are robust even against these inaccuracies.

The whole body motion generator inserts four-leg-support phases (red section) only whenever it detects disjoint support triangles in the swing-leg sequence. While executing steps 1 and 2 (Fig. 4.15) no four-leg-support phase is necessary, because the triangles are not disjoint. Only after returning to swing the right-front leg, the robot requires a four-leg support phase for the CoP to transition from the green (LH) to the yellow (RF) support triangle at $(x, y) = (1.1, 0)$.



(a) Stepping stones execution



(b) Stepping stones plan

Figure 4.15: (a) The body motion when walking over the stepping stones is shown in black. The planned footholds are shown and the irregular step sequence LF(1) → LH(2) → RF(3) is highlighted (red circles). (b) The 3 shrunk support triangles corresponding to the highlighted step sequence brown → green → yellow are shown. Additionally the planned CoM (solid line) and CoP trajectory for the duration of these 3 steps is illustrated (asterisk). While the CoM (solid line) does not reach the support triangles, the CoP does, causing dynamic stability. When switching between disjoint support triangles (green → yellow) four-leg support phases are inserted (red) to allow a smooth transition.

4.5 Discussion

In this section we will identify possible factors that can limit the success of our framework (failures). This is according to the trials that we have performed over a substantial period of working with the HyQ.

(1) The *cart-table model* estimates the CoP position but neglects the angular components of the motion of the body which can lead to inaccurate evaluation of the CoP within the support polygon. This does not affect the behavior when the attitude of the robot does not change but can affect stability going up or down stairs, crossing uneven stepping stones, etc., and the body trajectory changes along roll, pitch and yaw. In practice HyQ never generates angular body accelerations that can move the CoP that far away from the simplified estimate point. In addition the stability margin, r, that is defined inside the support polygon, ensures that error in estimation, model inaccuracies or the use of the simplified CoP measure, do not impact the robot's stability. Section 5.1 will present a methodology to guarantee dynamic walking with trunk attitude modulation.

(2) Unwanted *foot-slippage* can impact the performance of our framework. This can happen when the acceleration of the body is significant and one or some of the feet slip backwards, or when a foot is only slightly loaded so subsequent "pushing" backwards results in the foot sliding. The former occurs due to non-explicitly-bounded body acceleration at the body trajectory optimization phase and due to no friction-cone constraints in the inverse dynamics torque calculation step. The latter is also attributed to the lack of friction-cone constraints at the inverse dynamics torque calculation, while setting a minimum loading threshold, i.e. a minimum loading value beyond which to consider a leg as in stance, can solve this problem. In practice this limits the forward velocity that can be achieved by the framework, as large accelerations can result in slippage, and limit behaviors within conservative bounds. Approaches that include friction-cone constraints can be applied. In the next chapter we show a QP-based optimization of inverse dynamics, that limits the GRFs within the estimated friction-cone approximations and ensures adequate loading for all stance legs as presented in [19].

(3) Last, accurately estimating the state of the robot has also proven a challenging task. An adequately reliable state-estimate is important for executing planned walks, as foot placement directly depends on the estimate of the body position. In addition, the body-level feedback loop that the Virtual Model (VM) implements also depends on the body state estimate. This estimate generally slowly drifts in yaw, something that affects the position estimate and subsequently distorts the maps created over a longer time period. Using vision approaches can eliminate drift and improve the state estimate but up to now this has been beyond the scope of our research. Currently we have been working in the developing of state estimation algorithms for quadrupedal locomotion over rough terrain [**?**]. Another source of state estimate error is foot-slippage, as the algorithm used assumes that stance feet do not move with respect to the world frame. This can produce sudden jerks that can affect the inverse dynamics loop and also cause the position estimate

to drift. We believe that by alleviating the foot-slippage problem we would also considerably increase state estimation accuracy.

4.6 Conclusion

In this chapter, we presented a dynamic whole-body locomotion framework that executes versatile movements that are planned on-board. We showed, how sequences of footsteps are planned given a body action plan, and how a change in the environment causes the foothold generator to replan footholds online. We used a set of predefined body movement primitives to reduce the computational demand of planning, and compute plans online using incoming terrain information. We presented a whole body motion planner, which is able to generate a CoP-stable body trajectory despite irregular swing-leg sequences to execute footholds dynamically.

Decoupling motion and foothold planning increase the computation time by reducing the search space. In these approaches, the foothold selection can just consider the robot's kinematic model, which limits the locomotion capabilities of our system (e.g. step time modulation). As a consequence the dynamic trajectory generator requires to tune an appropriate step time given a foothold sequence. Furthermore, the foothold planning may not be able to plan for more complex terrain conditions such as stepping stones with fewer stones. In Chapter 5, we show how a coupled motion and foothold planning approach increases the capabilities of the locomotion system. This planning approach adapts the step duration and can tackle more complex stepping stones terrains.

The cart-table model reduces the dimensionality of the problem but neglects the angular momentum of the motion, which is required for climbing up and down tasks. However, our experiments show that we can tackle terrain with small elevation with the cart-table model. In Section 5.1 we develop a novel method that ensures dynamic stability (i.e. the CoP condition) while the trunk attitude is adapted. For that, the stability margin is used to allow the robot to limit the trunk attitude adjustment. We limit the trunk attitude accelerations because it allows us to bound the maximum applied CoM torque. In fact, this will limit the difference between the CoP and Centroidal Moment Pivot (CMP) points. And as result the CoP condition cannot guarantee dynamic stability if the CMP does not stay inside the support polygon.

The discretization of the terrain model shows good performance in terms of foothold selection and computation time. We could represent the different terrain benchmark with the aforementioned geometric features: standard deviation of height values, the slope and the curvature. Furthermore, a terrain model as costmap is a suitable description for motion planning in rough terrain as shown in other works [45][38][85]. Other ways of modeling the terrain can be implemented with constraint but they might increase the complexity of problem. Furthermore, the terrain costmap might not be a convex function, which increases the problem complexity. Section 5.2 presents a method that addresses these issues.

In the next chapter, we bring the kinematic planning (i.e. foothold planner) and the dynamic execution closer. The idea is to produce desired state trajectories and footholds through a *single* trajectory optimization problem (i.e. coupled planning). Additionally, we develop a QP-based controller where it ensures that the GRFs respect the friction-cone constraints.

5

Coupled Motion and Foothold Planning

In this chapter, we address the locomotion as a coupled planning problem of CoM motions and footholds, where the foothold locations are selected using a terrain costmap while the trunk height and attitude are adapted for coping with different terrain elevations (see Fig. 5.1). First, we jointly generate the CoM trajectory and the swing-leg trajectory using a sequence of parametric preview models and the terrain elevation map (Section 5.1). Then, we optimize a sequence of control parameters (the CoP displacement, the phase duration and the foothold locations) given the terrain costmap (Section 5.2). To realize the low-dimensional plan, the controller selects appropriate torque commands, which are computed by the combination of a trunk controller with a joint-space torque controller (Section 5.3). The proposed trajectory optimization method increases the locomotion capabilities compared to the decoupled planner presented in the previous chapter. All the material presented in this chapter has been previously published in [52].

In motion planning, terrain adaptation and automatic gait discovery can be posed as general trajectory optimization method, similar to [60][9][63][71]. Nonetheless, these optimization methods often suffer from local minima, limiting their applicability to rough terrain locomotion. Often, challenging terrain conditions may increase the nonconvexity of the problem, and defining a good enough warm-start point might not be possible. Moreover, automatic walking pattern generation increases the number of local minima, especially in rough terrain locomotion. For instance there are multiple choices for stepping stones crossing, i.e. by adjusting the step duration to maximize the safety of the task, and at the same time, to minimize the energy consumption. We can described the gait with a set of low-dimensional parametrized models, similar to [56]. Low-dimensional parametrized models allow us to combine stochastic-based optimization solvers that properly handle the aforementioned challenges.

5.1 Trajectory Generation

This section describes the low-dimensional trajectory generation from an optimized sequence of control parameters and a given terrain heightmap. We generate the horizontal CoM trajectory and the 2D foothold locations using a sequence of low-dimensional preview models. In order to cope with the terrain elevation, we modulate the trunk attitude and height using an estimate of the support plane, and the maximum allowed angular accelerations of the trunk (for more details see Section 5.1.1.2). We describe the sequence of control parameters w.r.t. the horizontal frame, which allows us to decouple the CoM and trunk attitude planning.

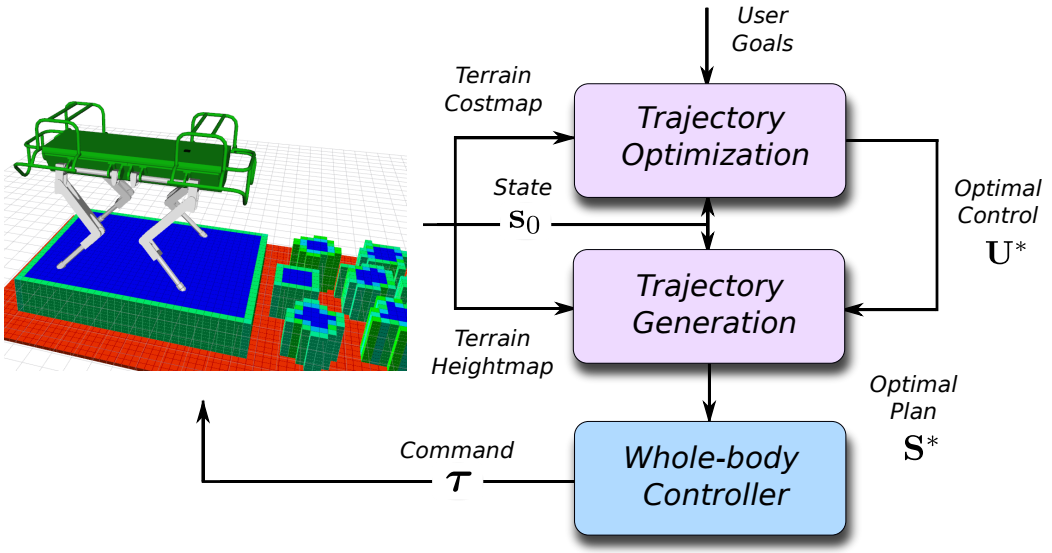


Figure 5.1: Overview of the trajectory optimization framework for locomotion on rough terrain. We compute offline an optimal control sequence $\mathbf{U}^*$ given the user's goals, the actual state $\mathbf{s}_0$ and the terrain costmap. Given this optimal control sequence, we generate the optimal plan $\mathbf{S}^*$ that copes with the changes in the terrain elevation through trunk attitude planning. Lastly, the whole-body controller calculates the joint torques τ^* that satisfy friction-cone constraints.

5.1.1 *Preview model*

Preview models are low-dimensional representations that describe and capture different locomotion behaviors, such as walking and trotting, and provide an overview of the motion [56]. Reducing the dimensionality of the optimization problem we can generate complex locomotion behaviors and their transitions. This is more suitable for rough terrain as it simplifies the problem landscape. In the literature, different models that capture the legged locomotion dynamics have been studied [21][64] such as point-mass, inverted pendulum, cart-table, or contact wrench.

Our preview model decouples the CoM motion from the trunk attitude[1] (Fig. 5.2). For the CoM motion, we use the cart-table template [37]. The cart-table model (linear inverted pendulum) encompasses a point mass assumption which has no angular momentum. However, to control the attitude we need to apply a torque to the CoM. High centroidal moments (e.g. due to high trunk angular acceleration) can hamper the postural stability condition (e.g. causing shifts on the CoP that can move it out of the support polygon [70]) making the robot losing its capability to balance. Consequently, for the attitude planning, we limit the maximum moments applied to the CoM by limiting the maximum angular acceleration and setting a correspondent margin for the CoP on the support polygon (Section 5.1.1.2).

1 In this thesis, with *trunk attitude* we refer to roll and pitch only.

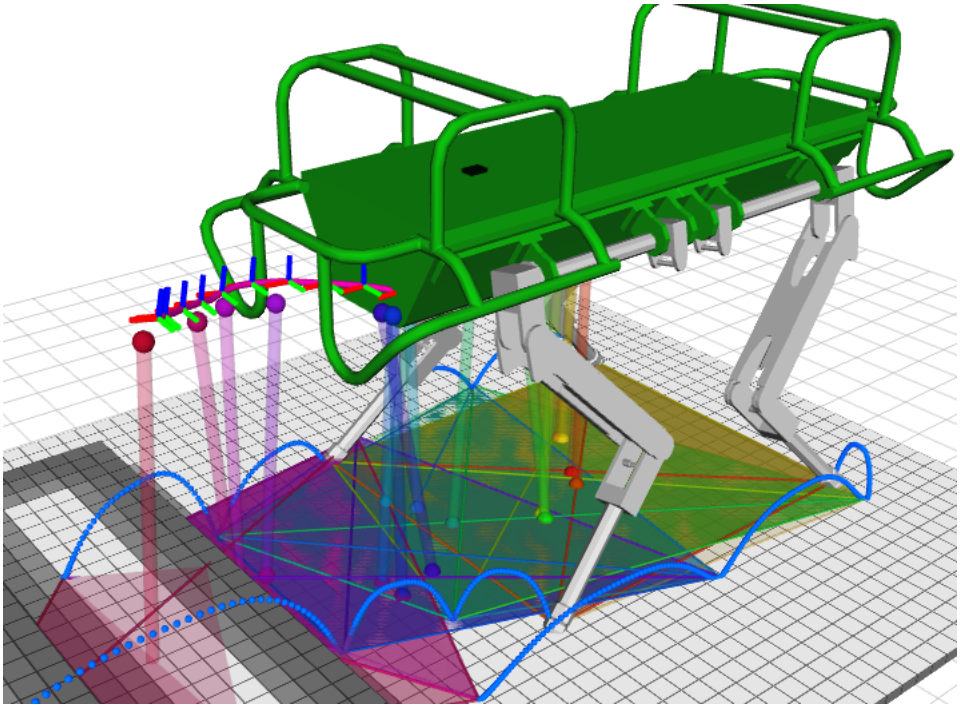


Figure 5.2: A trajectory obtained from a low-dimensional model given a sequence of optimized control parameters and the terrain heightmap. The colored spheres represent the CoM and CoP positions of the terminal states of each motion phase. The CoP spheres lie inside the support polygon (same color is used). Note that color indicates the phase (from yellow to red). The trunk adaptation is based on the estimated support planes in each phase. Since the control parameters are expressed in the horizontal frame [1], the horizontal CoM trajectories and the trunk attitude are decoupled.

5.1.1.1 *CoM motion*

In our previous work [86], we showed that for fixed step durations, the CoP movement is approximately linear, i.e.:

$$\mathbf{p}^H(t) = \mathbf{p}_0^H + \frac{\delta \mathbf{p}^H}{T} t. \tag{24}$$

Note that $\mathbf{p}^H = (x^H, y^H) \in \mathbb{R}^2$ is the horizontal CoP position, $\delta \mathbf{p}^H \in \mathbb{R}^2$ the horizontal CoP displacement and T is the phase duration.

Applying this linear control law in the cart-table model, we derive an analytic solution for the horizontal dynamics [56]:

$$\mathbf{x}^H(t) = \beta_1 e^{\omega t} + \beta_2 e^{-\omega t} + \mathbf{p}_0^H + \frac{\delta \mathbf{p}^H}{T} t, \tag{25}$$

where the model coefficients $\beta_{1,2} \in \mathbb{R}^2$ depend on the actual state $\mathbf{s}_0$ (horizontal CoM position $\mathbf{x}_0^H \in \mathbb{R}^2$ and velocity $\dot{\mathbf{x}}_0^H \in \mathbb{R}^2$, and CoP position), the trunk height h, the phase duration, and the horizontal CoP displacement:

$$\beta_1 = (\mathbf{x}_0^H - \mathbf{p}_0^H)/2 + (\dot{\mathbf{x}}_0^H T - \delta \mathbf{p}^H)/(2\omega T),$$

$$\beta_2 = (\mathbf{x}_0^H - \mathbf{p}_0^H)/2 - (\dot{\mathbf{x}}_0^H T - \delta \mathbf{p}^H)/(2\omega T),$$

where $\omega = \sqrt{g/h}$ and g is the gravity acceleration.

5.1.1.2 *Trunk attitude*

A trunk attitude modulation is required when the terrain elevation varies. A simple approach consists of aligning the trunk with respect to the estimated support plane, avoiding that the robot reaches its kinematic limits. On the other hand, adjusting the trunk attitude requires applying a moment at the CoM, and as consequence, the CoP $\mathbf{p} \in \mathbb{R}^3$ will be shifted by a proportional amount $\Delta\mathbf{p}$ (for more details see Eq. (28) in [70]):

$$\begin{aligned} \Delta p_x &= -\tau_{com_y}/mg, \\ \Delta p_y &= \tau_{com_x}/mg, \end{aligned} \tag{26}$$

where τ_{com_y}, τ_{com_x} are the horizontal components of the moment about the CoM. By exploiting a simplified flywheel model for the inertia of the robot we can link these moments to the CoP displacement $\Delta\mathbf{p}$ (rewritten in vectorial form) and to the angular acceleration $\dot{\omega}$:

$$\boldsymbol{\tau}_{com} = \mathcal{I}\dot{\boldsymbol{\omega}}, \tag{27}$$

$$\Delta\mathbf{p} = \boldsymbol{\tau}_{com} \times m\mathbf{g}. \tag{28}$$

where $\mathcal{I} \in \mathbb{R}^{3\times3}$ is the time-invariant inertial tensor approximation of the centroidal inertia matrix of the robot. Therefore, we can guarantee the CoP condition by limiting the angular accelerations $\dot{\omega}_{max}$ (i.e. the allowed applied moments) and setting a corresponding safety margin r on the support polygon (as in Section 4.2.1) in our optimization (Section 5.2.3) as:

$$r = \|(\mathcal{I}\dot{\boldsymbol{\omega}}_{max}) \times m\mathbf{g}\|. \tag{29}$$

Therefore, we adapt the trunk attitude in such a way that it does not affect the CoP condition (i.e. by using the maximum allowed angular acceleration $\dot{\omega}_{max}$). Note that we compute $\dot{\omega}_{max}$ given the stability margin r (i.e., the support polygon margin).

We employ cubic polynomial splines to describe the trunk attitude motion (pitch and roll). The attitude adaptation can be done in different phases. For instance, we can compute the required angular accelerations given the phase duration and guarantee that it does not exceed the allowed angular accelerations. The trunk height is computed given the estimated support plane and we keep it constant along one phase.

5.1.2 *Preview schedule*

Describing quadrupedal locomotion can be achieved through a sequence of different preview models — a preview schedule. Using this, the robot can automatically

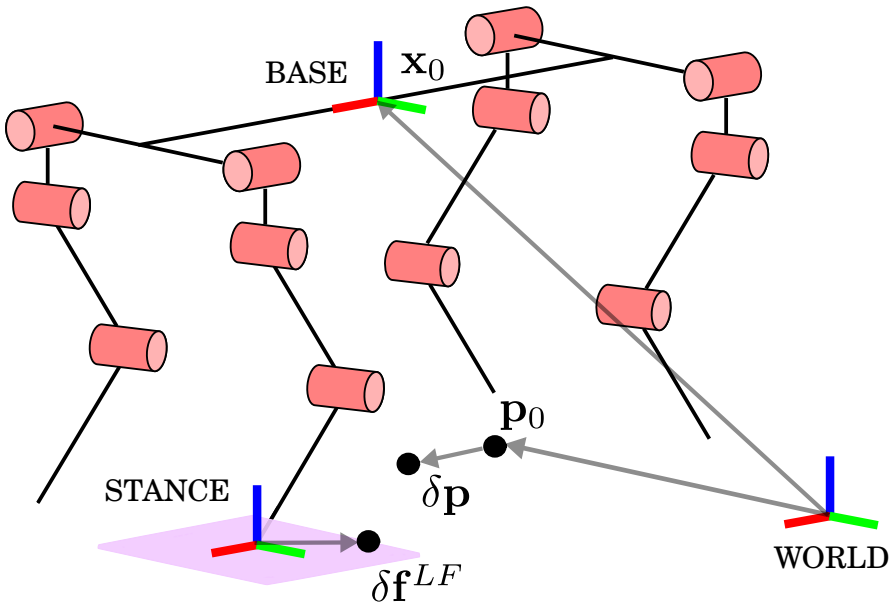


Figure 5.3: Sketch of different variables and frames used in our optimization. The footshift $\delta\mathbf{f}^{LF}$ is described w.r.t. the stance frame. The stance frame is calculated from the default posture and expressed w.r.t. the base frame.

discover different foothold sequences by enabling or disabling different phases in our optimization process.

In the preview schedule, we build up a sequence of control parameters that describes the locomotion action of the n phases:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1^{s/f} & \cdots & \mathbf{u}_n^{s/f} \end{bmatrix}, \tag{30}$$

where $\mathbf{u}_i^s = \begin{bmatrix} T & \delta\mathbf{p}^{H^T} \end{bmatrix}^T$ and $\mathbf{u}_i^f = \begin{bmatrix} T & \delta\mathbf{p}^{H^T} & \delta\mathbf{f}^{l^T} \end{bmatrix}^T$ are the preview control parameters for the stance and step phases, respectively. Additionally, the footshift $\delta\mathbf{f}^l$ is described w.r.t. the stance frame (Fig. 5.3), which is calculated from the default posture of the robot. Note that n is the number of phases, and l is the foot index.

We describe a dynamic walking gait as a combination of 6 different preview phases or timeslots (i.e. $n = 6$) where 4 of them are step phases. Our combination of phases is stance, LH swing phase, LF swing phase, stance, RH swing phase and RF swing phase[2]. With this fixed preview schedule, we can describe different walking patterns by assigning a zero duration to a specific phase ($T_i = 0$).

5.2 Trajectory Optimization

The trajectory optimization step computes an optimal sequence of control parameters $\mathbf{U}^*$ used for the generation of the low-dimensional trajectories (Section 5.1). We

2 The robot is in stance phase when all the feet are on the ground.

formulate this as a receding horizon trajectory optimization problem, where the current timeslot is optimized while taking future timeslots into account. The horizon is described by a predefined number of preview schedules N with n timeslots or phases (e.g. our locomotion cycle has 6 timeslots). Considering future phases presents several advantages for rough terrain locomotion. It enables us to generate desired behaviors that anticipate future terrain conditions, and it results in smoother transitions between phases.

In our approach, the optimal solution at the current phase i comprises of a set of control parameters $\mathbf{u}_i^*$ describing the duration of phase T_i^*, the CoP displacement $\delta\mathbf{p}^{H_i^*}$, and the footshift $\delta\mathbf{f}_i^*$ of the corresponding phase. We define the footshift in the nominal stance frame which corresponds to the default posture. Note that there are phases without foot swing. To the best of our knowledge, our approach is the first that jointly optimizes phase duration and foothold selection, while considering terrain conditions. This contribution has been presented in [52].

5.2.1 *Receding horizon planning*

Given an initial state $\mathbf{s}_0$, we optimize a sequence of control parameters inside a predefined horizon, and apply the optimal control of the current phase. We find the sequence of control parameters $\mathbf{U}^*$, through an unconstrained optimization problem, given the desired user commands (trunk velocities):

$$\mathbf{U}^* = \underset{\mathbf{U}}{\operatorname{argmin}} \sum_j \omega_j g_j(\mathbf{S}(\mathbf{U})), \tag{31}$$

where $\mathbf{S} = \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_{Nn} \end{bmatrix}$ is the sequence of preview states. The preview state is defined by the CoM position and velocity $(\mathbf{x}, \dot{\mathbf{x}})$, CoP position $\mathbf{p}$ and the stance support region $\mathbf{F}$, i.e. $\mathbf{s} = \begin{bmatrix} \mathbf{x} & \dot{\mathbf{x}} & \mathbf{p} & \mathbf{F} \end{bmatrix}$, where $\mathbf{F} = \begin{bmatrix} \mathbf{f}_1 & \cdots & \mathbf{f}_j \end{bmatrix}$ is defined by the position of the active feet $\mathbf{f}_j$. We solve the trajectory optimization using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [25]. CMA-ES is capable of handling optimization problems that have multiple local minima, such as those introduced by the costmap and the phase duration. In the description of our optimization problem, we use soft-constraints as these provide the required freedom to search in the landscape of our optimization problem. The cost functions and soft-constraints $g_i(\mathbf{S})$ describe: *1*) the user command tracking with step duration and length, and travel direction, *2*) the CoM energy, *3*) the terrain cost, *4*) stability soft-constraint, i.e. the CoP condition, and *5*) the preview model soft-constraint, i.e. the linear inverted pendulum.

5.2.2 *Cost functions*

We encode the desired body velocity from the user by mapping it into a 'default' step duration and length. Additionally, the CoM trajectory should accelerate as little as possible during the phases. Note that this implicitly reduces the required

joint torques. We evaluate the step duration and length in every locomotion phase i as follows:

$$g_{step-duration} = \left(\sum_{i=1}^{Nn} (t_i - i\, T_{step}) \right)^2, \tag{32}$$

$$g_{step-length} = \left(\sum_{i=1}^{Nn} (d_i - i\, d_{step}) \right)^2, \tag{33}$$

where t_i is the sum of individual durations until phase i, T_{step} the desired step duration, $d_i = \mathbf{d}^T(\mathbf{x}_i - \mathbf{x}_0)$ is the displacement of the CoM along the desired travel direction expressed in the horizontal frame (defined by the desired yaw angle), and d_{step} is the desired step length.

To encourage movements in the desired travel direction, we penalize lateral drift just in the 4-feet stance phase:

$$g_{step-drift} = \left(\sum_{i=1}^{Ns} d_i^{\perp} \right)^2, \tag{34}$$

where s is the number of 4-feet stance phase per locomotion cycle, $d_i^{\perp}$ is the orthogonal vector of the desired travel direction. Note that step duration and length define the desired linear velocity and the lateral drift defines the desired yaw angular velocity of the trunk. This choice of cost terms encourages equal trunk velocities between all the locomotion phases.

Minimizing the changes in the CoM accelerations reduces the required joint torques. We achieve this by applying:

$$g_{com-energy} = \sum_{i=1}^{Nn} \int_0^{T_i} \|\ddot{\mathbf{x}}\|_2(t)\, dt. \tag{35}$$

To cope with different terrain difficulties, we compute a costmap from an onboard sensor as described in Section 3.2.1. The costmap quantifies how desirable it is to place a foot at a specific location using geometric features such as *height standard deviation*, *slope* and *curvature*. This allows the robot to negotiate different terrain conditions (Fig. 5.4). Thus, given a footshift and CoM position, we compute the foothold location cost as:

$$g_{terrain} = \mathbf{w}^T \mathbf{T}(x, y), \tag{36}$$

where $\mathbf{w}$ and $\mathbf{T}(x, y)$ are the weights and feature values, respectively. We use a cell grid resolution of 4 cm, approximately equal to the robot's foot size, and the terrain features are computed from a voxel resolution of 2 cm (described in Section 3.2.1). As in [54], we demonstrated that this coarse map is a good trade-off in terms of computation time and information resolution for foothold selection. We cannot guarantee convexity in the terrain costmap, which has to be considered in our optimization process.

5.2.3 *Soft-constraints*

As mentioned in Section 5.1.1.2, the CoP trajectory must be kept inside the support polygon which is shrunk by a margin r. This margin guarantees dynamic stability when a maximum moment is applied to the CoM (Section 5.1.1.2). We use a set of nonlinear inequality constraints to describe the support region:

$$\mathbf{l}(\mathbf{F})^{\mathsf{T}} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} > \mathbf{0}, \tag{37}$$

where $\mathbf{l}(\cdot) \in \mathbb{R}^{l \times 3}$ are the coefficients of the l lines, $\mathbf{F}$ the support region defined from the selected foothold locations, and $\mathbf{p}$ the CoP position. Note that the stability constraints are nonlinear as a consequence of adding the foothold positions as decision variables.

Due to the decoupling of the horizontal and vertical motions, we implement a preview model soft-constraint that ensures the cart-table height is approximately equal to:

$$h = \|\mathbf{x} - \mathbf{p}\| \tag{38}$$

where $\mathbf{x}$ and $\mathbf{p}$ are the CoM and CoP positions, respectively. Note that when the cart-table is falling down, the CoM trajectory increases exponentially in (25). This effect arises from the fact that we decouple the horizontal and vertical dynamics, hence adding this soft-constraint guarantees the validity of the model.

To reduce the computation time, we impose both soft-constraints only in the initial and terminal time of each phase as they will be guaranteed in the entire phase. In fact, the linear CoP trajectory will belong to the convex support polygon if the initial and terminal positions are inside this region. We ensure this by limiting the foothold search region, i.e. by bounding the footshift (see Fig. 5.3). These soft-constraints are described as quadratic cost terms.

5.3 Control and Execution

Compared with Section 4.3, the motion of the robot body (CoM and trunk orientation) is controlled by a trunk controller developed by our group [19] that computes the joint torques necessary to achieve the desired motions without violating friction constraints.

At the joint-space level, an impedance controller is acting in parallel to address unpredictable events, such as a foot slippage on an unknown surface. This controller receives a set-point which is consistent with the body motion in order to prevent a conflicting target with the trunk controller. In nominal operations the biggest part of the torques is generated by the trunk controller.

The aim for balancing is to control the position of the robot's CoM, and the orientation of the trunk (base link). We compute a desired linear acceleration for

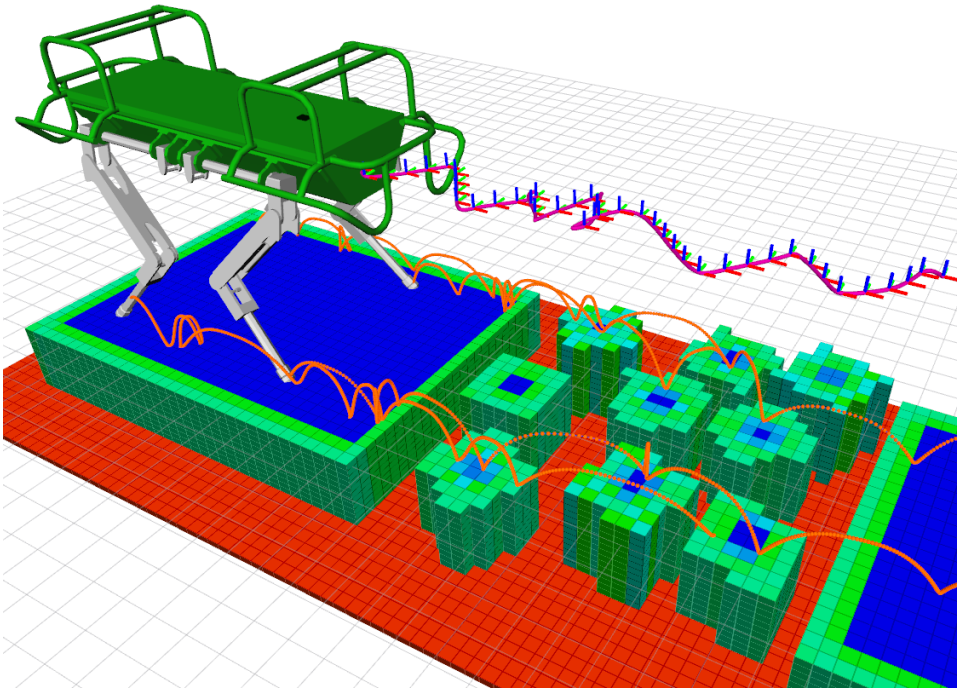


Figure 5.4: A costmap allows the robot to negotiate different terrain conditions while following the desired user commands. The costmap is computed from onboard sensors as described in Section 3.2.1. The cost values are continuous and represented in color scale, where blue is the minimum and red is the maximum cost.

the CoM ($\ddot{\mathbf{x}}^{d}_{com} \in \mathbb{R}^3$) and the trunk angular acceleration ($\dot{\boldsymbol{\omega}}^{d}_{b} \in \mathbb{R}^3$) using a PD control law written in the operational space, i.e. a *virtual model* of the form:

$$\begin{aligned} \ddot{\mathbf{x}}^{d}_{com} &= \mathbf{P}_x(\mathbf{x}^{d}_{com} - \mathbf{x}_{com}) + \mathbf{D}_x(\dot{\mathbf{x}}^{d}_{com} - \dot{\mathbf{x}}_{com}), \\ \dot{\boldsymbol{\omega}}^{d}_{b} &= \mathbf{P}_\theta e(\mathbf{R}^{d}_{b}\mathbf{R}^{\top}_{b}) + \mathbf{D}_\theta(\boldsymbol{\omega}^{d}_{b} - \boldsymbol{\omega}_b), \end{aligned} \tag{39}$$

where $\mathbf{x}^{d}_{com} \in \mathbb{R}^3$ is the desired CoM position, and $\mathbf{R}_b\, \mathbf{R}^{d}_{b} \in \mathbb{R}^{3\times3}$ are the rotation matrices representing the actual and desired orientation of the trunk respectively, $e(.) : \mathbb{R}^{3\times3} \rightarrow \mathbb{R}^3$ is a mapping from a rotation matrix to the associated rotation vector, $\boldsymbol{\omega}_b \in \mathbb{R}^3$ is the angular velocity of the base.

As shown in [65], if the CoM velocity is used as a generalized velocity instead of the base velocity, the robot's dynamic equations get simplified. In this case, we can write the centroidal robot dynamics as in [64]:

$$m(\ddot{\mathbf{x}}_{com} + \mathbf{g}) = \sum_{i=1}^{c} \mathbf{f}_i = \mathbf{F}_{com}, \tag{40}$$

$$\mathcal{I}_G\dot{\boldsymbol{\omega}}_b + \dot{\mathcal{I}}_G\boldsymbol{\omega}_b = \sum_{i=1}^{c}(\mathbf{p}_{com,i} \times \mathbf{f}_i) = \boldsymbol{\Gamma}, \tag{41}$$

where $\mathcal{I}_G$ is the instantaneous centroidal composite rigid body inertia that represents the aggregate rigid body inertia of the entire robot computed at its CoM,

$\mathbf{p}_{com,i} \in \mathbb{R}^3$ is a vector going from the CoM to the position of the i^{th} foot defined in an inertial world frame, c is the number of contact points and $\mathbf{f}_1, \ldots, \mathbf{f}_c \in \mathbb{R}^3$ are the GRFs. Since our platform has nearly point-like feet, we assume that it cannot generate moments at the contacts, but only pure forces. Furthermore, we neglect the term $\dot{\mathcal{J}}_G \boldsymbol{\omega}_b$ since we can assume the legs to be massless. For instance, the leg masses represent the 8% of the robot weight.

Then, the desired wrench $\mathbf{W}^d = [{\mathbf{F}^d_{com}}^T, \boldsymbol{\Gamma}^{dT}]^T$ can be computed from the desired CoM linear and trunk angular accelerations and by rewriting (41) in matrix form, we can then map $\mathbf{W}^d$ into GRFs:

$$\underbrace{\begin{bmatrix} \mathbf{I}_{3\times3} & \cdots & \mathbf{I}_{3\times3} \\ [\mathbf{p}_{com,1}\times] & \cdots & [\mathbf{p}_{com,c}\times] \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_c \end{bmatrix}}_{\mathbf{f}} = \underbrace{\begin{bmatrix} m(\ddot{\mathbf{x}}^d_{com} + \mathbf{g}) \\ \mathcal{J}_G \dot{\boldsymbol{\omega}}^d_b \end{bmatrix}}_{\mathbf{b}}. \tag{42}$$

The redundancy in the mapping yields 6 equations with up to 12 unknowns as we can have up to 4 feet on the ground. Hence, we can form a quadratic optimization problem aiming to satisfy additional optimality criteria, such as ensuring that the ground reaction forces lie inside the friction cones and fulfilling the unilaterality of the GRFs [19]. We approximate the friction cones with square pyramids to express them as linear inequality constraints:

$$\begin{aligned} \mathbf{f}^d = &\underset{\mathbf{f}\in\mathbb{R}^3}{\operatorname{argmin}} (\mathbf{A}\mathbf{f} - \mathbf{b})^\top (\mathbf{A}\mathbf{f} - \mathbf{b}) + \mathbf{f}^\top \mathbf{W}\mathbf{f} \\ &\text{s.t.} \quad \underline{\mathbf{d}} < \mathbf{C}\mathbf{f} < \bar{\mathbf{d}}, \end{aligned} \tag{43}$$

where $\mathbf{f}^\top \mathbf{W}\mathbf{f}$ is a regularization term to keep the solution bounded. We use the QuadProg++ library [23], an open-source QP solver based on an active set algorithm, to solve the optimization in real-time.

In a second step we map the optimal solution $\mathbf{f}^d$ into desired joint torques $\boldsymbol{\tau}^d \in \mathbb{R}^n$ (where n is the number of joints) considering the gravitational/Coriolis contribution $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$:

$$\boldsymbol{\tau}_{ff} = \mathbf{h} - \mathbf{S}\mathbf{J}_c^\top(\mathbf{f}^d), \tag{44}$$

where $\mathbf{J}_c \in \mathbb{R}^{k\times n+6}$ is the stacked Jacobian of the contact points ($k = 3$ is the number of kinematically constrained DoFs) and $\mathbf{S} = \begin{bmatrix} \mathbf{o}_{n\times6} & \mathbf{I}_{n\times n} \end{bmatrix}$ is a matrix that selects the actuated degrees of freedom.

Finally, the trunk controller torques $\boldsymbol{\tau}_{ff}$ are summed with the joint PD torques to form the desired torque command $\boldsymbol{\tau}^d$ that is sent to the low-level joint-torque controllers [5][3]:

$$\boldsymbol{\tau}^d = \boldsymbol{\tau}_{ff} + \mathrm{PD}(\mathbf{q}^d, \dot{\mathbf{q}}^d), \tag{45}$$

where $\mathbf{q}^d \in \mathbb{R}^n, \dot{\mathbf{q}}^d \in \mathbb{R}^n$ are the desired joint positions and velocities, respectively.

5.4 Results

To evaluate our approach, we first validate the trunk attitude modulation (pitch and roll) for dynamic walking on flat terrain. Subsequently, we quantify the capabilities of our framework through a set of different terrain conditions: crossing a gap and a set of sparse stepping stones. For that, we plan and execute dynamic walking behaviors which enable the robot to adapt to different terrain conditions given the high-level user commands (desired trunk velocity: step duration and length, and travel direction).

5.4.1 *Dynamic attitude modulation*

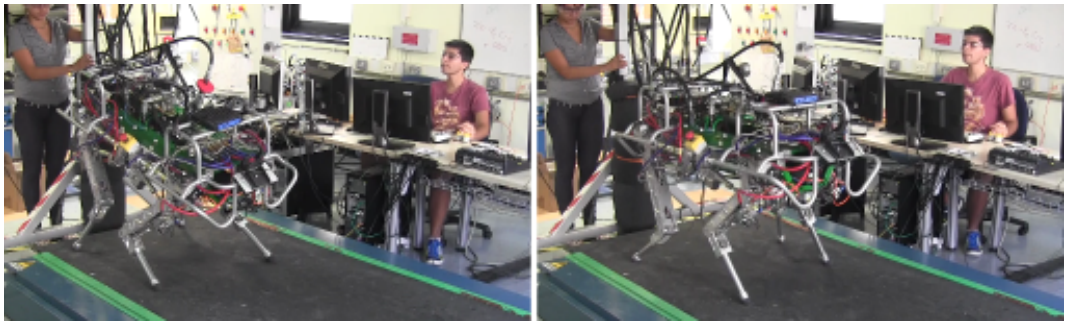
First, we showcase the automatic adjustment of the trunk attitude, during a dynamic walk, as illustrated in Fig. 5.5a. To evaluate the attitude modulation feature, we plan a *fast*[3] dynamic walk with an average body velocity of 0.18 m/s. We use a constant costmap for generating the corresponding footholds, thus the resulting locations come from the dynamics of walking itself, while maximizing the stability of the gait. We define a stability margin of $r = 0.1\,m$ for all our optimizations which is good trade-off between modeling error and allowed trunk attitude adjustment in HyQ. The maximum allowed angular acceleration is computed using the trunk inertia matrix of HyQ, which results in $0.11\,rad/s^2$. Note that the trunk attitude planner uses the maximum allowed angular velocity as explained in Section 5.1.1.2.

The resulting behavior shows HyQ successfully walking while changing its trunk roll and pitch angles. Note that the trunk attitude planner adjusts the roll and pitch angles given the estimated support region at each phase. Fig. 5.5b shows the tracking performance for initial trunk attitude of 0.17 and 0.22 rad in roll and pitch, respectively. In addition, Fig. 5.5c shows the attitude modulation, which is accomplished in the first 6 phases (i.e. one cycle of locomotion).

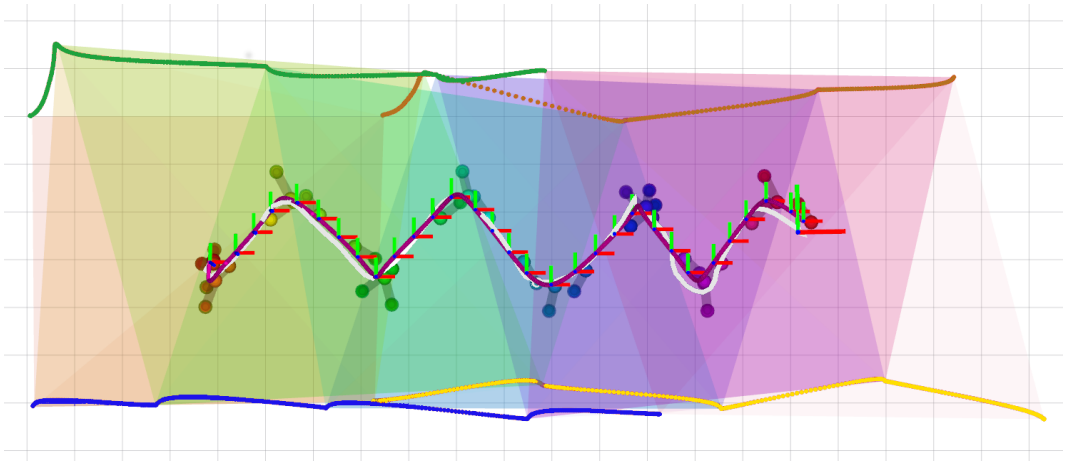
5.4.2 *Locomotion on challenging terrain*

We tested our approach on various challenging terrains: gap and stepping stones with different terrain heights. For all these scenarios, we computed the costmap using the *standard deviation of the height values*, which is estimated through a regression in a $4\,cm \times 4\,cm$ window around the cell of interest. The costmap is built using a resolution of $(4\,cm \times 4\,cm \times 2\,cm)$ in (x, y, z), respectively. The higher resolution value in z reduces the different between the time that is expected the foothold and the detected one. Reducing the foothold error improves the tracking performance of the controller since the desired base and joint positions and velocities are consistent to each other. We weigh equally and manually the desired user command and terrain costs, with a small weight for the CoM energy cost (around 5%). Both soft-constraints have higher weights, which ensures that their targets are met provided

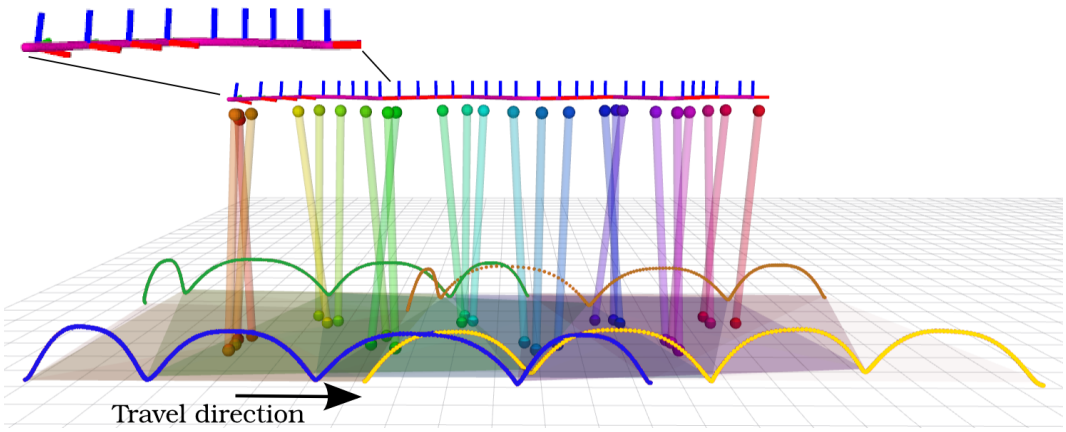
3 *fast* for common walking gait velocities of HyQ



(a) Dynamic walking and trunk modulation



(b) CoM tracking performance



(c) Trunk attitude modulation

Figure 5.5: (a) Dynamic attitude modulation. The initial trunk attitude is 0.17 and 0.22 radians in roll and pitch, respectively. (b) Body tracking when walking and dynamically modulating the trunk attitude. The planned CoM (magenta) and the executed trajectory (white) are shown together with the sequence of support polygons, CoP and CoM positions. Note that each phase is identified with a specific color. (c) A lateral view of the same motion shows the attitude correction (sequence of frames), and the cart-table displacement. Note that we use the RGB color convention for drawing the different frames. In (b)-(c) the brown, yellow, green and blue trajectories represent the LF, RF, LH and RH foot trajectories, respectively.

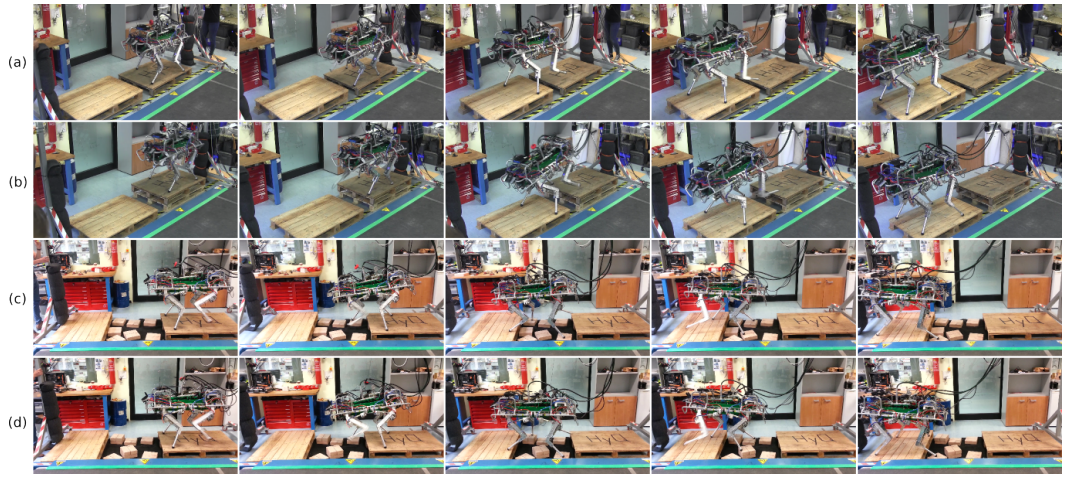


Figure 5.6: Snapshots of experimental trials used to evaluate the performance of our trajectory optimization framework. (a) crossing a gap of 25 cm while climbing up 6 cm. (b) crossing a gap of 25 cm while climbing down 12 cm. (c) crossing a set of 7 stepping stones. (d) crossing a sparse set of stepping stones while dealing with different stone elevations (6 cm)

with enough exploration steps to the CMA-ES solver. We hand-tune the weights in such a way that produce good results in the different terrain benchmarks, and at the same time, that ensure the soft-constraints. Note that we do not need to define an initial guess, and moreover this might not even help the search due to changes in the terrain topology. We used the same stability margin and allowed angular acceleration (as in Section 5.4.1) for the trunk attitude planner, and our horizon is $N = 1$, i.e. 1 cycle of locomotion or 4 steps.

Crossing a gap and/or trunk attitude adaptation tends to overextend the legs, since large motions are required (Fig. 5.6a). To avoid kinematic limits, we defined a foot search region that ensures leg kinematic feasibility up to 12 cm of terrain height difference, as is illustrated in Fig. 5.6b. For instance, we generated a trajectory with two stepping stones 6 cm higher to the other ones. These terrain irregularities produce a trunk modulation in roll and pitch as can be observed in the second sequence. The execution performance on stepping stones with and without changes in terrain elevation is shown in Fig. 5.6c,5.6d. Compared with our previous work [86], we increased the walking velocity by approximately 114%, while also modulating the trunk attitude. Furthermore, the foothold error is on average less than 2 cm, which increases the success rate of the stepping stones trials to 90%; an increment of 30% when compared with our previous work [86]. In Table 5.1, we compare the performance of the coupled planner with the decoupled planner (Chapter 4). The results of averaged speed and success rate comprise the trials with different terrain elevations. Thus, for the latest results, we also increase the complexity of the different terrains. The computation time is approximately around 10 min, much longer than the decoupled planner case (around 2 sec, see Table 4.1).

Table 5.1: Forward speed and success rate of experiments averaged over 10 trials and compared to the decoupled planner results from [86].

	Speed [cm/s]			Success Rate [%]		
Terrain	Curr.	Prev.	Ratio	Curr.	Prev.	Ratio
Step. Stones	15.6	7.3	2.14	90	60	1.5
Gap	14.1	12.7	1.11	78	0	-

This trajectory optimization framework uses as input the desired trunk velocities, which can generate intuitively locomotion policy given the desired velocity from a common user-interfaces, e.g. joystick. Moreover, optimizing a sequence of control parameters will potentially allow us to integrate reactive behaviors, which are important for increasing the robustness of the locomotion.

5.5 Discussion

As in Chapter 4, we have performed a substantial number of trials with the HyQ. We used the similar terrain environments for benchmarking (Chapter 4). In this section we describe the factors that improve the overall performance of the task.

(1) Coupled motion and foothold planning allows us to consider the dynamics for the foothold selection. This is an important factor especially for automatic walking pattern adaptation due to changes in the step duration. Compared with our previous results (Chapter 4), HyQ can cross terrains with only 7 stones (the same pallet distance) which was not able with the previous method (i.e. 9 stones terrain). For instance, in the previous method described in Chapter 4, the body motion cannot adapt dynamically to the allowed terrain regions (i.e. kinematic foothold planning), which reduces the number of possible foothold locations. The coupled planning improves the foothold selection, while it allows the robot to modulate the step duration. Furthermore, HyQ can cross terrains with different elevations since the foothold selection considers the body motion. For example, climbing up or down a gap tends to overextend the legs, since large motions are required (Fig. 5.6a). To avoid kinematic limits, the coupled planning ensures leg kinematic feasibility up to 12 cm of terrain height difference for the defined foot search region with respect to the base frame (as illustrated in Fig. 5.3). Note that the foothold selection always searches in a region where the kinematic constraint are ensured independently of the body velocity. All these factors are important for increasing the capabilities of our locomotion system. In fact, the coupled planning allows HyQ to cross faster, e.g 114% in walking speed for stepping stones.

(2) A linear displacement of CoP per phase produces good practical results, i.e. it describes the CoM movement as expected from the preliminary results of the Chapter 4 (see Fig. 4.15b). This assumption allows us to describe the optimization

problem with a sequence of low-dimensional parameterized models, which are based on the cart-table template. In other words, we can describe the movement as a sequence of control parameters. The main advantage of this model representation is that it can be combined with stochastic-based optimization solvers, thus, it can solve the local minima issue by including the terrain topology and adapting the walking pattern timing. On the other hand, stochastic-based optimization tends to increase the computation time due to the non-convexity nature of the problem. In addition, optimizing a sequence of control parameters is also convenient for learning locomotion policies due to the dimensionality reduction, and for integrating with reactive behaviors such as a step reflex controller for obstacle negotiation [18].

(3) In contrast to Chapter 4, we systematically address the trunk attitude adjustment that ensures dynamic walking. For that, we found a relationship between the applied torques to the CoM and the displacement of the CoP. Later we connected it with the stability margin by assuming a time-invariant inertial tensor approximation of the inertia matrix. Experimental results with HyQ validated this assumption on flat dynamic walking and rough terrain locomotion. Due to this novel method, HyQ increased its locomotion capabilities by crossing different rough terrain with significant terrain elevation changes. Furthermore, this method can be applied to other legged systems such as bipedal ones.

(4) Higher walking speed increases the probability of foot-slippage, when one or some of the feet slip backwards, or when a foot is only slightly loaded so subsequent “pushing" backwards results into foot sliding. Both events are more likely to happen in a terrain with different elevations due to errors in the state estimation or noise in the perception sensors. Including friction-cone constraints in the inverse dynamics torque calculation step has shown to generate movements without foot sliding in the experiments [19]. Nevertheless, state estimation errors and noise in the perception sensors can produce deviation from the planned trajectories, i.e. due to the unexpected contact events. This affects the overall execution performance of the locomotion. As a consequence replanning motions and integrating reactive behaviors can overcome these limitations. Despite of the current limitations, HyQ showed an increase of 114% in the walking speed and a decreased in foothold execution error (from 8 to 4 cm) compared with the results of Chapter 4.

(5) Considering the terrain topology increases the complexity of the trajectory optimization problem. For instance, we cannot guarantee convexity in the terrain model (i.e. terrain costmap). Moreover, optimizing the step duration introduces many local minima in the problem landscape. Imagine that the robot can choose to cross the terrain by selecting proper foothold locations, adapting the step durations, or a combination of both. For solving these issues, we propose a low-dimensional parametrized model which allows us to solve the optimization problem with stochastic-based exploration. We also impose the constraints as soft, which help the exploration. This is probably the biggest limitation of considering terrain topologies in trajectory optimization. Nevertheless, we can tackle these issues by transferring the set of optimized control parameters to a locomotion motor policy. For that, different machine learning algorithms have been proposed in

the literature [50][57]. These techniques are called guide policy search. Note that they combine the advantages of trajectory optimization (i.e. using model for exploring the search space), and policy learning (i.e. the automatic evaluation and self-improvement of the policy).

5.6 Conclusion

In this chapter, we presented a trajectory optimization approach for locomotion on rough terrain that directly uses terrain information. The approach delivers an optimal CoM motion and corresponding optimal foothold locations. Moreover, the solution takes into consideration the trunk attitude modulation required for a dynamic walking. We employ a combination of parametric preview models, stochastic-based exploration and receding horizon planning for successfully crossing over various rough terrain.

We demonstrated how the combination of an impedance controller —which prevents friction cone violations— alongside a trunk controller can compliantly, yet accurately, track the desired whole-body motion. Real-world experimental trials with the HyQ robot crossing over challenging terrain demonstrated the capabilities of our framework. Compared with our previous results [86][54] (Chapter 4), we improved the locomotion without any loss of performance. HyQ walked faster while making trunk attitude adjustments. Moreover, the accuracy of execution was improved, as the error between desired and achieved footholds was reduced from 8 cm to approximately 4 cm. This increased the success rate in the stepping stones by around 30%.

We validated that linear displacement of CoP per phase produces similar results to the Chapter 4. This assumption allows us to describe a movement as a sequence of parameters. Experimental results suggest that combining CoM trajectories and foothold selection produces better solutions in terms of avoiding joint limits (both in position and torques). In fact, the foothold locations help to minimize the CoM energy, thus it reduces the applied joint torques.

Future work includes integrating reactive behaviors, such as haptic triggering of stance and step reflex. The aim is to increase the robustness of the locomotion, for coping with errors in the terrain perception and state estimation. Another extension can be the automatic gait discovery in rough terrain locomotion, i.e. transition from walking to trotting, and vice-versa, while crossing rough terrains. Finally, working towards online planning is a crucial feature for real applications, which takes around 10 min for every task (i.e. gap and stepping stones).

In the next chapter, we propose a trajectory optimization method that considers a set of possible contact forces. This method allows the robot to decide whether or not to establish a contact. In this case we use a 2-DoFs leg of HyQ mounted to a vertical slider) Additionally, it takes into account torque/joint limits, and friction-cone constraints. The future aim of this approach is to produce even more dynamic motions such as jumping and rearing, during which fewer or no legs are in contact.

6

Whole-body Motion Planning with Contact Forces

One of the aspects that was not addressed in the previous motion planning approaches is the contact forces optimization. Both approaches use a predefined sequence of locomotion phases for generating the desired CoM motion. These methods are suitable for specific behaviors such as walking gaits, but they cannot synthesize more general behaviors. If we want to synthesize general behaviors, we need to consider the contact forces too. This problem is often hard to solve since contact forces produce discontinuities in the dynamics. For tackling this challenge, we present a hierarchical trajectory optimization approach for planning dynamic movements with unscheduled contact sequences. First, we find a feasible CoM motion according to the centroidal dynamics of the robot (Section 6.2.1). Then, we refine the solution by applying the robot's full-dynamics model, where the feasible CoM trajectory is used as a warm-start point (Section 6.2.2). To accomplish the unscheduled contact behavior, we use complementarity constraints to describe the contact model, i.e. environment geometry and non-sliding active contacts. Both optimization phases are posed as MPCC. All the material presented in this chapter has been previously published in [53].

In this chapter, we are concerned with finding feasible trajectories for complex tasks, i.e. tasks that require the exploration of different mode sequences through highly-dynamic movements. We choose a set of jumping tasks as examples, as these highlight the ability to explore the dynamical capabilities of the robot (i.e. 2-DoFs leg, for more details see Section 3.1.1) in order to reach goals that are unreachable in a kinematic manner. In fact, the trajectory optimization method computes whole-body motions that achieve goals that cannot be reached in a kinematic fashion.

6.1 Hierarchical Planning

This work was motivated by the observation that most animal and human locomotion behaviors involve dynamic motions through contact interactions. For instance, kangaroos are *dynamic jumpers* that use hopping as the main locomotion strategy. Indeed, in kangaroo locomotion, highly-dynamic movements and contact forces play an important role for finding efficient locomotion trajectories.

Although such dynamic maneuvers are undoubtedly beneficial, planning and execution of these whole-body trajectories is challenging due to the loss of control authority during flight phases and undefined contact events. We tackle it by generating a whole-body trajectory toward a body goal state (desired body height) that

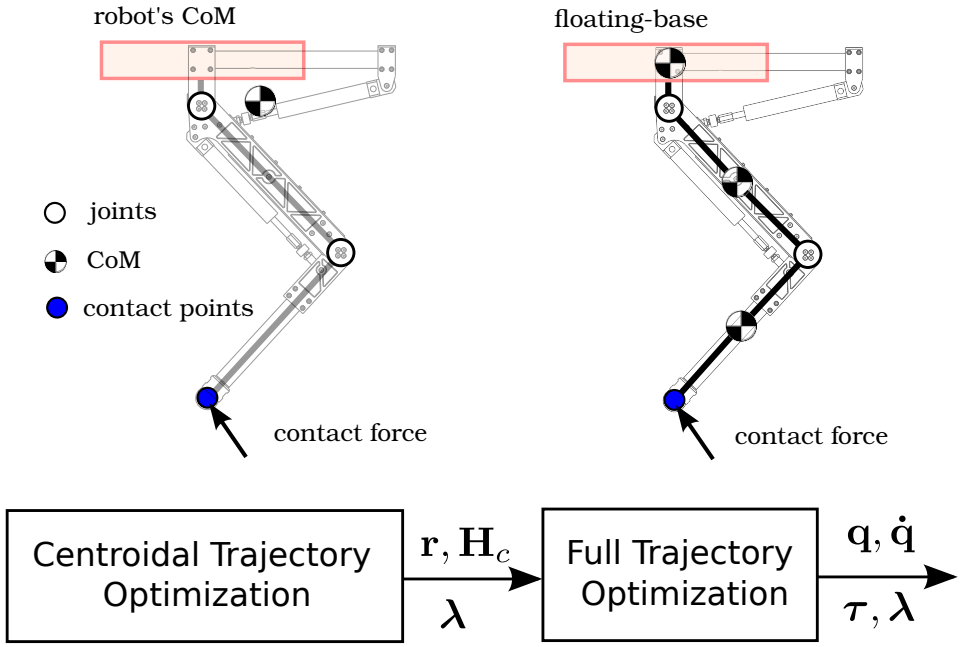


Figure 6.1: The proposed hierarchical trajectory optimization reduces the complexity of the motion planning problem by considering two different optimization phases: centroidal and full trajectory optimization. First, the centroidal trajectory optimization phase produces a locally optimal CoM motion using the centroidal dynamics model [64], which does not consider joint dynamics (i.e. link's CoM). Second, the full trajectory optimization phase refines the CoM trajectory by applying the robot's full-dynamics and joint limits. Both optimization phases use complementarity constraints to model the contact interactions.

ensures a dynamic motion plan through contact interactions. To accomplish this, we describe the contact model using complementarity constraints which defines our approach as a mode-invariant trajectory optimization.

6.1.1 *Generating dynamic motions*

Consider a rigid body system with n degrees of freedom, of which n_b are floating-base degrees of freedom. The state of the robot is represented by its floating-base and actuated joint components, $\mathbf{q} = (\mathbf{q}_b, \mathbf{q}_q)$. Additionally, the robot has p end-effector or contact points.

The system's evolution depends on the internal joint torques, $\boldsymbol{\tau}_q$, and the contact force, $\boldsymbol{\lambda}_j$, applied at the j^{th} end-effector. This evolution is subject to robot and environmental constraints such as: joint limits and environment geometry. Exploring different mode switches (contact events) and dynamic movements could produce unsuccessful locally optimal solutions. We improve the solutions by applying a hierarchical trajectory optimization. In the first phase, we model the system's evolution with the centroidal dynamics, i.e. in the CoM space. Then, we impose joint dynamics and limits using a full-dynamic model. Fig. 6.1 presents an overview of our hierarchical trajectory optimization approach.

6.1.1.1 *Centroidal-dynamic model*

The robot dynamics can be projected at the CoM, i.e. the *centroidal dynamics* of the robot. In a full floating-base system ($n_b = 6$ DoFs), the centroidal-dynamic model describes the rate of change of linear and angular momentum of CoM with respect to the inertial frame of reference [64]. The rate of change of linear and angular momentum is determined by contact forces $\boldsymbol{\lambda}_j$, gravitational force $m\mathbf{g}$ and the motion of the robot's links

$$m\ddot{\mathbf{x}} = \sum_{j=0}^{p} \boldsymbol{\lambda}_j + m\mathbf{g} \tag{46}$$

$$\dot{\mathbf{H}}_c(\mathbf{q}, \dot{\mathbf{q}}) = \sum_{j=0}^{p} (\mathbf{r}_j - \mathbf{x}) \times \boldsymbol{\lambda}_j \tag{47}$$

where m is the total mass of the robot, $\mathbf{x} \in \mathbb{R}^3$ is the CoM position, $\boldsymbol{\lambda}_j \in \mathbb{R}^3$ is the contact force applied at the j^{th} end-effector, $\mathbf{H}_c \in \mathbb{R}^3$ is the centroidal angular momentum and $\mathbf{r}_j \in \mathbb{R}^3$ is the end-effector position. The centroidal angular momentum is computed through the computation of the Centroidal Momentum Matrix (CMM) as defined in [64].

6.1.1.2 *Full-dynamic model*

The full-dynamic model enables us to compute the joint efforts given a whole-body state $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ subject to contact forces $\boldsymbol{\lambda}_j$. We partition the dynamics equation of the robot into the unactuated floating-base DoFs $\mathbf{q}_b$ (n_b equations) and the active robot joints $\mathbf{q}_q$ (n_q equations):

$$\underbrace{\mathbf{H}(\mathbf{q}) \begin{bmatrix} \ddot{\mathbf{q}}_b \\ \ddot{\mathbf{q}}_q \end{bmatrix} + \begin{bmatrix} \mathbf{c}_b \\ \mathbf{c}_q \end{bmatrix} (\mathbf{q}, \dot{\mathbf{q}}) - \sum_{j=0}^{p} \begin{bmatrix} \mathbf{J}_{b_j}^T \\ \mathbf{J}_{q_j}^T \end{bmatrix} \boldsymbol{\lambda}_j}_{\mathbf{b} = \mathrm{ID}(model, \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}_q \end{bmatrix} \tag{48}$$

where $\mathbf{H} \in \mathbb{R}^{n \times n}$ is the floating-base inertial matrix, $\mathbf{c} = (\mathbf{c}_b, \mathbf{c}_q) \in \mathbb{R}^n$ is the force vector that accounts for Coriolis, centrifugal, and gravitational forces, $\boldsymbol{\lambda}_j \in \mathbb{R}^3$ are the ground contact forces at the j^{th} end-effector (i.e. point feet), and their corresponding Jacobian, $\mathbf{J}_j = \begin{bmatrix} \mathbf{J}_{b_j} & \mathbf{J}_{q_j} \end{bmatrix} \in \mathbb{R}^{3p \times n}$ and $\boldsymbol{\tau}_q \in \mathbb{R}^{n_q}$ are the joint efforts that we wish to calculate.

The left-hand term $\mathbf{b} = (\mathbf{b}_b, \mathbf{b}_q)$ is computed efficiently using the Featherstone implementation of the RNEA [17].

6.1.2 *Contact model*

In dynamic movements, contact forces play an important role, e.g. a jumping or hopping task. Traditional approaches compute trajectories given a predefined contact sequence. These approaches do not exploit the fact that an optimized mode switching could be required for the success of a certain task.

A contact event occurs when a signed distance to the surface is strictly zero, and additionally, there is a contact force acting along the surface normal. Moreover, a null normal contact force is expected when the contact is inactive, i.e. a positive signed distance. In other words, normal contact forces and signed distances are orthogonal and positives functions (49). Additionally, we desire that active contacts do not slide. Such condition implicates an orthogonality between normal contact forces and tangential velocities (50). In the optimization literature [77], constraints with combinatorial nature, such as the set of contact model equations (49)(50), can be described as complementarity constraints

$$0 \leqslant \lambda_j^{\hat{n}} \perp \phi_j(\mathbf{q}) \geqslant 0 \tag{49}$$

$$\mathbf{0} \leqslant \lambda_j^{\hat{n}} \perp \dot{\mathbf{r}}_j^{\hat{t}}(\mathbf{q}, \dot{\mathbf{q}}) \geqslant \mathbf{0} \tag{50}$$

where $\lambda_j^{\hat{n}}$ is the contact force acting along the surface normal at the j^{th} end-effector (i.e. contact point), $\phi_j(\mathbf{q})$ is the signed distance between the j^{th} contact point $\mathbf{r}_j$ and the surface $\mathcal{S}_i$, and $\dot{\mathbf{r}}_j^{\hat{t}}(\mathbf{q}, \dot{\mathbf{q}})$ is the velocity of the contact point along the tangential surface. Contact-point positions and velocities are calculated efficiently using spatial algebra.

6.2 Trajectory Optimization

Planning problems without scheduled contact sequences are often hard to solve since the contact forces produce discontinuities in the dynamics. Here, we tackle this issue by applying a hierarchical trajectory optimization scheme, which uses different dynamic models in a two-phase manner. Using a different (simpler) dynamic model in the first optimization phase, we impose a dynamic relaxation, that helps to explore different mode switches. Thus, we find a feasible CoM motion in terms of the robot's centroidal dynamics. Then, we refine it by applying the full-dynamic model in the second, more complex, trajectory optimization phase.

6.2.1 *Centroidal trajectory optimization*

The centroidal trajectory optimization step computes a feasible CoM trajectory through the mapping of contact forces inside the centroidal dynamics. The CMM maps the robot's generalized velocities to its spatial momentum (for more details see [64]). We sample the trajectory in N knot-points with a fixed-time duration h. In this optimization phase, the decision variables of the optimization problem are the robot position $\mathbf{q}$, the robot velocity $\dot{\mathbf{q}}$, the CoM position $\mathbf{x}$, the CoM velocity $\dot{\mathbf{x}}$, the contact forces $\boldsymbol{\lambda}$, and the end-effector (contact) positions $\mathbf{r}$. The cost function evaluates the trajectory in terms of the desired high-level goal of the task $\mathbf{w}(\mathbf{q})$ as:

$$\min_{\substack{\mathbf{q}[k],\dot{\mathbf{q}}[k],\mathbf{x}[k],\dot{\mathbf{x}}[k],\\ \mathbf{H}_c[k],\dot{\mathbf{H}}_c[k],\boldsymbol{\lambda}[k],\mathbf{r}[k]}} h \sum_{k=1}^{N} \left(\|\mathbf{w}(\mathbf{q}[k]) - \mathbf{w}(\mathbf{q}^*[k])\|_{\mathbf{Q}_q} \right) \tag{51}$$

where $\mathbf{w}(\mathbf{q})$ constructs a task-specific value from relevant features of the task, and $\|\mathbf{w}(\mathbf{q}[k]) - \mathbf{w}(\mathbf{q}^*[k])\|_{\mathbf{Q}_q}$ computes its associated quadratic cost given a desired robot position $\mathbf{q}^*$. Note that $\|\mathbf{x}\|_{\mathbf{Q}}$ is an abbreviation for the quadratic cost $\mathbf{x}^\top \mathbf{Q}\mathbf{x}$.

We transcribe the centroidal dynamics differential equations (47) to algebraic ones by applying an Euler-backward integration rule with a fixed-time step h

$$\mathbf{x}[k-1] - \mathbf{x}[k] + h\dot{\mathbf{x}}[k] = \mathbf{0} \tag{52}$$

$$\mathbf{H}_c[k-1] - \mathbf{H}_c[k] + h\dot{\mathbf{H}}_C[k] = \mathbf{0} \tag{53}$$

$$m(\dot{\mathbf{x}}[k] - \dot{\mathbf{x}}[k-1]) - h\left(\sum_{j=0}^{p} \boldsymbol{\lambda}_j[k] + m\mathbf{g}\right) = \mathbf{0} \tag{54}$$

$$\dot{\mathbf{H}}_c[k] - \sum_{j=0}^{p} (\mathbf{r}_j[k] - \mathbf{x}[k]) \times \boldsymbol{\lambda}_j[k] = \mathbf{0} \tag{55}$$

where the centroidal angular momentum is computed from the CMM, $\mathbf{A}(\mathbf{q})$, as is explained in [64], i.e. $\mathbf{H}_c[k] = \mathbf{A}(\mathbf{q}[k])\dot{\mathbf{x}}[k]$. Additionally, we impose contact position constraints in order to describe the contact interactions

$$\mathbf{r}_j[k] - \boldsymbol{\kappa}_j(\mathbf{q}[k]) = \mathbf{0} \tag{56}$$

where $\boldsymbol{\kappa}_j(\cdot)$ is the direct kinematic function which computes the position of the j^{th} end-effector. We also include joint position and velocity limits.

$$\mathbf{q}_q^l \leqslant \mathbf{q}_q \leqslant \mathbf{q}_q^u \tag{57}$$

$$\dot{\mathbf{q}}_q^l \leqslant \dot{\mathbf{q}}_q \leqslant \dot{\mathbf{q}}_q^u. \tag{58}$$

To describe different possible mode switches, we add contact position and velocity constraints. These constraints are described as complementarity constraints as follows

$$\lambda_j^{\hat{n}}[k], \phi_j(\mathbf{q}[k]) \geqslant 0 \tag{59}$$

$$\lambda_j^{\hat{n}}[k]\phi_j(\mathbf{q}[k]) = 0 \tag{60}$$

$$\lambda_j^{\hat{n}}[k]\left(\mathbf{r}_j^{\hat{t}}[k] - \mathbf{r}_j^{\hat{t}}[k-1]\right) = \mathbf{0}. \tag{61}$$

We approximate the contact velocity as contact displacement along the tangential surface. Note that a contact velocity constraint does not guarantee zero displacement between knots.

6.2.2 *Full trajectory optimization*

Once a feasible, and locally optimal, CoM trajectory is computed, we use this CoM trajectory as a warm-start point of the full trajectory optimization phase. We transcribe the full-dynamic model with the same time step value of the centroidal trajectory optimization phase. In this optimization phase, we formulate the problem with the following decision variables: the robot position $\mathbf{q}$, the robot velocity $\dot{\mathbf{q}}$, the joint efforts $\boldsymbol{\tau}_q$ and the contact forces $\boldsymbol{\lambda}$.

In this stage, the cost function also considers the joint effort energy of the movement τ_q as

$$\min_{\substack{\mathbf{q}[k],\dot{\mathbf{q}}[k],\\ \tau[k],\lambda[k]}} h \sum_{k=1}^{N} \left(\|\mathbf{w}(\mathbf{q}[k]) - \mathbf{w}(\mathbf{q}^*[k])\|_{Q_q} + \|\tau_q[k]\|_R \right). \tag{62}$$

We apply the same integration rule to the full-dynamic differential equation (48). Additionally, we add a selection matrix $\mathbf{B}$ in order to impose a null wrench vector to the floating-base:

$$\mathbf{q}[k-1] - \mathbf{q}[k] + h\dot{\mathbf{q}}[k] = \mathbf{0} \tag{63}$$

$$\begin{aligned} &\mathbf{H}[k]\left(\dot{\mathbf{q}}[k] - \dot{\mathbf{q}}[k-1]\right) \\ &\quad + h\left(\mathbf{c}[k] - \sum_{j=0}^{p} \mathbf{J}_j[k]^T \boldsymbol{\lambda}_j[k]\right) - \mathbf{B}\tau[k] = \mathbf{0} \end{aligned} \tag{64}$$

where the contact forces are determined using the complementarity constraints (59)(60)(61).

In the full trajectory optimization phase, we impose position and velocity bounds (57)(58), and additionally joint effort bounds

$$\tau_q^l \leqslant \tau_q \leqslant \tau_q^u. \tag{65}$$

We derive a continuous motion plan, from the N optimized knot-points, using a polynomial interpolation. Both optimization phases model contact interactions using complementarity constraints. In general, optimization problems with complementarity constraints are difficult to solve because constraint qualifications are hard to satisfy. We solve the MPCC using interior point method as this is faster than a Sequential Quadratic Programming (SQP) algorithm when the number of complementarity constraints increases [72]. We use the IPOPT library [83]. We relax the orthogonality between the complementarities, for example $\lambda_j^{\hat{n}}[k]\phi_j(\mathbf{q}[k]) = 0$ is posed as $\lambda_j^{\hat{n}}[k]\phi_j(\mathbf{q}[k]) \leqslant 0$. For more information about different interior point methods see [72].

6.3 Results

We conduct three experimental trials with the HyL robot: jumping task, small step jumping (10 cm of height) and big step jumping (15 cm of height). For each experiment, we specify the goal state of the robot's trunk[1], and the desired final joint position as a terminal cost. The hierarchical trajectory optimizer finds a sequence of footsteps through dynamic movements without a predefined order, which the controller then executes dynamically. We use a PD controller and the planned joint efforts as feedforward inputs. We validate the performance of our framework in 3 different examples as seen in Fig. 6.2, and compare against the full dynamic optimization (Table 6.1) on the same benchmark examples. The first example consists

1 In this chapter, with *robot* we refer to the HyL robot

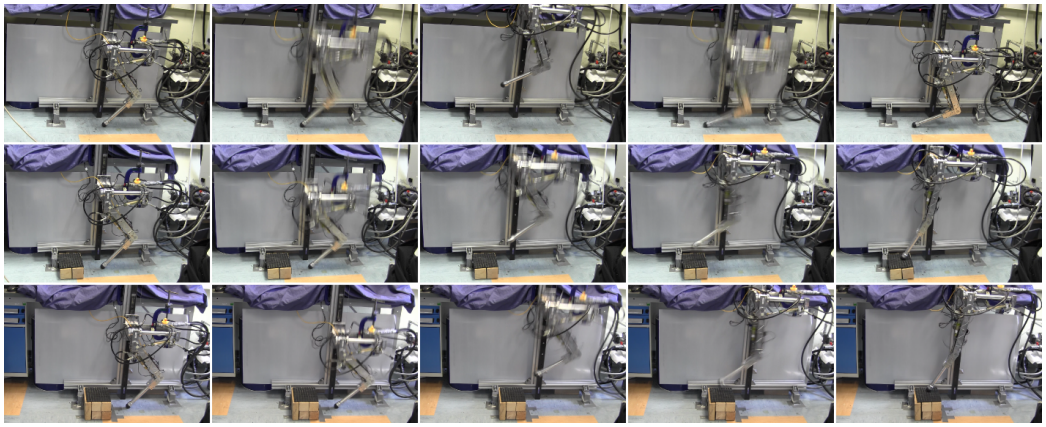


Figure 6.2: Snapshots of three experimental trials with the HyL robot (Section 3.1.1) used to evaluate the performance of our hierarchical trajectory optimization approach. From top to bottom: jumping task; small step jumping (10 cm of height); big step jumping (15 cm of height).

of reaching a goal that is kinematically not feasible, called the *jumping task*. In the next examples, the *step-jumping tasks*, the robot has to reach and keep the desired trunk height, which is done through two different step: a small step (10 cm) and a big step (15 cm).

6.3.1 *Motion through dynamical relaxation*

We focus on finding trajectories that are only feasible when dynamics and contact forces are considered. This motion planning approach describes contact events in a MPCC problem. Since the problem is non-convex, the hierarchical optimization tends to guide the exploration away from infeasible regions through dynamical system relaxation, i.e. centroidal to full dynamics. This dynamical relaxation helps to reduce the computation time and cost value. For instance this approach finds a *counter-movement jump* (a transitory movement). In contrast to the *squat jump*, the counter-movement jump involves a preliminary lift off the ground that maximizes the jump height and minimizes the control energy using the system's inertia (see Fig. 6.3), since this reduces the cost value.

These experiments suggest that dynamical system relaxation is key for finding successful motion plans. Table 6.1 shows the time and cost reduction of our approach compared with a single full trajectory optimization. We can see that the hierarchical optimization approach tends to have better performance in complex tasks. Nevertheless, in general, the central tendency (median) of the computation time reduction is decreased, while, on average, we improve the quality of the solution (cost reduction). In average, the computation time, of the hierarchical trajectory optimization, for the jumping and step jumping tasks are 1 and 8 min, respectively. For this comparison, I define a set of 8 different goal states (i.e. trunk height ranging from 0 to 35 cm) for computing the time and cost reduction of our approach.

Table 6.1: Time and cost reduction over 8 trials compared to a single full trajectory optimization.

	Time reduction [%]		Cost reduction [%]	
Task	Md.	Av.	Md.	Av.
Jumping	10.36	0.0	0.0	1.44
Step Jumping	48.29	30.46	0.0	12.91

6.3.2 *Reaching goals that are kinematically not feasible*

The jumping task demonstrates the ability of exploring the dynamical capabilities of the robot in order to reach goals that are not kinematically possible. In this particular case, we desire to reach with the trunk, a height of 0.85 cm w.r.t. the ground (or 0.27 cm with respect to the initial position), which is kinematically not feasible. Thus, our hierarchical motion planner explores different mode sequences in order to plan a dynamically feasible motion. In Fig. 6.3, the robot plans a counter-movement jump (around 7 cm) without being predefined. In counter-movement jumps, a preliminary downward movement is executed which increases the jump height because the robot is carried by its own inertia. Then, a fast movement of the foot is planned considering a desired task behavior, e.g. joint position in the apex point.

6.3.3 *Discovery of new contacts*

For the success of some tasks, it is crucial to exploit the environmental conditions, e.g. reaching and keeping a desired trunk position that is kinematically not reachable. So, imagine that we want to keep a desired trunk position but due to gravitational forces this is not possible with just a vertical jump. Instead, we need to climb onto an obstacle to accomplish this. With the hierarchical trajectory optimization, we can plan such kind of maneuvers. In fact, Fig. 6.4 shows that this motion planner solves these tasks by defining a foothold on top of an available step. Note that a pre-defined footstep sequence is not required to find such kind of solutions.

6.4 Discussion

We conducted trials with the HyL robot (for more details see Section 3.1.1) performing highly-dynamic and challenging tasks, which demonstrate the capability of the hierarchical trajectory optimization method. A set of jumping tasks were used, as these highlight the ability to explore the dynamical capabilities of the robot and different contact sequences, that cannot be achieved in a kinematic man-

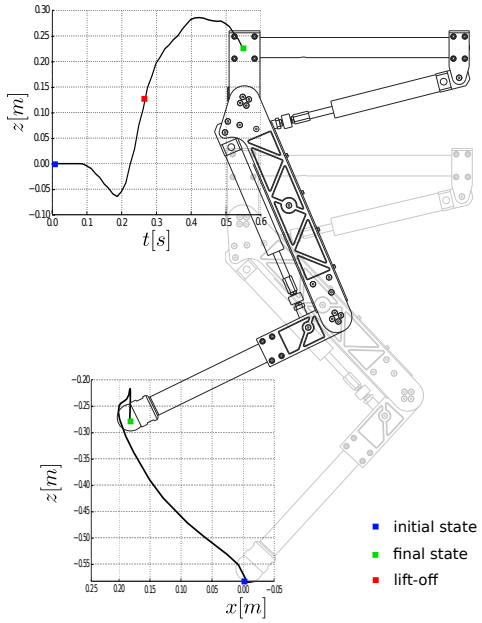


Figure 6.3: Optimized CoM and foot trajectory for a jumping task that shows a dynamic movement through different phases: thrust and flight. We can see that the hierarchical optimization maximizes the jump energy by planning a counter-movement jump, i.e. a preliminary downward movement.

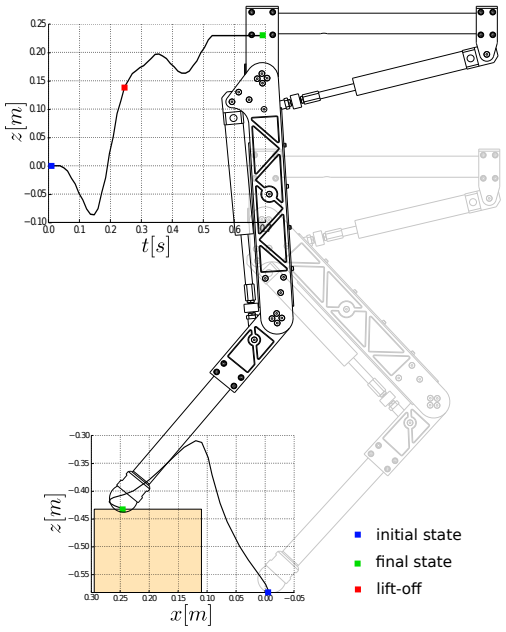


Figure 6.4: Optimized CoM and contact sequence for reaching and keeping a desired trunk position (big step jumping), which is not reachable with just a vertical motion. The hierarchical trajectory optimization finds a transitory foothold in order to keep the desired trunk position. After landing in the planned foothold, the trunk moves up until the desired goal.

ner. In this section we briefly discuss the improvements of the novel hierarchical trajectory optimization compared with state-of-the-art approaches.

(1) Generating automatic behaviors with a predefined schedule is a hard problem due to the high-dimensionality of the search space and the nonconvexity of the problem. The hierarchical trajectory optimization improves the quality of the solution and computation time. In the centroidal trajectory optimization step, the main goal is to find a trajectory that is feasible in terms of centroidal dynamics and contact forces. Next, the full trajectory optimization step ensures that the robot is able to execute that movement by considering the robot's joint limits (i.e. position, velocity and torques limits). Since we ensure the robot's joint limits, we can validate this method in the real system unlike previous works [60][9][22].

(2) The contact model, through complementary constraints, adds discontinuities in the optimization problem. When a contact event changes its state (i.e. active to inactive, or vice-versa), we cannot compute the gradient. Therefore, the problem cannot be solved using gradient-based solvers such as interior-based methods (e.g. IPOPT [83]) or sequential quadratic methods (e.g. SNOPT [24]). In practice this limitation means that the trajectory optimization requires to define good warm-start point and smoothing in those transitions. In other words, we are constrained to describe a rough path of the movements, which limits the applicability of these techniques. In fact, previous works have addressed this issue by smoothing the contact forces, inside the trajectory optimization [60] or as numerical relaxation to the complementary constraints [77], and exploiting reduced models [9]. Nevertheless, those works have been validated just in simulation. In our hierarchical trajectory optimization, we combined both ideas in order to discover new contact interactions, and at the same time, guarantee the joint limits of the system. The *step-jumping* task shows the capability of the hierarchical trajectory optimization to figure out different contact sequences. Compared with a simple jumping task, the former requires more computation time (around 8 times, i.e. 8 min compared with 1 min), and easier to find unfeasible solutions, due to contact switching requires to run a number of iteration that relax the contact constraints. Note that the jumping task has less contact switches than the step jumping tasks.

(3) We demonstrated how a hierarchical trajectory optimization improves the computation time and quality of the solution (i.e. it reduces the cost value) in simpler robot than HyQ. In the literature, several works demonstrated the capabilities either of the full trajectory optimization [71] or the centroidal dynamic optimization [8] in higher-dimensional systems. In those approaches, the authors need to predefined a good enough warm-start point (i.e. initial trajectory of the movement). This limits the applicability of the above-mentioned motion planners. We believe that our hierarchical trajectory optimization method will similarly improve the solution and computation time in higher-dimensional system such as HyQ. With this motion planner, we expect that HyQ will be able to jump long gaps, and to plan aggressive maneuvers such as rearing. Furthermore, including contact forces allows the planner to make contact in other parts of the robot, e.g. climbing up an obstacle using the knees.

(*4*) In general, trajectory optimization produces motion plans for a fixed sequence of points, N knot-points. The continuous motion plan is obtained by applying a polynomial interpolation. However, the polynomial interpolations cannot predict accurately changes in the contact forces, which generally happen in less than 10 ms. Reducing the step integration will approximate more accurate the robot's dynamics but it will increase considerably the computation time. On the other hand, a longer step integration might increase the complexity of the problem due to the model inaccuracies

6.5 Conclusion

In this chapter we presented a hierarchical trajectory optimization approach for planning dynamic movements through unscheduled contact sequences. First, the hierarchical trajectory optimization finds a feasible CoM motion according to the centroidal dynamics of the robot. Then, a second phase of optimization considers the full-dynamics of the robot. In both phases a set of complementarity constraints model the contact interaction. We demonstrated that, with this approach, the robot can plan a different movements that consider the full-dynamics and joint effort limits of the robot. We believe that these considerations are crucial for highly-dynamic locomotion tasks, i.e. step-jumping tasks that cannot be accomplished in a kinematic fashion. For instance, a kinematic model cannot compute a motion plan that ensures the joint torques limits of the robot. It is shown how the hierarchical trajectory optimization improves the solutions and significantly reduces the computation time, compared with the full dynamic optimization. Experimental trials with a robotic leg performing highly-dynamic and challenging tasks demonstrate the capability of this planning approach.

Future works include generation of motion plans given a library of synthesized motions, improving the quality of the solutions, and permitting online computation and execution.

7

Conclusion and Future Work

In this thesis we introduced a new framework for dynamic legged locomotion over challenging terrain. We developed a decoupled motion planner, a foothold planner that selects foothold positions from a terrain costmap, and a motion planner that ensures dynamic stability. We identified the weak points of decoupled planning techniques. We proposed two coupled motion planners that tackle the identified limitations of the decoupled motion planning. Each motion planner method aims to increase the mobility of legged robots over challenging terrains. Nevertheless, this increase in mobility produces an increase in complexity too. In this chapter we summarize the results of these findings and present ideas and directions for future work.

7.1 Conclusion

In this thesis we started by presenting a new dynamic whole-body locomotion framework for rough terrain locomotion. In this framework the motion planner orchestrates the perception and execution modules. The perception module quantifies how desirable it is to place a foot at a specific location by building a terrain costmap and terrain heightmap. The execution module sends precise torque commands that track accurately and compliantly the desired motions. We plan separately the motion and the foothold sequence. We showed how the foothold planner computes online and onboard ($\sim 0.5\,\mathrm{Hz}$) a sequence of footholds. We proposed a whole-body motion planner that ensure dynamic stability despite irregular swing-leg sequences generated by the foothold planner. We demonstrated how the robot executed accurately, yet compliantly, the desired whole-body motions by combining a virtual model and floating-base inverse dynamic controllers. We presented the performance of our framework in real-world experimental trials.

Next, we brought the kinematic foothold planning and the dynamic execution closer together. The goal was to produce desired state trajectories and footholds through a unified trajectory optimization problem (i.e. coupled planning) that takes into consideration the robot's dynamic and terrain topology. This approach delivers an optimal CoM motion and corresponding optimal foothold locations. We decoupled the horizontal and vertical dynamics by ensuring that the trunk attitude adjustments will not invalidate the CoP condition. In our trajectory optimization method, we employed a combination of parametric low-dimensional models, stochastic-based exploration and receding horizon planning. This method allows us to tackle the nonconvexities as a consequence of optimizing the stepping duration and considering the terrain topologies (i.e. terrain costmap). Note that state of the art stochastic-based exploration can solved low-dimensional optimization

problems. We showed how the robot can cross various terrains with an increase in the complexity compared with the decoupled planner. In fact the terrain has few safe regions to step and different elevations. We demonstrated that a linear displacement of CoP per phase produces good practical results. This increased the success rate of the stepping stones trials to 90%, an increment of 30%, compared to the decoupled motion planner. Finally, we demonstrated how the combination of an impedance controller, that ensures the friction cone constraints, alongside a trunk controller can compliantly, yet accurately, track the desired whole-body motion. With this execution performance, we reduced the error between desired and achieved footholds locations (from 8 cm to approximately 2 cm), and increased the achievable walking velocities (from 0.12 m/s to 0.18 m/s). In fact this increased the success rate in the stepping stones by around 30%.

The decoupled and coupled motion planners are able to generated specific behaviors such as the walking gait. However, some terrain conditions cannot be successfully crossed with a pre-specified behavior/gait. Hence it is required to synthesize more general behaviors. In such cases, we need to consider the contact forces, but they are often hard to solve due to discontinuities. Thus, we proposed a hierarchical trajectory optimization approach for planning dynamic movements through unscheduled contact sequence. Our hierarchical trajectory optimization computes a feasible CoM motion that satisfies the robot's centroidal dynamic. Later, we ensure the robot's joint limits constraints by optimizing, in a second step, the robot's motion using the full dynamic of the system. We showed that our hierarchical trajectory optimization increases the range of planned movements, and yet ensures the robot's joint limits (cost reduction until 12.9% compared to a single full trajectory optimization). We also demonstrated that our method reduces the computation time upto 48.3% compared to a single full trajectory optimization. To our best knowledge, our trajectory optimization, with unscheduled contact sequence, method is the first that has been validated in a real robot.

In this thesis, we proposed a set of different motion planning methods for dynamic whole-body locomotion on challenging terrain. We studied the advantages and disadvantages of coupled and decoupled motion and foothold planning. In our planners, we built a unified method for quantifying the terrain difficulty (i.e. terrain costmap). We showed that our terrain model is suitable for decoupled and coupled planning, as well as for graph searching or trajectory optimization. We showed that reduced models (such as cart-table and the contact wrench) allow us to better formulate the trajectory optimization while also considering the terrain topology. We demonstrated that coupled planners increase the locomotion capabilities, but also the computation time. Finally, we showed that contact forces play an important role for synthesizing a wider range of motions. These motion planners provide the ability to increase the mobility of legged machines in challenging terrain.

Previous research in rough terrain locomotion have mainly focused on generating reactive behaviors that tackle small terrain changes or selecting kinematically foothold locations. On the other hand, legged motion planning community focused on planning complex whole-body behaviors but assuming flat terrain con-

ditions. In contrast, we consider that navigating over challenging domains requires taking into account future terrain conditions and also generating complex whole-body behaviors (i.e. motion planning). For that, our motion planners have to consider the terrain conditions as well as the robot's dynamics. Thus, this thesis contributed to close the gap between locomotion approaches and motion planning methods. With this, we increased the legged locomotion capabilities in rough terrain.

7.2 Future work

The work presented in this thesis was focused on increasing the locomotion capabilities of legged machines. Both the coupled motion and foothold planner and the whole-body motion planner with contact forces cannot compute trajectories online due to the nonconvexity of these optimization problems. Re-planning can deal with unexpected terrain changes and execution errors, thus increasing the success rate of the task. In future work, we would like to develop a method that can generate plans online. A promising direction is to transfer a set of optimized trajectories into a control policy. Several machine learning algorithms have been proposed in the literature, in the form of guided policy search. With these methods, we will aim to interactively generate complex behaviors that incorporate high-dimensional sensory modalities such as vision (i.e. the terrain costmap).

The coupled planner optimizes a sequence of control parameters. In fact, a walking gait can be described by a set of parameters and not a trajectory. The description through parameters helps us to integrate reactive behaviors such as haptic triggering of stance and step reflex into planned motions. We recognize the importance of having reactive behaviors, even in planned motions, for overcoming unexpected events. Future work will aim to increase the robustness of the locomotion. For instance, these reactive behaviors could stabilize the robot in cases of unexpected terrain changes that cannot be perceived (e.g. terrain changes under the legs). Additionally, reactive behaviors could increase the robot stabilize in unstable and dynamic terrain or even against push forces.

We showed that a terrain costmap is a suitable model for trajectory optimization. It allows the robot to properly select foothold locations. We used different terrain features (i.e. slope, curvature, height deviation, etc.) for computing an associated cost value. However, there is a remarkable difficulty in quantifying how desirable a footstep location (i.e. cost value) is. One exciting future work will be to self-improve the terrain model by exploiting the robot's experiences. In an ideal scenario the robot plans a motion according to its understanding of the terrain difficulty. Then after the execution, the robot updates/improves its understanding of the terrain difficulty according to this experience. With this approach, we aim to be able to build a terrain model suitable for all the possible topologies.

There are few hardware limitations that decreases the performance of our motion planners. For instance, the robot's torque limits and range of motion limits the different possible maneuvers in stairs climbing and gap crossing with different elevations. Furthermore, the lack of foot force sensors compels us to estimated

the GRFs, which reduces significantly the execution of the planned motions, especially in cases of trunk attitude adaptation. Including these sensors will improve the overall performance in the execution of the planned motions.

Bibliography

[1] Barasuol, V., Buchli, J., Semini, C., Frigerio, M., De Pieri, E. R., and Caldwell, D. G. (2013). A Reactive Controller Framework for Quadrupedal Locomotion on Challenging Terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[2] Bloesch, M., Hutter, M., Hoepflinger, M., Leutenegger, S., Gehring, C., Remy, D., and Siegwart, R. (2012). State Estimation for Legged Robots - Consistent Fusion of Leg Kinematics and IMU. In *Robotics: Science and Systems Conference (RSS)*.

[3] Boaventura, T., Focchi, M., Frigerio, M., Buchli, J., Semini, C., Medrano-Cerda, G. A., and Caldwell, D. G. (2012a). On the role of load motion compensation in high-performance force control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[4] Boaventura, T., Medrano-Cerda, G. a., Semini, C., Buchli, J., and Caldwell, D. G. (2013). Stability and performance of the compliance controller of the quadruped robot HyQ. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1458–1464. Ieee.

[5] Boaventura, T., Semini, C., Buchli, J., Frigerio, M., Focchi, M., and Caldwell, D. G. (2012b). Dynamic torque control of a hydraulic quadruped robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1889–1894. IEEE.

[6] Buchli, J., Kalakrishnan, M., Mistry, M., Pastor, P., and Schaal, S. (2009). Compliant quadruped locomotion over rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 814–820.

[7] Caccavale, F., Natale, C., Siciliano, B., and Villani, L. (1999). Six-DOF impedance control based on angle/axis representations. *IEEE Transactions on Robotics and Automation*, 15:289–300.

[8] Dai, H. and Tedrake, R. (2016). Planning Robust Walking Motion on Uneven Terrain via Convex Optimization. In *IEEE International Conference on Humanoid Robots*.

[9] Dai, H., Valenzuela, A., and Tedrake, R. (2014). Whole-body Motion Planning with Simple Dynamics and Full Kinematics. In *IEEE International Conference on Humanoid Robots*.

[10] de Lasa, M., Mordatch, I., and Hertzmann, A. (2010). Feature-based locomotion controllers. *ACM Transactions on Graphics*, 29(4):1.

[11] Deits, R. and Tedrake, R. (2014). Footstep Planning on Uneven Terrain with Mixed-Integer Convex Optimization. In *IEEE International Conference on Humanoid Robots*.

[12] El Khoury, A., Lamiraux, F., and Taix, M. (2013). Optimal motion planning for humanoid robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 0, pages 3136–3141. IEEE.

[13] Erez, T., Lowrey, K., Tassa, Y., Kumar, V., Kolev, S., and Todorov, E. (2013). An integrated system for real-time Model Predictive Control of humanoid robots. In *IEEE/RAS International Conference on Humanoid Robots*.

[14] Escande, A., Kheddar, A., and Miossec, S. (2006). Planning support contact-points for humanoid robots and experiments on HRP-2. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2974–2979.

[15] Escande, A., Kheddar, A., and Miossec, S. (2013). Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, 61(5):428–442.

[16] Fankhauser, P. (2012). *Optimizing Robotic Single Legged Locomotion with Reinforcement Learning*. PhD thesis, Swiss Federal Institute of Technology Zurich.

[17] Featherstone, R. (2008). *Rigid Body Dynamics Algorithms*. Springer US, Boston, MA.

[18] Focchi, M., Barasuol, V., Havoutis, I., Semini, C., Caldwell, D. G., Barasuol, V., and Buchli, J. (2013). Local Reflex Generation for Obstacle Negotiation in Quadrupedal Locomotion. *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*, pages 1–8.

[19] Focchi, M., del Prete, A., Havoutis, I., Featherstone, R., Caldwell, D. G., and Semini, C. (2017). High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots*, 41(1):259–272.

[20] Fujimoto, Y., Obata, S., and Kawamura, A. (1998). Robust biped walking with active interaction control between foot and ground. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3.

[21] Full, R. and Koditschek, D. (1999). Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of Experimental Biology*, 202(23):3325–3332.

[22] Gabiccini, M., Artoni, A., Pannocchia, G., and Gillis, J. (2015). A Computational Framework for Environment-Aware Robotic Manipulation Planning. In *International Symposium on Robotics Research (ISRR)*.

[23] Gaspero, L. D. (2016). Quadprog++: A C++ library for Quadratic Programming which implements the Goldfarb-Idnani active-set dual method.

[24] Gill, P. E., Murray, W., and Saunders, M. A. (1997). Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Jouranl on Optimization*, 12:979–1006.

[25] Hansen, N. (2014). CMA-ES: A Function Value Free Second Order Optimization Method. In *PGMO COPI 2014*, Paris, France.

[26] Hauser, K. (2008). *Motion Planning for Legged and Humanoid Robots*. PhD thesis, Stanford University.

[27] Hauser, K., Bretl, T., Harada, K., and Latombe, J.-c. (2006). Using motion primitives in probabilistic sample-based planning for humanoid robots. In *In proceedings of the Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

[28] Hauser, K. and Latombe, J. C. (2009). Multi-modal Motion Planning in Non-expansive Spaces. *The International Journal of Robotics Research (IJRR)*, 29(7):897–915.

[29] Hauser, K. and Ng-Thow-Hing, V. (2010). Randomized multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research (IJRR)*, 30(6):678–698.

[30] Hauser, K., Ng-Thow-Hing, V., and Gonzalez-Baños, H. (2011). Multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research (IJRR)*.

[31] Havoutis, I., Mastalli, C., Winkler, A., Focchi, M., Caldwell, D. G., and Semini, C. (under-review). Planning, Perception and Whole-Body Control of Dynamic Quadrupedal Locomotion. *Journal on Autonomous Robots (AURO)*.

[32] Havoutis, I., Semini, C., and Caldwell, D. G. (2014). Virtual model control for quadrupedal trunk stabilization. In *In Dynamic Walking*.

[33] Herdt, A., Diedam, H., Wieber, P.-B., Dimitrov, D., Mombaur, K., and Diehl, M. (2010). Online Walking Motion Generation with Automatic Footstep Placement. *Advanced Robotics*, 24(March 2015):719–737.

[34] Hornung, A., Dornbush, A., Likhachev, M., and Bennewitz, M. (2012). Anytime search-based footstep planning with suboptimality bounds. In *IEEE International Conference on Humanoid Robots (IROS)*.

[35] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*.

[36] Ijspeert, A., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1398–1403. IEEE.

[37] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1620–1626.

[38] Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., and Schaal, S. (2010a). Fast, robust quadruped locomotion over challenging terrain. In *IEEE international conference on Robotics and Automation (ICRA)*.

[39] Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., and Schaal, S. (2010b). Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research (IJRR)*, 30(2):236–258.

[40] Kalakrishnan, M., Buchli, J., Pastor, P., and Schaal, S. (2009). Learning locomotion over rough terrain using terrain templates. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 167–172.

[41] Kober, J., Bagnell, J. A., and Peters, J. (2009). Reinforcement Learning in Robotics: A Survey. *The International Journal of Robotics Research (IJRR)*.

[42] Kober, J. and Peters, J. (2010). Policy search for motor primitives in robotics. *Machine Learning*, 84(1-2):171–203.

[43] Kolter, J. Z., Abbeel, P., and Ng, A. Y. (2008a). Hierarchical apprenticeship learning with application to quadruped locomotion. In *Neural Information Processing Systems (NIPS)*.

[44] Kolter, J. Z., Kim, Y., and Ng, A. Y. (2009). Stereo vision and terrain modeling for quadruped robots. In *IEEE International conference on Robotics and Automation (ICRA)*, pages 3894–3901.

[45] Kolter, J. Z., Rodgers, M. P., and Ng, A. Y. (2008b). A control architecture for quadruped locomotion over rough terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 811–818.

[46] Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., and Tedrake, R. (2015). Optimization-based locomotion planning, estimation, and control design for Atlas. *Autonomous Robots*.

[47] Kuindersma, S., Permenter, F., and Tedrake, R. (2013). An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion. *CoRR*, abs/1311.1839.

[48] Kuindersma, S., Permenter, F., and Tedrake, R. (2014). An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion. In *International Conference on Robotics and Automation (ICRA)*, pages 2589–2594, Hong Kong, China.

[49] Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers.

[50] Levine, S. and Koltun, V. (2014). Learning Complex Neural Network Policies with Trajectory Optimization. In *International Conference on Machine Learning (ICML)*, volume 32.

[51] Likhachev, M., Gordon, G., and Thrun, S. (2004). ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference (NIPS-03)*. MIT Press.

[52] Mastalli, C., Focchi, M., Havoutis, I., Radulescu, A., Calinon, S., Buchli, J., Caldwell, D. G., and Semini, C. (2017). Trajectory and Foothold Optimization using Low-Dimensional Models for Rough Terrain Locomotion. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[53] Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., and Semini, C. (2016). Hierarchical Planning of Dynamic Movements without Scheduled Contact Sequences. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[54] Mastalli, C., Winkler, A., Havoutis, I., Caldwell, D. G., and Semini, C. (2015). On-line and On-board Planning and Perception for Quadrupedal Locomotion. In *IEEE International Conference on Technologies for Practical Robot Applications (TEPRA)*.

[55] Mistry, M., Buchli, J., and Schaal, S. (2010). Inverse dynamics control of floating base systems using orthogonal decomposition. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3406–3412. IEEE.

[56] Mordatch, I., de Lasa, M., and Hertzmann, A. (2010). Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics*, 29(4):1.

[57] Mordatch, I., Lowrey, K., Andrew, G., Popovic, Z., and Todorov, E. (2015). Interactive Control of Diverse Complex Characters with Neural Networks. In *International Conference on Neural Information Processing Systems (NIPS)*, NIPS-15, pages 3132–3140, Cambridge, MA, USA. MIT Press.

[58] Mordatch, I., Popović, Z., and Todorov, E. (2012a). Contact-invariant optimization for hand manipulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 137–144. Eurographics Association.

[59] Mordatch, I. and Todorov, E. (2014). Combining the benefits of function approximation and trajectory optimization. *Robotics: Science and Systems*.

[60] Mordatch, I., Todorov, E., and Popović, Z. (2012b). Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4):1–8.

[61] Mordatch, I., Wang, J. M., Todorov, E., and Koltun, V. (2013). Animating human lower limbs using contact-invariant optimization. *ACM Transactions on Graphics*, 32(6):1–8.

[62] Nakanishi, J., Radulescu, A., and Vijayakumar, S. (2013). Spatio-temporal optimization of multi-phase movements: dealing with contacts and switching dynamics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5100–5107. IEEE.

[63] Neunert, M., Farshidian, F., Winkler, A. W., and Buchli, J. (2016). Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds. *ArXiv preprint arXiv:1607.04537*.

[64] Orin, D. E., Goswami, A., and Lee, S. H. (2013). Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35:161–176.

[65] Ott, C., Roa, M. a., and Hirzinger, G. (2011). Posture and balance control for biped robots based on contact force optimization. In *IEEE International Conference on Humanoid Robots*, pages 26–33.

[66] Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural networks: the official journal of the International Neural Network Society*, 21(4):682–97.

[67] Peters, J., Vijayakumar, S., and Schaal, S. (2005). Natural Actor-Critic. In *European Conference on Machine Learning (ECML)*, pages 280–291, Porto, Potugal.

[68] Pippine, J., Hackett, D., and Watson, A. (2011). An Overview of the Defense Advanced Research Projects Agency's Learning Locomotion Program. *Int. J. Rob. Res.*, 30(2):141–144.

[69] Ponton, B., Herzog, A., Schaal, S., and Righetti, L. (2016). A Convex Model of Momentum Dynamics for Multi-Contact Motion Generation. *arXiv preprint arXiv:1607.08644*.

[70] Popovic, M. B., Goswami, A., and Herr, H. (2005). Ground Reference Points in Legged Locomotion: Definitions, Biological Trajectories and Control Implications. *The International Journal of Robotic Research (IJRR)*, 24:1013–1032.

[71] Posa, M., Cantu, C., and Tedrake, R. (2013). A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research (IJRR)*.

[72] Raghunathan, A. U. and Biegler, L. T. (2005). An Interior Point Method for Mathematical Programs with Complementarity Constraints (MPCCs). *Journal on Optimization*, 15(3):720–750.

[73] Rebula, J. R., Neuhaus, P. D., Bonnlander, B. V., Johnson, M. J., and Pratt, J. E. (2007). A controller for the littledog quadruped walking on rough terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1467–1473.

[74] Rusu, R. B. (2009). *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany.

[75] Semini, C., Barasuol, V., Boaventura, T., Frigerio, M., Focchi, M., Caldwell, D. G., and Buchli, J. (2015). Towards versatile legged robots through active impedance control. *The International Journal of Robotics Research (IJRR)*, 34(7):1003–1020.

[76] Semini, C., Tsagarakis, N. G., Guglielmino, E., Focchi, M., Cannella, F., and Caldwell, D. G. (2011). Design of HyQ – a Hydraulically and Electrically Actuated Quadruped Robot. *Institution of Mechanical Engineers Part I: Journal of Systems and Control Engineering*, 225(6):831–849.

[77] Stewart, D. and Trinkle, J. (2000). An implicit time-stepping scheme for rigid body dynamics with Coulomb friction. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[78] Tassa, Y., Erez, T., and Todorov, E. (2012). Synthesis and stabilization of complex behaviors through online trajectory optimization. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[79] Tassa, Y. and Todorov, E. (2010). Stochastic Complementarity for Local Control of Discontinuous Dynamics. In *Robotics: Science and Systems (RSS)*.

[80] Theodorou, E., Buchli, J., and Schaal, S. (2010). Reinforcement Learning of Motor Skills in High Dimensions: A Path Integral Approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, number 4, pages 2397–2403.

[81] Vernaza, P., Likhachev, M., Bhattacharya, S., Kushleyev, A., and Lee, D. D. (2009). Search-based planning for a legged robot over rough terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[82] Vukobratović, M. and Borovac, B. (2004). Zero-moment point: thirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157–173.

[83] Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

[84] Wensing, P. M. and Orin, D. E. (2013). Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3103–3109. IEEE.

[85] Winkler, A., Havoutis, I., Bazeille, S., Ortiz, J., Focchi, M., Caldwell, D. G., and Semini, C. (2014). Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[86] Winkler, A., Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., and Semini, C. (2015). Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[87] Zucker, M., Bagnell, J. A., Atkeson, C., and Kuffner, J. (2010). An Optimization Approach to Rough Terrain Locomotion. In *IEEE International Conference on Automation and Robotics (ICRA)*.

[88] Zucker, M., Ratliff, N., Dragan, a. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. a., and Srinivasa, S. S. (2013). CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research (IJRR)*, 32(9-10):1164–1193.

[89] Zucker, M., Ratliff, N., Stolle, M., Chestnutt, J., Bagnell, J. A., Atkeson, C. G., and Kuffner, J. (2011). Optimization and learning for rough terrain legged locomotion. *The International Journal of Robotics Research (IJRR)*, 30(2):175–191.