DISS. ETH N° 25135

Optimization-based motion planning for legged robots

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

ALEXANDER W. WINKLER
M. Sc. in ME, Karlsruhe Institute of Technology
born on August 20, 1988
citizen of Germany

accepted on the recommendation of

Prof. Dr. Jonas Buchli (ETH Zurich), examiner
Prof. Dr. Marco Hutter (ETH Zurich), co-examiner
Dr. Nicolas Mansard (LAAS-CNRS), co-examiner

2018

Agile & Dexterous Robotics Lab
Institute for Robotics and Intelligent Systems
ETH Zurich
Switzerland

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

ETH zürich

Abstract

Legged machines have the potential to traverse terrain that wheeled robots cannot. These capabilities are useful in scenarios such as stairs in homes or debris-filled disaster scenes, such as earthquake areas. This thesis develops one of the algorithms necessary to achieve such a task, a *motion-planner*, which translates a high-level task into a desired motion-plan for a legged robot to execute.

A core difficulty in legged locomotion is that a forward body movement cannot be directly generated, but results from contact forces between the end-effectors and the environment. Furthermore, there are various physical restrictions on the contact forces that can be generated. These can make hand-crafting valid motions for all the interdependent quantities such as body motion, end-effector motion and contact forces tedious and even infeasible for complex tasks. Alternatively, Trajectory Optimization can be utilized to generate motions in a more general, automated way. Only a high-level task is specified, while the algorithm determines the motions and forces taking into account the physical restrictions of legged locomotion. This approach is attractive because once the problem has been properly modeled, the algorithm would, in an ideal case, produce motions for *any* high-level task, solving legged locomotion planning on a general level.

This thesis presents approaches to transcribe, using e.g. collocation, the legged locomotion problem into a mathematical optimization problem, then solvable by off-the-shelf software. The three developed motion-planning algorithms generate motion-plans for increasingly complex terrains and tasks. Simulation and experiments on the quadruped robots HyQ and ANYmal verify the physical correctness of the motion-plans.

The final formulation and main contribution of this thesis holistically determines the gait-sequence, step-timings, footholds, contact forces, swing-leg motions and 6-dimensional body motion, given a desired goal state and 2.5D height map of the non-flat terrain. We model the robot as a single rigid body (SRBD) controlled by the contact forces at the end-effectors. Our novel phase-based parameterization of end-effector motion and forces allows optimizing over the discrete gait sequence using only continuous decision variables. The algorithm efficiently generates highly dynamic motion-plans with flight-phases for legged systems, such as monopeds, bipeds, and quadrupeds.

Zusammenfassung

Laufroboter haben das Potenzial Gelände zu überwinden, welches für rollende Roboter nicht zu bewältigen ist. Diese Fähigkeiten sind hilfreich in Szenarien wie beispielsweise Treppen in Häusern oder Katastrophengebieten, gefüllt mit unstrukturiertem Geröll. Diese Dissertation präsentiert einen der notwendigen Algorithmen um solche Aufgaben auszuführen. Der entwickelte *Bewegungsplanner* generiert aus einer Aufgabenbeschreibung einen gewünschten Bewegungsablauf, der von einem Roboter ausgeführt werden kann.

Eine der grundlegenden Schwierigkeiten der Fortbewegung auf Beinen ist, dass eine Vorwärtsbewegung des Körpers nicht direkt, sondern durch die Kontaktkräfte zwischen den Endeffektoren und der Umgebung generiert werden kann. Zudem gibt es grosse Einschränkungen, welche Kontaktkräfte physikalisch erzeugt werden können. Diese machen es mühsam und für komplexere Bewegungen gar unmöglich, physikalisch korrekte Bewegungen für all diese abhängigen Elemente (Körperbewegung, Endeffektorbewegung, Kontaktkräfte) manuell zu generieren. Stattdessen kann Trajektorienoptimierung solche Bewegungspläne auf eine generellere, automatisiertere Art erzeugen. Eine grobe Aufgabenbeschreibung wird vom Algorithmus in Bewegungen und Kräfte umgewandelt, die den physikalischen Gesetzen der Fortbewegung auf Beinen entsprechen. Dieser Ansatz ist attraktiv, da sobald das Problem einmal korrekt modelliert ist, der Algorithmus im Idealfall in der Lage sein sollte, Bewegungen für *beliebige* Aufgabenbeschreibungen zu generieren und damit die Bewegungsplanung für Laufroboter grundlegend zu lösen.

Diese Dissertation präsentiert Ansätze um das Problem der Fortbewegung auf Beinen als ein mathematisches Optimierungsproblem zu formulieren, welches dann von existierender Software gelöst werden kann. Die drei entwickelten Algorithmen generieren Bewegungspläne für Gelände und Aufgabenbeschreibung mit kontinuierlich wachsender Schwierigkeit. Die physikalische Richtigkeit der generierten Bewegungspläne ist durch Simulationen und Experimente auf den vierbeinigen Robotern HyQ und ANYmal verifiziert.

Der finale Algorithmus, und der Hauptbeitrag dieser Thesis, generiert die Schrittfolge, Schrittzeiten, Fusspositionen, Kontaktkräfte, Beinbewegungen und 6-dimensionale Körperbewegung, von einer gewuneschten Zielposition und einer 2.5D Gelände Repräsentation. Unsere neuartige, phasenbasierte Parametrisierung der Endeffektorbewegung und Kontaktkräfte ermöglicht die Optimierung der diskreten Schrittfolge durch nur kontinuierlichen Entscheidungsvariablen. Wir modellieren den Roboter als einen einzelnen, star-

ren Körper, beeinflusst durch die Kontaktkräfte an den Endeffektoren. Der Algorithmus generiert hochdynamische Bewegungspläne mit Flugphasen für eine Vielzahl von Laufmaschinen mit ein, zwei, oder vier Beinen.

Acknowledgments

I would like to thank my supervisor Jonas for giving me the opportunity to pursue a Ph.D. at the Agile and Dexterous Robotics Lab. He created a mindset in the entire lab with a progressive long-term vision that greatly inspired my research. In this environment, I was able to pursue my ideas while receiving his valuable input pointing me in promising directions. I would also like to thank Marco for serving on my Ph.D. committee and giving me the opportunity to work at the Robotics Systems Lab. It's been very exciting getting to know such a great team and robot. Thanks also goes to Nicolas for also serving on my PhD committee and taking the time to attend and review this thesis.

The work in this thesis would not have been possible without the invaluable input from all the great colleagues and friends at ADRL that I've had the pleasure to interact with. I want to thank Farbod, Diego, and Michael for their close collaboration with many fruitful discussions and inspiring ideas. Thanks also goes to Markus, Tim, Thiago, Johannes, Edo, Manu, Kusi, Lukas, Julio, Peng, Martina, Nitish, Simon, and Asutosh for an exciting time at work and afterward. I also want to thank the great team at RSL for creating such a welcoming working environment. Special thanks to Dario, Marko, Ruben, Jan, Lorenz, Péter, and Christian for a productive but also fun time in room H301. I am also grateful to Claudio and Ioannis for giving me the opportunity to discover such an exciting field as legged robots at IIT.

Finally, I want to also thank all my other friends I met along the way, in Reddinge, Eschdringe, Karlruhe, Lafayette, Genova, Zürich and elsewhere. You have each made this journey an unforgettable experience. Last, but not least, I want to thank my amazing family, especially my parents and brother. You have been such a supportive stronghold and I am forever grateful to have you in my life.

Alex

June 26, 2018

Funding This research has been funded through a Swiss National Science Foundation Professorship awarded to Jonas Buchli and the National Centre of Competence in Research Robotics (NCCR Robotics). This work has also been conducted as part of ANYmal Research, a community to advance legged robotics.

Contents

Symbols

$\mathbf{r}$	Center of Mass position
$\boldsymbol{\theta}$	Base orientation
$\boldsymbol{\omega}$	Base angular velocity
$\mathbf{p}_i$	End-effector position
$\mathbf{f}_i$	End-effector force
$\boldsymbol{\lambda}_i$	End-effector load value
c_i	End-effector contact flag
$\boldsymbol{\Delta}\mathbf{T}_i$	End-effector phase duration
$\mathbf{p}_c$	Center of Pressure
$\mathbf{w}$	NLP decision variables
$\mathcal{P}$	Support polygon
$\mathcal{R}$	Range of motion
$\mathcal{F}$	Friction cone
$\mathcal{C}_i$	Set of stance timings
T	Total duration of motion
$\mathbf{n}$	Terrain normal vector
$\mathbf{t}$	Terrain tangent vector
$\mathbf{F}$	System dynamics function
$\mathbf{q}$	Full robot state
$\mathbf{q}_b$	Floating-base pose
$\mathbf{q}_j$	Joint angles
$\boldsymbol{\tau}$	Joint torques
$\mathbf{M}$	Joint space inertia matrix
$\mathbf{h}$	Generalized forces (Centrifugal, Coriolis, Gravity)
$\mathbf{S}$	Joint selection matrix
$\mathbf{A}$	Centroidal Momentum matrix
$\mathbf{I}$	Inertia matrix

Acronyms

CD	Centroidal Dynamics
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CMM	Centroidal Momentum Matrix
CoM	Center of Mass
CoP	Center of Pressure
CP	Capture Point
DDP	Differential Dynamic Programming
DoF	Degrees of Freedom
iLQG	Iterative Linear Quadratic Gaussian Regulator
IMU	Inertial Measurement Unit
IP	Interior Point Method
LCP	Linear Complementary Problem
LIPM	Linear Inverted Pendulum Model
MICP	Mixed Integer Convex Programming
MIP	Mixed Integer Programming
MPC	Model Predictive Control
NEE	Newton-Euler Equations
NLP	Nonlinear Programming Problem
OC	Optimal Control
ODE	Ordinary Differential Equation
QP	Quadratic Program
RBD	Rigid Body Dynamics
SLQ	Sequential Linear Quadratic Programming
SQP	Sequential Quadratic Programming
SRBD	Single Rigid Body Dynamics
TO	Trajectory Optimization
ZMP	Zero Moment Point

Preface

This doctoral thesis is written as a cumulative dissertation, which means it is based on three articles published in international, peer-reviewed journals (RA-L) and conferences (ICRA). The corresponding chapters 3, 4, 5 are therefore the main contributions of this thesis and copies of these articles, in which mainly the notation across articles has been unified. Chapter 2 extracts the specific contributions of each paper and compares them to each other as well as to the state-of-the-art and chapter 1 gives a general introduction to legged locomotion.

1

Introduction

Robots have started to become commonplace in today's society. We are starting to get accustomed to robots vacuuming our floors, or drones recording us from the sky. We are beginning to give cars more and more autonomy and letting go of the steering wheel on the high way. Fully autonomous cars are just around the corner; steering wheels replaced by sensors, algorithms and actuators.

Despite having seen great technological improvement for flying and rolling robots, one mode of transport has not yet made the jump to mainstream: legged locomotion. Although actively pursued in research, legged robot performance has not yet come close to their biological counterparts. This discrepancy is unfortunate since legged locomotion has a variety of advantages over flying or rolling.

One significant advantage is that no continuous path between start and goal must exist, as is necessary for rolling machines. Discrete positions to place the feet suffice to move around. This capability enables legged systems, like humans, to climb stairs and ladders, traverse debris in earthquake areas or work underground in highly unstructured terrain. Furthermore, legged machines use the environment to move around, which usually enables larger payloads with less effort compared to flying machines.

These capabilities of legged machines have the potential to benefit society. Humans are living longer, and with that, the demand for caretakers is rising. Walking robot nurses would have the ability to move freely in a house with stairs and steps and interact with the elderly. In dangerous situations such as burning buildings, earthquake-struck houses or nuclear disasters, legged robots instead of humans, could be sent to check for and rescue survivors or perform other tasks that require physical interaction. Furthermore, machines that walk can also be used in symbiosis with humans, e.g., help paralyzed patients regain mobility. These exoskeletons could replace a restrictive wheelchair and increase the quality of life for the paralyzed.

Figure 1.1: *Individual capabilities to generate motions for autonomous systems. This thesis introduces an algorithm to plan physically feasible motions for legged robots. These can subsequently be executed using a tracking controller or embedded into an MPC formulation [1]. Images by [2].*

1.1 The bigger picture

Now that the benefits of legged locomotion have been motivated, it is reasonable to ask what is missing on the technological side to create systems that can move using legs. Such a complex task requires various capabilities shown in Fig. 1.1 that must interact to generate, e.g., a walking motion. The following describes the responsibilities of the three components *Sense*, *Act* and, as the focus of this thesis, *Plan*.

Sense The first task of an "autonomous agent" is to perceive its environment. To retain balance while walking, the robot relies on an IMU and state estimation to determine its global position and how its body is oriented. It needs to know what terrain obstacles lie in front of it that should be avoided. Streams coming from cameras or laser scanners must be analyzed to build an accurate map of the environment.

Act After the environment and the own state is perceived, a motion is planned (explained in next paragraph) that is subsequently executed on the robot. This execution ("acting") includes applying torque to the joints to generate a movement of the system. This requires high-quality hardware and software to work reliably. The software calculates the correct motor current (or hydraulic valve opening) to create, or "track", a desired torque. We also place *motion tracking controllers* into this category, which are responsible for generating appropriate joint torques to follow a given motion plan. These must continuously observe the current state of the robot and adapt the joint torques accordingly. This problem of generating torques for floating-base systems to achieve a desired motion or acceleration has been widely studied, and many useful solutions exist. Most approaches are based on using the Rigid Body Dynamics (RBD) while enforcing additional constraints [3] or directly tracking the planned end-effector forces [4].

(Re-)plan The link between the sensing of the environment and the execution of a motion is called *motion-planning*. If the entire loop in Fig. 1.1 is repeated sufficiently

fast and the plan is produced by the optimization of the robot dynamics, the method is termed Model Predictive Control (MPC). This frequent re-optimization is attractive, as without any additional methods the system gains a variety of reactive capabilities. These capabilities, including recovery from slipping and external pushes or re-planning if the motion is not executing as predicted, are crucial to deploy robots in real-world environments.

However, a critical building block included in all MPC structures is the motion planning performed in every loop of Fig. 1.1. If the motion-plan is already physically infeasible, it is impossible for even the best controllers or an MPC structure to execute it. As we will explain next, the range of possible motions for legged systems is strongly restricted. Therefore, coming up with a physically feasible motion-plan that respects all these restrictions is far from trivial. The main result and focus of this thesis is such a *motion-planning algorithm for legged robots*. The equally important tasks of robustly tracking these motions or designing high-performance hardware to enable this are outside the scope of this thesis.

The rest of this chapter recapitulates essential theory related to legged locomotion. We introduce different ways to model the dynamics of a legged machine, highlight distinctive physical constraints present when walking on legs, briefly explain how locomotion planning is traditionally solved and why Trajectory Optimization (TO) might be a useful tool for this task.

1.2 Dynamic models for legged systems

A variety of mathematical models can be used to design efficient algorithms for legged locomotion planning. A motion plan is considered physically correct according to a given model, if the modeling equations are fulfilled at all times. These models quantify the relationship between an input $\mathbf{u}$ to the current state $\mathbf{x}$ of the system, and the resulting change of state $\dot{\mathbf{x}}$. This can be modeled by the Ordinary Differential Equation (ODE)

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

The following introduces different dynamic models to represent legged robots and highlights which assumptions are necessary for their physical correctness. These representations vary by how many dimensions the state $\mathbf{x}$ represents, what physical quantity is considered as the input $\mathbf{u}$ and how this input affects the motion defined by $\mathbf{F}$. Models are merely approximations of the actual physics and vary by how many assumptions they require. In the following, the models with the least assumptions are introduced first and assumptions are continuously added to derive simplified dynamic representations.

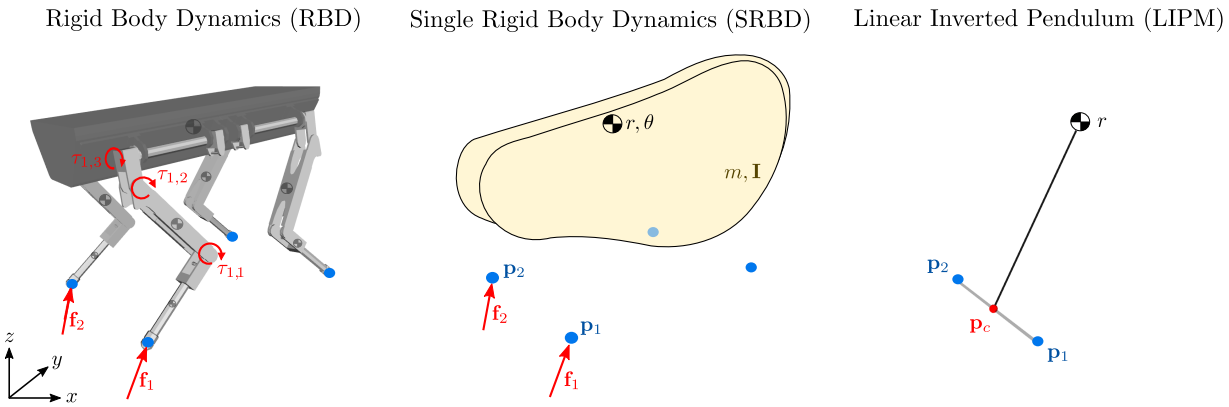


Figure 1.2: *Visual representations of the dynamic modeling equations that can be used to approximate the physics of the quadruped robot HyQ[5] pushing off its hind legs.*

1.2.1 Rigid Body Dynamics

One possibility to model a system is through each of its rigid bodies. *Rigid* in this context implies that

Assumption A1 (RBD)**.** *Bodies do not deform when forces are applied.*

With this assumption, the system in Fig. 1.2 can be fully described through the generalized coordinates $\mathbf{q} = [\mathbf{q}_b^T \ \mathbf{q}_j^T]^T \in \mathbb{SE}(3) \times R^n$ [1], which holds the position and orientation of the base, as well as the angle of each joint. By viewing a robot as a collection of rigid bodies connected through actuated joints we can derive a relationship between the torque $\boldsymbol{\tau} \in \mathbb{R}^n$ acting at each joint and the corresponding motion as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}(\mathbf{q})^T\mathbf{f}, \tag{1.2}$$

where $\mathbf{M} \in \mathbb{R}^{(6+n)\times(6+n)}$ is the joint-space inertia matrix, $\mathbf{h} \in \mathbb{R}^{6+n}$ is the effect of Centrifugal, Coriolis and gravity terms, $\mathbf{S}^T = [\mathbf{0}_{n\times 6} \ \ \mathbf{I}_{n\times n}]^T$ is a Selection Matrix which applies the torque $\boldsymbol{\tau}$ to only the n joint rows and the Jacobian $\mathbf{J}$ maps forces $\mathbf{f} = [\mathbf{f}_1^T, \ldots, \mathbf{f}_{n_i}^T]^T \in \mathbb{R}^{3n_i}$ at the n_i end-effectors to $6 + n$ dimensional generalized forces.

We can split (1.2) into 6 unactuated (u) and n actuated (a) rows [6]. The unactuated rows correspond to the equations for the base, and the other rows describe the joint motions.

$$\mathbf{M}_u(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}_u(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{J}_u(\mathbf{q})^T\mathbf{f} \tag{1.3a}$$
$$\mathbf{M}_a(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}_a(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau} + \mathbf{J}_a(\mathbf{q})^T\mathbf{f}, \tag{1.3b}$$

The unactuated subsystem (1.3a) mathematically shows that a torque $\boldsymbol{\tau}$ cannot directly influence the base motion of, e.g., a humanoid. Differently stated, there exists no motor

[1] R^n describes a vector of real numbers $\mathbb{R}$, but not necessarily a vector space.

for this virtual six Degrees of Freedom (DoF) torso joint. The feet have to push off the ground in the opposite direction to achieve forward motion. This contact force can be generated through the motors in the leg joints, however, they only indirectly actuate the torso, or *base*. The base is considered *unactuated* and we talk about a *floating-base* system. This unactuated subsystem can be used to verify that the relation between external forces $\mathbf{f}$ and the system's accelerations $\ddot{\mathbf{q}}$ are physically correct.

Once these two quantities are determined, ensuring the relationship defined by the actuated subsystem (1.3b) is trivial. Since there exists a torque-generating motor for each robot joint, this subsystem is fully determined. Just as for a fixed-based manipulator, it is always possible to choose appropriate torques $\boldsymbol{\tau}$ to fulfill these actuated equations.

If torque limits are large enough, this can justify disregarding the actuated subsystem during motion-planning. By doing so, we can eliminate n equations as well as the dependency on the joint torques in the planning problem, without sacrificing generality or accuracy. We can shift the emphasis to the six rows representing the unactuated dynamics, as done in [7]–[9]. These crucial dynamics for legged locomotion can also be described as shown in the following.

1.2.2 Centroidal Dynamics

By expressing the change of momentum in a frame anchored at the current Center of Mass (CoM) $\mathbf{r}$, also called *centroid*, we can rewrite the dynamics expressed in (1.3a) as

$$\mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{A}}(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} m\mathbf{g} + \sum_{i=1}^{n_i} \mathbf{f}_i \\ \sum_{i=1}^{n_i} \mathbf{f}_i \times (\mathbf{r}(\mathbf{q}) - \mathbf{p}_i(\mathbf{q})) \end{bmatrix}. \tag{1.4}$$

This formulation is called Centroidal Dynamics (CD) [10]. The Centroidal Momentum Matrix (CMM) $\mathbf{A} \in \mathbb{R}^{6\times(6+n)}$ maps the velocities, and therefore momentum, of each individual body into a common reference frame, expressed at the CoM. The gravity acceleration is given by $\mathbf{g}$, the mass of the entire robot by m, the position of each end-effector i is denoted as $\mathbf{p}_i$ and the force acting there is given by $\mathbf{f}_i$. The left-hand side denotes the change of momentum $\frac{\mathrm{d}(\mathbf{A}\dot{\mathbf{q}})}{\mathrm{d}t}$, which must be equal to the sum of external forces and moments on the right-hand side.

These six equations relate the change of linear and angular momentum of all rigid bodies projected into the CoM with the external forces. This system of equations is underdetermined with respect to the accelerations $\ddot{\mathbf{q}}$, just as (1.3a). Given an input force $\mathbf{f}$, there exist multiple physically feasible ways the system can evolve. This ambiguity makes the equation more suitable for verification whether a specific state is physically correct than for forward simulation. Also, (1.4) depends on the joint state, which contributes to the nonlinearity of the system. The following introduces a way to eliminate the joint dependency through additional assumptions.

1.2.3 Single Rigid Body Dynamics

To remove the dependency on the joint angles in (1.4), while keeping the dynamics approximately correct, the following assumptions additional to A1 are necessary:

Assumption A2 (NEE)**.** *Momentum produced by the joint velocities is negligible.*

Assumption A3 (NEE)**.** *Full-body inertia remains similar to the one in nominal joint position.*

This might be reasonably assumed for robots in which the base makes the largest contribution to the total mass of the robot, e.g. HyQ [5], ANYmal [11], MIT Cheetah [12], Spot-Mini [13] and Cassie [14]. From their negligible limb masses follows that (i) even fast limb motion does not contribute significantly to the momentum of the system and (ii) their position does not influence the full-body inertia. Another case that might justify these assumptions is a robot that has significant limb masses and inertia, but these move very slow and stay close to their nominal configurations. This could, for instance, be a humanoid robot such as HRP-2 [15], when its slow steps do not significantly influence the momentum and its short steps with the resulting similar joint configurations also barely change the full-body inertia.

With these assumptions (1.4) can be simplified as shown in Appendix A.1. By separating the linear and angular parts we get the Newton-Euler Equations of a single rigid body, the Single Rigid Body Dynamics (SRBD)

$$m\ddot{\mathbf{r}} = m\mathbf{g} + \sum_{i=1}^{n_i} \mathbf{f}_i \tag{1.5a}$$

$$\mathbf{I}(\boldsymbol{\theta})\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}(\boldsymbol{\theta})\boldsymbol{\omega} = \sum_{i=1}^{n_i} \mathbf{f}_i \times (\mathbf{r} - \mathbf{p}_i), \tag{1.5b}$$

where $\mathbf{r}$ is the CoM, $\boldsymbol{\theta}$ is the base orientation and $\boldsymbol{\omega}$ the angular velocity of this combined rigid body (see Fig. 1.2). The combined mass all the limbs and base is given by m and $\mathbf{I}(\boldsymbol{\theta}) \in \mathbb{R}^{3\times3}$ is the full-body inertia of all individual limbs in nominal joint configuration combined, with respect to a frame anchored at the CoM and whose axis are parallel to the inertial frame (see Appendix A.1).

Through the additional assumptions, we sacrifice some accuracy of the dynamic representation, however, we gain full independence of the joint angles. This allows us to express the dynamics in (1.5) purely through Cartesian coordinates and rotations in 3-dimensional space. This eliminates the nonlinearity introduced by the joint angles. Additionally, in legged locomotion, it is often less critical how exactly the joints of a leg are oriented, as long as the foot is at a specified position. If we neglect the joint motions' effects on the dynamics, we can completely decouple robot-dependent joint kinematics from the general physical legged locomotion problem. This allows a simpler formulation of the physical laws and constraints of legged locomotion and has been used in [16], [17].

Assumption A2 and A3 might seem overly strong, yet approaches using the Linear Inverted Pendulum Model (LIPM) are also making these assumptions (and more) and have shown to be sufficiently accurate for a variety of use-cases [18]–[21]. Therefore, this model lies somewhere in-between the CD and the LIPM regarding complexity as well as accuracy. It retains the 6-dimensional base motion as well as the individual contact forces. However, the cross products still introduce nonlinearity, which cannot be handled in, e.g., Quadratic Program (QP) constraints. This nonlinearity is eliminated with further assumptions as seen in the following.

1.2.4 Linear Inverted Pendulum Model

The starting point for the LIPM in Fig. 1.2 are the SRBD. In order to remove the nonlinearity introduced by the cross product, the following assumptions additional to A1, A2, A3 are necessary:

Assumption A4 (LIPM)**.** *CoM height r_z is constant.*

Assumption A5 (LIPM)**.** *Angular velocity $\boldsymbol{\omega}$ and acceleration $\dot{\boldsymbol{\omega}}$ of the base are zero.*

Assumption A6 (LIPM)**.** *Footholds are at constant height p_z.*

With these assumptions Appendix A.2 simplifies equation (1.5a) to the form

$$m\ddot{r}_x = \sum_{i=1}^{n_i} f_{i,x} \overset{A.2}{=} \frac{mg}{r_z - p_z}\left(r_x - \frac{\sum_{i=1}^{n_i} f_{i,z} p_{i,x}}{\sum_{i=1}^{n_i} f_{i,z}}\right) = \frac{mg}{h}(r_x - p_{c,x}), \tag{1.6}$$

where x indexes the horizontal motion (y accordingly), h is the walking height of the robot and $p_{c,x}$ the x-position of the Center of Pressure (CoP), which is manipulated through the vertical contact forces $f_{i,z}$ and the position $p_{i,x}$ of the feet. This model describes how the position of the CoP linearly affects the horizontal CoM acceleration. It can be solved analytically and efficiently and has therefore been used in a variety of approaches [20], [22]–[24].

The disadvantages of this model are the restrictive assumptions A4, A5, A6 made for the model to be valid. For more complex motions it could be essential to reorient the body to reach specific footholds, accelerate vertically to jump, or place feet at different heights to cross uneven terrain. Furthermore, the input to this system is the CoP created by a combination of the vertical components of each individual contact force. This abstraction loses critical information about the vertical force at each foot, as well as ignores their tangential components. The 3-dimensional forces at each foot, however, are the crucial elements that move the floating base and are subject to a variety of constraints (e.g., friction cone). By summarizing them into the CoP, relevant information is lost which can make it difficult to model many of the characteristics of legged locomotion.

Some extensions to mitigate some of these restrictions imposed by the LIPM have been developed, e.g., [25]. Another option if the LIPM is too restrictive for a specific use-case

(and nonlinearity of the formulation can be tolerated) is to directly use more general formulations such as the full RBD (1.2) or the SRBD (1.5). These are still only approximate models, but rely on fewer assumptions and sometimes make it easier to recognize the underlying physical principles.

A summary of the different dynamic models described previously and what characterizes their state and input can be seen in Table 1.1.

Table 1.1: *Dynamic models for legged robots and their relation to (1.1). Only the position components are listed, although* $\mathbf{x}$ *includes both position and velocity.*

	$\mathbf{F}$	State $\mathbf{x}$	Input $\mathbf{u}$	Assumptions
Rigid Body Dynamics (RBD)	(1.2)	$\mathbf{q}_b, \mathbf{q}_j$	$\boldsymbol{\tau}$, $\mathbf{f}_i$	A1
Centroidal Dynamics (CD)	(1.4)	$\mathbf{q}_b, \mathbf{q}_j$	$\mathbf{f}_i$	A1
Single Rigid Body Dynamics (SRBD)	(1.5)	$\mathbf{r}, \boldsymbol{\theta}, \mathbf{p}_i$	$\mathbf{f}_i$	A1,A2,A3
Linear Inverted Pendulum Model (LIPM)	(1.6)	r_x, r_y	$\mathbf{p}_c$	A1,A2,A3,A4–A6

1.3 Physics of legged locomotion

Since the reaction forces with the environment are the main way to move a floating-base system, we will now have a closer look at these forces with the goal of more precisely defining the legged locomotion problem.

A commonly seen floating-base system is a wheeled vehicle as seen in Fig. 1.3. The system cannot be directly accelerated forward, but the torque applied to the wheels creates a reaction force between the tires and the asphalt, which moves the car forward. The vertical forces always balance the weight of the car, so planning or control in this direction is not required. Steering is achieved by creating lateral forces at the front wheel by turning them.

The drone in Fig. 1.3 moves due to the thrust forces created by each rotor. These thrust forces can be controlled through the rotor velocities to move the 6-dimensional base in a desired way. This is not trivial since the forces can only act in normal direction, pushing up- or downwards depending on the rotor's rotational direction. Fortunately, forces can still be created at all times, independent of where the system is in the environment.

What complicates the control of legged systems compared to rolling cars or flying drones, are the various restrictions on the achievable reaction forces. These force restrictions, which consequently also limit the possible base motions, are:

- Forces can only be created when a foot is touching the environment. This restricts *where* the forces can act to the surface of the terrain and nowhere else. This is in contrast to drones, which can always exert a force at the center of their rotors by spinning them, independent of the terrain.

Figure 1.3: *Floating-base systems for which the base cannot be directly actuated. Algorithms for the control of self-driving cars [26] and drones [27] are quite mature, whereas those for legged robots, such as quadrupeds [28] or bipeds [29], are still an open research question. The difficulty to control legged systems comes largely from the various restrictions for the creation of external forces.*

- For locomotion without slipping, the locations of the force is unmovable. To generate a force at a different location, we must accept a duration during which no force is possible to re-position the foot. While taking this step, the leg typically does not significantly contribute to moving the base. This abrupt loss of control of such a *switching system* is a substantial restriction that does not affect drones or cars.
- Physically it is only possible to create forces that push into the ground, and not pull on it. This is why our heels cannot pull us back upright once we leaned too far past the tip of our toes. The *unilateral* forces make many base accelerations physically impossible.
- Tangential forces, the driving forces that move the body forward, must remain inside the friction cone. We can only create larger tangential forces by simultaneously increasing the normal force (Coulomb friction). This however also affects the body acceleration in normal direction.

These strong restrictions and their effect on the base motion is what makes legged locomotion so difficult from a planning and control point of view.

1.4 Traditional legged locomotion planning

Previously, we introduced the dynamic models and physical restrictions that characterize legged locomotion. In the following, we will briefly explain existing approaches to control legged robots and discuss which aspects could be improved.

To traverse any terrain, it is helpful to know where to step. Since foothold selection is a difficult problem, a variety of heuristics is often used for simplification: (i) Footholds are often categorized as good if they are on flat ground so slipping can be avoided, (ii) they do not have any high obstacles nearby that the foot can collide with and (iii) they guide

the robot to a desired goal. This selection of appropriate footholds and step timings given a height-map of the terrain is usually the first stage for traditional legged locomotion planning. Expressed in terms of the previously introduced restrictions, this predefines *where* (footholds) and *when* (step timings) external forces can act.

Only after these force locations have been predetermined a body motion is found that can be created with the available forces. A common approach [22] is to model the robot as a LIPM, and find a motion for which the CoP is always inside the area of the feet in contact (*support area*). This is an alternative way to enforce that only pushing forces are required to generate this body motion.

Once footholds and a base motion-plan have been found, the goal is to apply this to a physical system. This requires the generation of the appropriate joint torques to track the reference motions. This is often done by solving the RBD model (1.2) for those joint torques that track the planned accelerations (Inverse Dynamics), subject to additional constraints [3].

Limitations

This partitioning of the problem into solvable subtask is sensible, but also has its limitations. For instance, the separation between foothold and body motion planning can limit performance. The footholds specify *where* the contact forces are allowed to be applied. Afterwards a physically correct body motion must be determined using these possibly suboptimal contact locations. However, for complex motions, it can be challenging to select these locations without taking into account the dynamics of how the generated forces affect the base. This close coupling hints towards finding a solution for the feet and the base motion *simultaneously* to generate more complex motions.

Furthermore, more complex motions and terrains might require the base to reorient itself, jump vertically and place feet at different heights. This violates the assumptions underlying the LIPM (1.6), possibly necessitating a different representation of the dynamics. Since many of these LIPM restrictions come from the abstraction of the contact forces into the CoP, a dynamic representation that directly takes forces as input might be more appropriate (Table 1.1). The constraints on precisely these contact forces are after all an integral part of legged locomotion as explained in Section 1.3. Direct modeling of forces requires a higher-dimensional model, as well as a way to represent the complex interplay between floating-base accelerations, external forces and various constraints on these. Since hand-designing all these different quantities in a coherent way becomes increasingly difficult, we will next have a look at mathematical optimization to outsource this complexity.

1.5 Trajectory optimization

Generating motions for dynamic systems subject to a variety of physical constraints has been studied in the field of Trajectory Optimization (TO). Our goal is to reduce the amount

of hand-crafted components by leveraging these numerical optimization techniques. There exist different methods to formulate such a TO problem, namely Dynamic Programming using the Bellman Optimality Equation, indirect using the Maximum Principle and direct methods. For a complete overview as well as detailed information on TO methods see [30]–[32].

We will focus on *direct methods*, in which the continuous time TO problem is transcribed to a Nonlinear Programming Problem (NLP) affected by a finite number of decision variables as

$$\min_{\mathbf{w}} a(\mathbf{w}) \quad \text{subject to} \quad \mathbf{b}(\mathbf{w}) = \mathbf{0}, \quad \mathbf{c}(\mathbf{w}) \geq \mathbf{0}. \tag{1.7}$$

A solver attempts to find the variables $\mathbf{w}$ that minimize the cost a, while fulfilling the equality $\mathbf{b} = \mathbf{0}$ and inequality $\mathbf{c} \geq \mathbf{0}$ constraints. After formulating the problem in this way, it can be solved with a variety of methods for mathematical optimization, using eg. an Interior Point Method [33] or Sequential Quadratic Programming (SQP) [34] implemented by off-the-shelf solvers such as IPOPT [35] or SNOPT [36].

The appeal of TO is that once the physics have been appropriately modeled, the algorithm can produce motions for a wide variety of tasks, solving legged locomotion planning on a more general level. Therefore, capturing the physically relevant quantities of legged locomotion through the quantities $\mathbf{w}, a, \mathbf{b}, \mathbf{c}$ is a more precise formulation of the goal of this thesis.

A main objective for all these approaches is to satisfy the dynamics ODE (1.1). The following explains two common ways to transcribe a continuous TO problem into the discrete form in (1.7). These methods are independent of the problem of legged locomotion and applicable to a variety of dynamical systems.

1.5.1 Sequential methods

An intuitive way to transcribe the continuous TO problem into a finite dimensional NLP is *single shooting*. Only the continuous input $\mathbf{u}(t)$ is parameterized through a set of discrete variables, e.g., polynomial coefficients $\mathbf{w}$. This input is applied to the dynamic system (1.1) and the states $\mathbf{x}(t)$ are forward simulated using an ODE solver. The NLP solver verifies if the resulting states fulfill the specified equality ($\mathbf{b}$) and inequality ($\mathbf{c}$) constraints, such as e.g. reach a goal or avoid high velocities. If this is not the case, the parameters $\mathbf{w}$ describing the input trajectory are adapted based on gradient information and the dynamic system is simulated (forward integrated) again. This method continuously "shoots" the dynamics forward until input parameters are found for which the states evolve according to the cost and constraints.

The advantages of this method are that it can make use of adaptive, error-controlled ODE solvers. These can change their integration step-size depending on the current dynamics, thereby avoiding unnecessary calculation and reducing computation time. Also, the problem stays relatively small, since only the control trajectory is optimized. The disadvantages are that the state trajectory cannot be initialized directly, as only the often less

intuitive input variables are available. Furthermore, unstable systems can be difficult to handle, since small modifications of the input can have large effects on the state $\mathbf{x}$ later in the trajectory [30], [32].

The duration of the shooting intervals can be reduced to avoid the trajectory becoming unstable. One approach to accomplish this is *multiple shooting*, which strings together multiple shots from optimized starting values. To produce a continuous final trajectory, the NLP enforces that each shot ultimately ends up at the starting state of the next. This formulation increases the sparsity of the problem, as well as convergence and stability since inputs in one shot do not directly influence the trajectory of the next shot.

There also exists a different way to parameterize the problem, which does not require the use of an ODE solver altogether and will be presented in the following.

1.5.2 Simultaneous methods

If the decision variables $\mathbf{w}$ parameterize the input $\mathbf{u}(t)$ *and* state $\mathbf{x}(t)$ over time, the method is classified as a *simultaneous* method. In this case, the system dynamics are not enforced by forward integration, but through equality constraints in $\mathbf{b}(\mathbf{w})$. The solver varies the state $\mathbf{x}(t)$ and input $\mathbf{u}(t)$ trajectory simultaneously while trying to enforce the system dynamics relationship (1.1) between them at specified times.

This requires pre-specifying the times at which the dynamic constraint is enforced, and is computationally expensive to be changed during the iterations if a different accuracy is required (as opposed to adaptive step-size ODE solvers). On the positive side, since the state trajectory is part of the decision variables, an initial motion can be easily set. Also, due to the decoupling between state and input, changes in the input only affect the state at that time instance, while future states remain unaffected. This characteristic, which is reflected in the sparse structure of the Jacobian, increases the robustness of the solver towards unstable systems [30], [32], [37].

Due to these advantages, all the TO approaches presented in this thesis use variants of the simultaneous method to formulate problem (1.7). They vary in which DoF of the system are optimized, what dynamic model is used and how the legged locomotion constraints are formulated.

2

Contributions

The main result of this thesis is a motion-planning algorithm for legged robots. This thesis is based on three published papers [17], [38], [39], each with their dedicated chapter (3, 4, 5). This chapter starts by motivating and summarizing each paper, then defines a set of metrics to compare motion-planning algorithms, subsequently presents a variety of state-of-the-art approaches, and concludes by summarizing the contributions of this thesis.

2.1 Relevant publications

The following lists the published papers and summarizes the motivation and contributions of each. It highlights which capabilities might be desirable for an algorithm for legged locomotion planning, and how these were gradually added. Each specific contributions is denoted by (Cx).

Paper I

Alexander W. Winkler, Farbod Farshidian, Michael Neunert, Diego Pardo, Jonas Buchli. *Online Walking Motion and Foothold Optimization for Quadruped Locomotion.* In IEEE International Conference on Robotics and Automation (ICRA), pp. 5308-5313, 2017.

Motivation

Traditional motion planning for legged robots separates the foothold planning from the body-motion planning [18]. This keeps each sub-problem tractable, but also necessitates heuristics to link the two. For instance, when planning footsteps to reach a goal state, a maximum step-length is often set. A more accurate formulation would be to limit the distance between the base and each foot. However, since the footholds are traditionally planned *before* the position of the base is known, this heuristic maximum step length must be used. We attempt to minimize the use of heuristics such as maximum or nominal step length since these are often approximations of actual physical laws and become increasingly difficult to determine for more complex motions. Typically, since these values are not precisely known, they are set more on the conservative side. The sum of all these conservative approximations then limits the achievable motions.

Contribution

To reduce heuristics, we optimize over the footholds and the body motion simultaneously. This makes the constraint that keeps the CoP inside the area of the stance feet nonlinear and requires an NLP, as opposed to a traditional QP, to solve the problem. This additional complexity however also allows us to create arbitrarily oriented support areas exactly where they are needed for the CoM to stably reach a commanded goal (C1). Additionally, we show that despite the nonlinearity of the problem, the solution time remains in the order of milliseconds.

Paper II

Alexander W. Winkler, Farbod Farshidian, Diego Pardo, Michael Neunert, Jonas Buchli. *Fast Trajectory Optimization for Legged Robots using Vertex-based ZMP Constraints.* In Robotics and Automation Letters (RA-L), pp. 2201–2208, 2017.

Motivation

In the previous paper, we demonstrate that it is possible to find the footholds and body motion simultaneously, thereby reducing heuristics compared to a decoupled footstep planner. One limitation of the approach is that it can only generate motions for quadruped robots with three or more legs on the ground. However, in many biological gaits, there are often only one or two legs in contact at a given time. Since our previous formulation requires support *areas*, these point- and line-contacts cannot be modeled. Typically such more dynamic motions are generated using Capture Point [40] approaches – determining where to step to upright a LIPM. If however a slower, statically stable gait is desired,

the approach is switched to one using the Zero Moment Point (ZMP) with support areas. This "either-or" prohibits the mixing of motions, where the gait cannot be precisely categorized, e.g., limping, trot with a slight overlap of stance phases. However, specifically these non-traditional gaits might be required for more complex environments in which legged robots can demonstrate their advantages over wheeled systems.

Contribution

Working towards a unified motion-planning algorithm, this paper introduces a TO formulation where the CoP is viewed as input and the support areas as input bounds, which can be modified through the optimized foothold locations (C2). This formulation generalizes the traditional ZMP planning, so the Capture Point (CP) is the solution of our TO problem for a specific case (CoP on point-foot). In our formulation, the CP and ZMP approaches differ through the shape of the support areas. To represent the line-contacts present in, e.g., quadruped trotting, we introduce a vertex-based representation of the support areas (C3). This formulation allows treating arbitrarily oriented point-, line- and area-contacts uniformly. It also eliminates the need to order contact points in a specific way before calculating the support edges. Through this general TO formulation, as well as the vertex-based ZMP-constraint representation, we can efficiently generate motions such as quadrupedal walking, trotting, bounding, pacing, combinations and transitions between these, limping, bipedal walking and push-recovery with a single algorithm.

Paper III

Alexander W. Winkler, C. Dario Bellicoso, Marco Hutter, Jonas Buchli. *Gait and Trajectory Optimization for Legged Systems through Phase-based Endeffector Parameterization.* In Robotics and Automation Letters (RA-L), pp. 1560-1567, 2018.

Motivation

The above approaches combine and optimize over many of the relevant quantities of legged locomotion. However, they are both based on the LIPM model, controlled by the position of the CoP. This helpful simplification of the dynamics reaches its limits as motions and terrains become more complex. Some motions might require reorienting the body to reach specific footholds, accelerate vertically to jump, or place feet at different heights when crossing uneven terrain, which however violates the assumptions of the LIPM. Other limitations come from abstracting the individual 3D contact forces into CoP. Some of the characteristic constraints in legged locomotion (Section 1.3) can only be modeled if the contact forces are explicitly represented in the optimization problem. Finally, our previous papers fix the order and duration of each step. Especially for more complex, non-flat

terrain, a suitable order can be difficult to determine beforehand. Therefore, it would be useful if the motion-planning algorithm could find the optimal step sequence and durations autonomously.

Contribution

In this paper we replace the LIPM model with the 6D SRBD (1.5) and the CoP with the 3D contact forces of each end-effector. This model allows reformulating concepts such as support-area constraints into unilateral contact forces, which can be more directly connected to underlying physical laws. Since the forces are part of the optimization problem, friction constraints can also be directly formulated. Kinematic limits we previously expressed as 2D surface patches are now constrained by 3D cubes. Since the CoP is not defined for zero forces, the LIPM could not be used to generate motions with flight-phases. However, the SRBD handle the contact forces directly and therefore generate valid accelerations also for flight-phases and other vertical motions (C4). To allow the algorithm to choose the optimal gait and step durations, we introduce a novel phase-based foot parameterization (C5). This avoids Mixed Integer Programming (MIP) and Linear Complementary Problem (LCP) constraints, keeping the optimization variables continuous and allowing the NLP solver to simultaneously optimize the gait and motion.

2.2 Capability metrics of motion-planning algorithms

Significant work has been done in the field of motion planning. Before comparing approaches, we define metrics by which to evaluate the abilities of motion planning algorithms. The following lists five criterion that in combination could be used to compare the capabilities of various motion-planning algorithms. The individual importance of each criterion might vary depending on the application, however, they all influence the evaluation.

Which components are optimized?

By solving different aspects of the legged locomotion problem separately, constraints between them cannot be directly formulated. The conservative heuristics can limit the range of achievable motions. Therefore, defining as little as possible a-priori, giving the algorithm the most freedom in how to complete a specified task, is often favorable.

The components we view as critical to legged locomotion are summarized in Table 2.1. Firstly, it is helpful that the entire 6-dimensional base motion is optimized, allowing tilting the body to reach specific footholds or jumping vertically to cross a gap. Furthermore, we emphasized the tight connection between footholds and base motion in Section 1.4, which suggests it is beneficial to optimize them simultaneously. By concurrently also generating the motion of the swing-leg, the range of motion constraints and obstacle avoidance can

be directly taken into account. Finally, an algorithm that also determines the number of steps, their order (gait) and the duration of each step to best achieve a given task might allow a broader range of motions to be generated.

How difficult are the shown tasks and terrain?

The quality of a motion-planning algorithm can also be evaluated based on the difficulty of the terrain or task for which it finds a solution. Unfortunately, there exists no standardized scenarios and terrains which all algorithms attempted to solve. Therefore, we make the assumption that the motions shown in the accompanying videos are the most complex ones the algorithm is capable of producing with reasonable tuning effort. In this category, we therefore only evaluate the algorithms' quality based on the motions presented. A more difficult motion or task might include not just area, but also point- and line-contacts, whether the terrain is non-flat or the surface non-horizontal, if flight-phases with no contacts with the environment emerged, whether the contact points can slide as in skating, whether impulsive contacts with the environment are handled correctly or if the algorithm is able to generate motions for robots of various morphologies.

How reliable are solutions found?

A factor that is more difficult to quantify is how reliable the algorithm finds a solution to multiple sensibly proposed tasks without requiring retuning. We propose the three levels, low, medium and high.

Low – Depending on the given task, a variety of parameters have to be tuned or modified specially for this situation to make the algorithm produce feasible motion-plans. If a different terrain or gait is desired, a new set of parameters is required. Often, motion-planning algorithms with, e.g., hand-tuned hint-costs, that guide the algorithm into a particular direction fall into this category. Algorithms very sensitive to the stiffness and damping of the soft contact model could belong here as well. Finally, some multi-staged optimizations that require the number of stages to be set with specific objectives depending on the task specifications could be considered less reliable.

Medium – Given a feasible task, the algorithm often directly generates a valid motion plan. Sometimes a few algorithm parameters have to be slightly adapted, or the initialization modified to aid convergence. Then, however, even noticeable changes of the task specifications, goal or the terrain shape can be dealt with by the algorithm.

High – Given a feasible task, the algorithm generates a motion-plan with the default parameters. Even if the goal is changed, the terrain adapted, or the initial position modified a motion plan is reliably found with the same structure and default parameters.

How long does it take to find a solution?

Another important factor for motion-planning can be how long it takes for the algorithm to generate a plan. The longer this takes, the longer the robot has to "think" before reacting to unknown situations. To deploy robots in real-world scenarios and allow the robot to react to slippage and external pushes, it is helpful to keep the computational time as short as possible (e.g. 100 ms MPC loop). Although the computation times are strongly affected by the initialization, discretization of the problem and the type of robot and task, a rough time-range to solve a specific task, e.g., 1 s time-horizon, 4-step quadruped motion, can already be revealing.

How are the motion-plans verified?

The quality of a motion-planning algorithm is also determined by the physical correctness of its produced motion-plan. This verification can be done, ordered from most permissive to strict, through pure visualization, simulation or robot experiments.

Visualization – The most permissive form of verification is visually observing whether a motion-plan appears physically correct. This ensures that no grand modeling mistakes have been made and the rules of physics are at least approximately respected. However, these played-back motions only enforce those physical constraints that are also modeled in the problem. Contact forces could still be produced in the air, or inconsistent body accelerations can be generated, which cannot always be detected through visual inspection.

Simulation – A stricter form of verification is sending the motion-plan as a reference to a physics-based simulator, e.g., Gazebo [41], and either directly applying the torques or using a tracking controller to generate these. The robot and contact model in the physics simulator should be unknown (independent) to the algorithm. This ensures that trajectories are not just played back with the same, possibly false, physical assumptions. Motion-planning algorithms subjected to this type of verification can generally be regarded as of higher quality, since successful execution proves that the rules of physics implemented in the independent simulator have been correctly modeled in the algorithm and that there exists a tracking controller that can execute these motion-plans.

Experiment – The ultimate verification is executing the motion plan on a physical system. Sensor noise, model mismatches, and imperfect force tracking pose a variety of additional problems. While verification by simulation relies on the approximated physics of the simulator, the physics in the world are correct by definition. This type of verification not only proves that the motion-plan is physically correct, but also that it is robust against the above imperfections and that there exists a controller that can track the plan on the physical system.

2.3 State-of-the-art motion-planning algorithms

Each of our three papers begins by reviewing the state-of-the-art and placing the contributions in the overall context. This section is meant as an additional resource, comparing mostly the final version of our algorithm introduced in Chapter 5 to existing approaches. Table 2.1 shows an overview of the capabilities of the motion-planning algorithms developed in this thesis. The following highlights some other approaches capable of producing high-quality motions.

There exists a variety of successful work using the LIPM as a model to generate motion-plans [22], [40], [42]. In [21], [24] the authors optimize the motion-plan in an MPC fashion, resulting in a controller that can react to disturbances and handle model inaccuracies. Foothold selection and body motion generation are decoupled to obtain this speed. The approach presented by [43] combines the planar foothold selection and lateral CoM motion generation in the same online optimization problem. The nonlinear extension of this work, while still using the LIPM, permits to find also the optimal orientations for biped feet, avoids obstacles, and still runs online on an HRP-2 [15], [44]. Such a combined optimization of footholds and body motion is also shown in [45] using direct collocation with an SQP solver.

To eliminate some of the restrictions imposed by the LIPM, a variety of successful work for humanoid robots was developed using models such as the CD or the full RBD. Foothold and body motion planning are often decoupled to make the problem more tractable. Multiple-Shooting is used to generate and execute stair climbing using hand-rail support on a physical biped [46]. The approaches by [9], [47] successfully generate a variety of motions for a biped robot. This is achieved by optimizing a robot base path first and then finding static equilibrium configurations, creating an interactive multi-contact planning algorithm. These configurations are then used to generate physically feasible motions-plans using a CD model.

Similarly, physically correct motion-plans can also be generated by alternating between finding the robot momentum from a given kinematic motion-plan and then optimizing the corresponding contact forces and locations to fulfill the CD [6], [8], [48]. A similar approach to separate the problem is to alternately optimize the footholds and then the body motion and forces coherent with the CD [49]. The contact schedule and patches are either fixed or optimized separately. They show motion-plans for a full humanoid robot, including joints, and demonstrate how to track them efficiently.

Approaches based on dynamic programming, such as Differential Dynamic Programming (DDP) [50], [51], Iterative Linear Quadratic Gaussian Regulator (iLQG) [52], [53] and Sequential Linear Quadratic Programming (SLQ) algorithms can also be used to generate optimal motion-plans. Application of this work to humanoids has been done by [54] extended by [55]–[57] to additionally allow state and input constraints and efficiently optimize motion-plans for switched systems, such as a quadruped robot. The hybrid model can also be handled using projected dynamics [58] together with Collocation [59]

as shown by [60]. This method demonstrates a variety of impressive dynamic motions, such as walking, trotting and jumping with impulse forces on a physical quadruped robot. Because of the way this method treats switched systems it requires specifying the contact sequence in advance.

Since predefining the contact schedule in advance can restrict the achievable motions, algorithms that automatically determine this have been developed. Some approaches [55], [61], [62] tackle this by fixing only the contact sequence, but allowing the optimization to eliminate specific phases by setting their duration to zero. Others encode the notion of taking a step through binary variables [63]. The gait sequence, together with CD, is then solved using Mixed Integer Convex Programming (MICP). These motions-plans have been verified on a quadruped robot walking up inclined terrain and steps.

There exist further methods giving the optimizer full freedom to choose the contact schedule and timings by minimizing a cost function. Optimization methods like CMA-ES are then used to determine the controller parameters that generate the motion with lowest cost [64]–[66]. While the motion-plans can be quite sophisticated, this stochastic, and not gradient-based optimization technique can be slow in finding a solution. Other approaches based on the SLQ algorithm generate motions-plans for a wide range of systems quick enough to be used in an MPC control loop [57], [67], [68]. Motion generation in an MPC fashion using direct collocation and a SRBD model is also used in [16]. Another algorithm that produces highly complex motions, including multiple interacting robots with legs and arms was developed by [69]. This approach uses a full RBD model and minimizes the cost function using a Quasi-Newton method. Since these approaches are mostly based on a cost function, instead of hard constraints, they can require careful tuning of weighting parameters to achieve convergence for more complex tasks.

Another way to represent the legged locomotion problem is through a combination of cost and hard LCP constraints which model the contact with the environment. These successful approaches generate dynamic motion-plans that optimize also over the gait and step timings using the CD or the full RBD [7], [70], [71]. They directly include joint limits and other constraints in the optimization problem and show a variety of motions, including flight-phases. From a more critical standpoint, this modeling of the contact through an LCP can be computationally expensive, which might be mitigated through our phase-based end-effector parameterization. While this allows optimizing the step sequence efficiently, it also requires fixing the number of steps in advance, which the above algorithms can automatically discover.

Each of these approaches has their advantages and disadvantages. To compare different motion-planning algorithms, it is useful to evaluate them based on *all* criteria mentioned in Section 2.2, namely (i) which components are optimized (ii) how difficult are the shown tasks (iii) how reliable are solutions found (iv) how long does it take to find a solution (v) have they been verified in simulation and on a physical robot. For example, predefining the contact schedule in advance, or determining the footholds separately can be restrictive for more complex motions, but might be sufficient for a large set of tasks. The same holds for the used dynamic model, where the LIPM might be a good approximation for some

motions, but challenging to use for non-flat terrain and motions which require substantial adaptations of the body orientation. Furthermore, some use-cases might require more sophisticated motion-plans at the cost of longer computational time, while others demand quick re-planning in more simple environments. Since the weighting of these criteria depends on the use-case, Table 2.1 lists merely the capabilities of our algorithm. This permits point-by-point comparison of those individual criteria relevant to the desired task.

Table 2.1: *Capabilities (2.2) of our developed motion-planning algorithms.*

	Paper I	Paper II	Paper III
Dynamic model (accuracy)	LIPM (1.6)	LIPM (1.6)	SRBD (1.5)
Optimized components			
Base motion	2D	2D	6D
Footholds	2D	2D	3D
Step sequence	×	×	✓
Step timing	×	×	✓
Contact force	×	×	✓
Swingleg motion	×	×	✓
Number of steps	×	×	×
Difficulty of shown task			
Point- and line contacts	×	✓	✓
Non-flat terrain	×	×	✓
Inclined terrain	×	×	✓
Flight-phases	×	×	✓
Sliding contacts	×	×	×
Impulses	×	×	×
Number of end-effectors	4	4	1, 2, 4
Reliability	high	high	medium
Computation Time[1]	3 ms	35 ms	100 ms
Verification[2]	100 E	50 E, 0 S	10 E, 20 S

2.4 List of contributions

The specific contributions of this thesis mentioned in Section 2.1 are summarized below.

C1. Simultaneous foothold and CoM motion optimization to generate walking of the quadruped robot HyQ. This enables to create and shift arbitrarily oriented support polygons to aid the CoM to reach a desired goal state, without the need of an explicit footstep planner [38].

[1] Planning a flat ground, 1 s-horizon, 4-foothold motion for a quadruped.
[2] Percent of all visualized motions shown in Experiments (E) and simulation (S).

C2. Reformulation of the traditional ZMP-based legged locomotion problem into a standard TO formulation with the CoP as input and support-areas as modifiable input constraints. This permits easier comparison with existing methods in the TO domain [39].

C3. A vertex-based representation of the support-area constraint, which can treat point-, line-, and area-contacts uniformly. This allows generating non-traditional or mixtures of gaits (e.g., limping, walk-trot combinations), laying the foundation to traverse more complex terrains when appropriate footholds become sparse [39].

C4. Automatic generation of flight-phase motions using the SRBD. These emerge naturally whenever the terrain (e.g. a gap) or task specifications require it [17].

C5. Automatic gait discovery by a novel phase-based parameterization of end-effector motion and forces. This allows to keep the optimization variables continuous, while optimizing a discrete gait sequence [17].

C6 (Main). An algorithm utilizing all the above to efficiently generate motion-plans for legged robots determining gait, step-timings, footholds, contact forces, swing-leg motions and 6D body motion with flight-phases over non flat-terrain [72].

3

Paper I: Simultaneous foothold and body optimization

Alexander W. Winkler, Farbod Farshidian, Michael Neunert, Diego Pardo, Jonas Buchli. *Online Walking Motion and Foothold Optimization for Quadruped Locomotion.* In IEEE International Conference on Robotics and Automation (ICRA), pp. 5308-5313, 2017.

Abstract We present an algorithm that generates walking motions for quadruped robots without the use of an explicit footstep planner by simultaneously optimizing over both the CoM trajectory and the footholds. Feasibility is achieved by imposing stability constraints on the CoM related to the Zero Moment Point and explicitly enforcing kinematic constraints between the footholds and the CoM position. Given a desired goal state, the problem is solved online by a Nonlinear Programming solver to generate the walking motion. Experimental trials show that the algorithm can generate walking gaits for multiple steps in milliseconds that can be executed on a physical quadruped robot.

Paper: https://doi.org/10.1109/ICRA.2017.7989624
Video: https://youtu.be/EBW3lpr1tB8
Code: https://github.com/ethz-adrl/towr/tree/0.3.0-icra17

3.1 Introduction

The task of controlling a legged system seems trivial at first sight, as humans can walk with ease. However, making machines replicate this seemingly easy task is difficult, as demonstrated by the results of the DARPA Robotics Challenge. One limitation is that the underactuated base of legged systems cannot be directly controlled. Additionally, the contact forces with the ground used to generate movement of the base are restricted to pushing motions (unilateral forces) and must lie inside the friction cone.

Optimization has been successfully used in legged locomotion to generate such motions (see Fig. 3.1). It allows a high-level specification of a desired task and the specific motions of the joints to be generated by a mathematical program. Some of the approaches are based on TO, e.g. transformation of a continuous time Optimal Control (OC) problem into a discrete mathematical optimization problem and solved with off-the-shelf solvers ([7], [9], [37], [70], [73]–[75]). Other approaches solve the OC problem directly through efficient solvers based on Dynamic Programming [55], [67]. These formulations are powerful since they can deal with nonlinear dynamics and constraints and directly produce the optimal inputs $\boldsymbol{\tau}$ to the system. Some frameworks, originating more from the graphics community, directly formulate a mathematical optimization problem that generates the required motions [61], [64], [66], [69], [76].

All these successful approaches can generate complex motions for a variety of systems. However, their use to control real systems is not straight-forward. This is partially due to the discrepancy between the underlying rigid body dynamics model and the real robot, which makes it difficult to apply the optimized inputs to the real system directly. Additionally, for high dimensional systems such as legged robots, the resulting optimization problems are often difficult and time-consuming to solve. As they are commonly solved offline, this limits their use for real robots in which continuous re-planning and adapting to unknown disturbances is necessary.

A traditional way to reduce the complexity of walking motion generation is to decompose the problem into distinct sub-problems [18], [20], [77], [78] as seen in Fig. 3.1. Contact locations are often chosen by a heuristic planner which tries to roughly approximate a path towards the goal. Given these fixed footholds, a stable CoM trajectory is found and then tracked by a low-level controller that generates the inputs $\boldsymbol{\tau}$ to the system to execute the walking motion.

Given a set of steps to execute, the generation of the CoM trajectory uses the ZMP [79] stability criteria: The acceleration of the CoM must be chosen so that the generated ZMP always lies inside the convex hull of the feet in contact. Since the footholds are already given, this optimization has a much smaller search space than full-body optimizations and is easier to solve. This approach is used by [22] to develop a Linear Quadratic Regulator using a preplanned ZMP trajectory for a bipedal robot. In [18] this idea is extended by not specifying the ZMP trajectory to follow in advance, but formulating the stability criteria as a constraint and then solving the QP for the ZMP and CoM trajectory together. However,

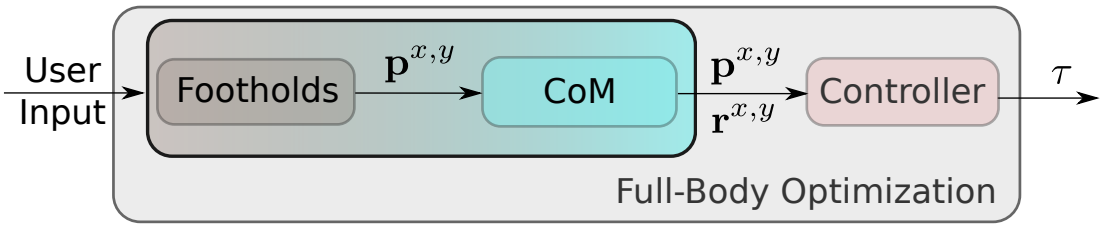


Figure 3.1: *Different approaches to solve a legged locomotion task: The traditional approach consists of finding footholds, a CoM trajectory and a controller that produces the control inputs τ to track this motion. Full-body Optimization combines these modules, directly producing the inputs to the system. The approach presented in this paper combines the foothold selection and CoM trajectory optimization, but leaves the generation of control inputs to the full-body controller.*

in this approach, the footholds must still be known in advance (e.g., by a footstep planner), since otherwise, the stability constraint is non-linear.

The drawback of these hierarchical approaches is that they decouple two inherently connected quantities: the foothold and the body movement. The contact forces and locations are the cause of the body movement, therefore specifying them beforehand based on some heuristics strongly restrains the feasible motions.

To mitigate this, approaches that optimize over both these quantities together have been proposed in [43], [80]. These successful approaches, however, require that the orientation of the support polygon edges is fixed in advance. This is a reasonable assumption for bipeds in single support phase, as the orientation of the feet might not be of high importance. However, for point feet quadruped robots the orientation of the support polygons change with every step.

3.1.1 Contribution

The novelties of this paper are *simultaneous* foothold and CoM motion optimization to generate quadrupedal walking, while explicitly enforcing kinematics limits (see Fig. 3.1). This enables to create and shift arbitrarily oriented support polygons to aid the CoM to reach a desired goal state, without the need of an explicit footstep planner. Additionally, since the generation of the full-state inputs are left to the controller, the optimization problem stays reasonably small to allow for online optimization in milliseconds. We show that the generated motions can be successfully executed on the quadruped robot seen in Fig. 3.2.

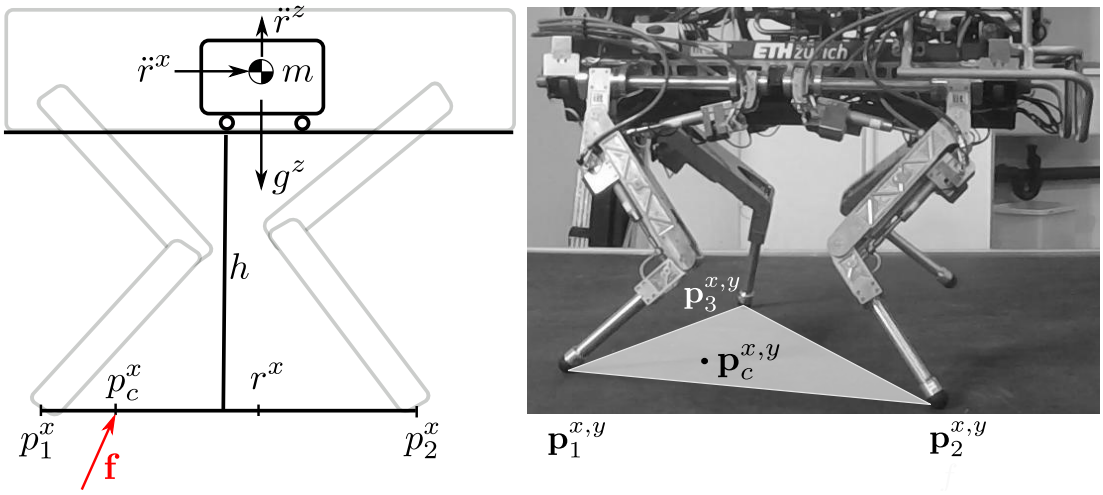


Figure 3.2: *A quadruped robot modeled as a cart-table. The position of the cart corresponds to the CoM and the width of the cart base (base of support) to the distance between the footholds. The quantity to control the motion of the system is the position (ZMP)* $\mathbf{p}_c$*, angle and magnitude* $|f|$ *of the resultant contact force* $\mathbf{f}$ *(red). The white triangle shows the base of support while swinging the right-hind leg.*

3.2 Approach

In order to understand the important connection between footholds and the CoM motion and the reasoning in optimizing these together, we briefly recap the physical aspects of legged locomotion, then present the formulation of the developed framework and describe the components.

3.2.1 An abstracted view of legged locomotion

Consider a robot modeled as a cart-table as seen in Fig. 3.2. By applying torque $\boldsymbol{\tau}$ in the joints, reaction forces can be generated in the footholds. The resulting force $\mathbf{f}$ is the equivalent force that has the same effect as the combination of the individual contact forces and directly influences the acceleration of the CoM. The difficulty in the control of legged system arises from the constraints on this resulting contact force. First, the force cannot pull, but only push into the ground (unilateral constraints). Secondly, a resulting force can never act outside of the footholds p_1^x and p_2^x that are producing it. Assuming sufficient friction, the resulting contact force is constrained by $p_c^x \in [p_0^x, p_1^x]$ and $f^z > 0$.

The Euler equation of motion around p_c^x of the resultant contact force with no change in angular momentum can be stated as

$$\ddot{r}^x = \frac{g^z + \ddot{r}^z}{h}(r^x - p_c^x). \tag{3.1}$$

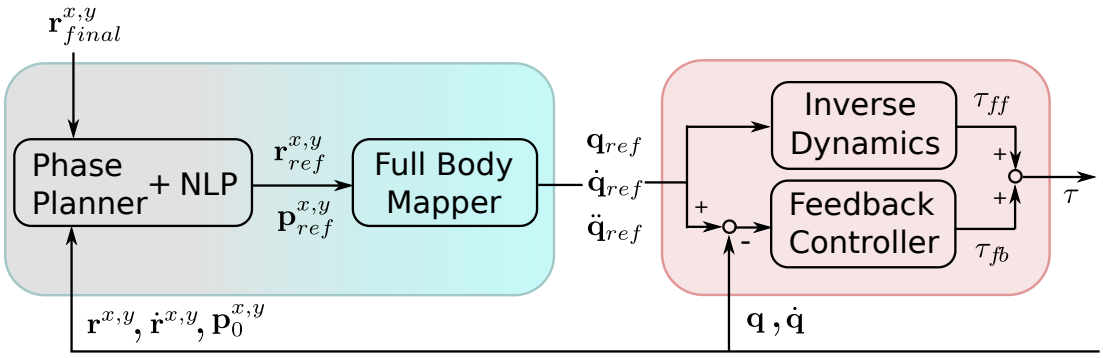


Figure 3.3: *The complete pipeline from planning to executing optimal walking motions: A Phase Planner decides on the amount and sequence of steps, an NLP then solves the optimization problem that is then mapped to the full dimensions of the system by the Full Body Mapper and ultimately executed on the system by the low-level controller.*

It can be seen that this equation is only influenced by the position p_c^x of the resulting contact force, not its magnitude. To walk at a constant height the above equation with $\ddot{r}^z = 0$ must hold. In this case, the position $\mathbf{p}_c$ of the resulting contact force is called the ZMP or Center of Pressure (CoP). When more acceleration is desired, the distance between the resultant contact force and the CoM must increase as well to keep the system from rotating around the foothold. If the desired acceleration is too large, the resultant contact force would have to act outside the base of support of the cart table $p_c^x \notin [p_0^x, p_1^x]$, which is physically not possible. This implies the gait is not dynamically balanced and the system is starting to fall with $\ddot{r}^z \neq 0$. There are two ways to avoid this:

1. Restrict the CoM acceleration $\ddot{\mathbf{r}}^{xy}$, so that it can be generated by a resultant contact force *inside* the base of support.

2. Modify the base of support to accommodate the CoM acceleration. This requires shifting the footholds $\mathbf{p}^{xy}$ to create the support base exactly where it is needed to generate the current body acceleration.

It becomes clear that the body acceleration and the footholds are strongly connected, the footholds serving as an aid to allow the CoM to move where it desires. The following describes the developed framework that finds optimal solutions for these two inherently connected quantities together.

3.2.2 Overview

The developed framework as seen in Fig. 3.3 takes as input the current state and a desired goal state to create full-body states to execute a walking motion for a legged robot. It is composed of a Phase Planner that decides on the amount and sequence of steps (not their location), a NLP solver that produces a CoM motion plan and footholds, a Full

Body Mapper that maps the Cartesian output of the NLP to a full-body (base and joints) motion of the robot and finally a controller that generates the required torques to track the motion.

The complete motion is divided into separate phases, within which the feet in contact do not change (Fig. 3.4). The leg to use to establish contact $\mathbf{p}_i^{xy}$ is determined by the *Phase Planner*. For quadrupedal walking, a standard sequence to follow is the lateral sequence walk left hind → left front → right hind → right front. Depending on the distance to the goal and the maximum step length the planner adds n steps following this sequence, or none if the distance is close enough to only move the body. Each of these phases j is therefore represented by a fixed set of legs in contact, e.g., in phase 3: {LH, LF, RF}, and the duration T_j of each phase. Additionally, the initial position of the legs in contact is also given and cannot be altered.

In order to generate the continuous CoM motion through optimization, we must first parametrize it by a finite number of decision variables. We therefore represent each phase j as a polynomial defined as

$$\mathbf{r}_j^{xy}(t, \mathbf{a}_j) = \begin{bmatrix} r^x \\ r^y \end{bmatrix} = \sum_{k=0}^{5} \mathbf{a}_{j,k} t^i, \tag{3.2}$$

where the values in xy-direction are parametrized by a fifth-order polynomial with coefficients $\mathbf{a}_{j,k} \in \mathbb{R}^2$ and the complete phase polynomial by $\mathbf{a}_j \in \mathbb{R}^{12}$. This gives the optimizer enough freedom to shape the motion to respect the imposed constraints. As motivated previously, the optimizer is also able to modify the position of the m footholds of each step defined as

$$\mathbf{p}_i^{xy} = \begin{bmatrix} p_i^x & p_i^y \end{bmatrix}^T.$$

Given the n phases, the solver optimizes over the decision variables to find the optimal motion and m footholds, minimizing a performance criteria while fulfilling equality and inequality constraints. This can be stated as the NLP

$$\begin{aligned} \text{find} \quad & \mathbf{a}_1, \ldots, \mathbf{a}_n, \ \mathbf{p}_1^{xy}, \ldots, \mathbf{p}_m^{xy} \\ \text{subject to} \quad & (3.6), (3.7), (3.8) \\ & \mathbf{a}_1, \ldots, \mathbf{a}_n = \arg\min (3.3). \end{aligned}$$

The following describes the constraints (3.6), (3.7), (3.8) and cost (3.3) necessary to generate the walking motions.

3.2.3 Performance criteria

The CoM should accelerate as little as possible during the motion, as introduced in [18]. This facilitates tracking, reduces required joint torques and energy consumption and produces more natural looking motions. Therefore, the total xy-acceleration for all phases j

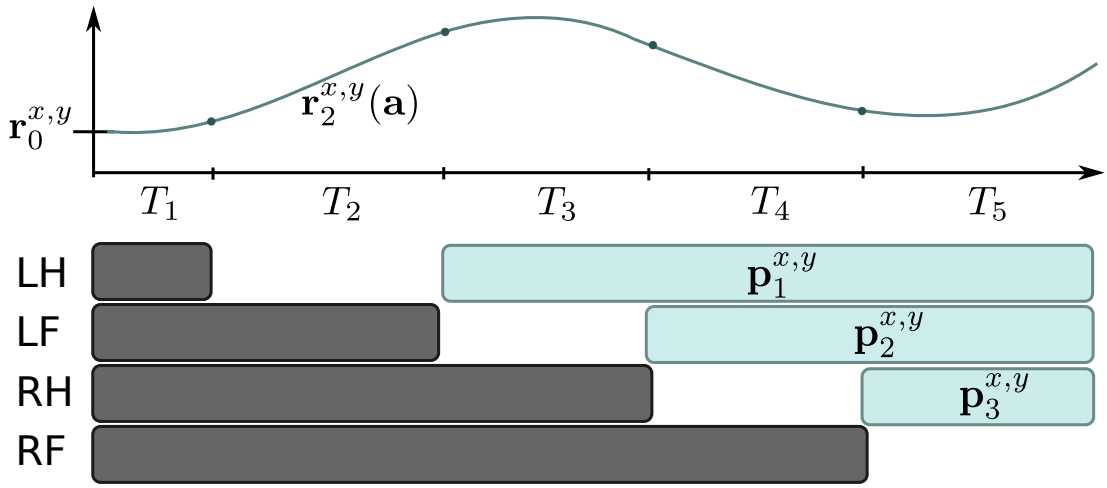


Figure 3.4: *The structure of the planned motion. The boxes symbolize contact of the respective leg, left-front (LF), right-front (RF), left-hind (LH), right-hind (RH). The gray boxes are the fixed initial contacts at the start of the motion. The optimizer finds the best positions for the $m = 3$ contacts $\mathbf{p}_i^{xy}$ shown in blue, as well as the $n = 5$ CoM polynomials.*

(see Fig. 3.4) is given by

$$J(\mathbf{a}) = \sum_{j=1}^{n} \int_{t=0}^{T_j} \ddot{r}_j^{x^2}(t, \mathbf{a}_j^x) + \ddot{r}_j^{y^2}(t, \mathbf{a}_j^y)\, \mathrm{d}t. \tag{3.3}$$

3.2.4 Dynamic feasibility constraint

To ensure that the planned motion is dynamically feasible, we impose constraints between the CoM motion and the footholds as first introduced in [18]. For each stance we represent the base of support by the convex hull of the current footholds. Each of the edges of the convex hull, represented by a line, is defined as $0 = \begin{bmatrix} x & y & 1 \end{bmatrix} \mathbf{n}$. For example, the edge e defined by the footholds $\mathbf{p}_1^{xy}$ and $\mathbf{p}_2^{xy}$ is calculated by

$$\mathbf{n}_e = \begin{bmatrix} n^x \\ n^y \\ d_o \end{bmatrix} = \frac{1}{d} \begin{bmatrix} p_1^y - p_2^y \\ p_2^x - p_1^x \\ p_1^x p_2^y - p_2^x p_1^y \end{bmatrix} \tag{3.4}$$

where d is the distance between the footholds (Euclidean norm). In order to satisfy (3.1) without resorting to vertical $\ddot{c}^z$ accelerations, the ZMP must be inside the convex hull $\mathcal{P}(\mathbf{p}^{xy})$ of the current footholds, so

$$\mathbf{p}_c^{xy} = \mathbf{r}^{xy}(\mathbf{a}) - \frac{h}{g^z} \ddot{\mathbf{r}}^{xy}(\mathbf{a}) \in \mathcal{P}(\mathbf{p}^{xy}). \tag{3.5}$$

In the one-dimensional case this reduces to $p_1^x < z^x < p_2^x$. For two dimensions, the constraint translates to being on the inside of each line composing the convex base of support.

However, the cart-table model does not adequately capture the dynamics of the real system. To account for this, we require the ZMP to stay away from the edge of the support polygon by a margin m. Since the line equation is normalized and represents the orthogonal distance of a point (x,y) to the line, including this margin, is straightforward. This condition that depends both on the CoM and the footholds is evaluated at every discrete time t_k. For every edge of the current support polygon, the nonlinear inequality constraint

$$\begin{bmatrix} p_c^x[t_k] & p_c^y[t_k] & 1 \end{bmatrix} \mathbf{n}_e > m \tag{3.6}$$

must hold. This equation highlights the strong interconnection between the location of footholds, which affect the line coefficients and the acceleration of the CoM that affects the location of the ZMP. Both of them can be used to ensure stability according and are optimized simultaneously in our formulation.

3.2.5 Kinematic reachability constraint

The previous constraint gives the optimizer the possibility to either adapt the CoM accelerations to ensure stability or modify the footholds to respect this constraint. This flexibility increases the range of possible motions, but also necessitates the additional kinematic constraints that the desired foothold must be reachable with the leg from the current CoM position.

Apart from the walking height h and the mass m of the system, the algorithm so far does not need any additional knowledge about the actual system that is being controlled. This constraint, however, is specific to the kinematics of each system. The farther the quadruped in Fig. 3.2 moves its CoM towards p_2, the more the leg at p_1 must extend to remain in contact. At some point, the difference between p_1 and r is too large and exceeds the kinematic range of the robot. To ensure reachability, we check at every discrete time t_k how far each foothold is from its nominal stance position ${}^B\mathbf{p}_{nom}^{xy}$ of the respective leg, expressed in the frame attached to the robots CoM. The constraint for each foothold at time t_k is then formulated as

$$0 < \left\| \mathbf{p}_i^{xy} - \mathbf{r}^{xy}[t_k] - {}^B\mathbf{p}^{xy}{}_{nom} \right\|_2 < r \tag{3.7}$$

This restricts the foothold to deviate less that a radius r from its nominal position. The value for this allowable deviation can be approximated using Inverse Kinematics: The limits of the joints can be mapped to Cartesian space $q_{max} \mapsto (x, y)_{max}$ to obtain an approximation for r. Again, the interconnection between the CoM and the foothold shows itself. This enables the optimizer to explicitly respect kinematics limits by knowing the actual CoM position, without resorting to heuristics used in many footstep planners.

3.2.6 CoM continuity and goal constraints

We ensure continuous position and velocity at the phase junctions as well as equality with the initial and final conditions. This enables smooth motions between phases, starting the optimization from the current robot state and handling user specified goal states. Combining the CoM position and velocity as $\mathbf{x} = \begin{bmatrix}\mathbf{r}^{xy} & \dot{\mathbf{r}}^{xy}\end{bmatrix}^T$, the equality constraints of the NLP can be stated as

$$\begin{aligned} \mathbf{x}_1(0) &= \mathbf{x}_{initial} \\ \mathbf{x}_j(T_j) &= \mathbf{x}_{j+1}(0), \quad \text{for } j = 1, \dots, n \\ \mathbf{x}_n(T_n) &= \mathbf{x}_{final}. \end{aligned} \tag{3.8}$$

3.2.7 Mapping CoM and footholds to full-body states

The results of the reduced dimension optimization are now mapped back to the full body state. The base state is reconstructed using the optimized CoM motion $\mathbf{r}^{xy}$, assuming that the origin of the base frame is located at the CoM of the robot. Using the constant base height h and zero orientation, the 6 degrees of freedom base state is set to

$$\mathbf{q}_{b,ref}(t) = \begin{bmatrix}0 & 0 & 0 & r^x(t) & r^y(t) & h\end{bmatrix}^T. \tag{3.9}$$

The joint state of the stance legs can be constructed through each foothold, a predefined polynomial swingleg trajectory and the base position using Inverse Kinematics

$$\mathbf{q}_{j,ref}(t) = \mathrm{IK}(\mathbf{q}_{b,ref}(t), \mathbf{p}^{xy}). \tag{3.10}$$

Combining the above, the optimized full body state for the controller to track is given by

$$\mathbf{q}_{ref}(t) = \begin{bmatrix}\mathbf{q}_{b,ref}(t) & \mathbf{q}_{j,ref}(t)\end{bmatrix}^T. \tag{3.11}$$

3.2.8 Tracking the planned motion

As can be seen from Fig. 3.3 the optimization framework produces desired full body accelerations $\ddot{\mathbf{q}}_{ref}$ for the controller to follow. The inverse dynamics controller is responsible for generating required joint torques $\boldsymbol{\tau}$ to create a given acceleration. This is done based on the rigid body dynamics model of the system

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_{ref} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}_c(\mathbf{q})^T\mathbf{f}, \tag{3.12}$$

with the joint space inertia matrix $\mathbf{M}$, the effect of Coriolis forces on the joint torques $\mathbf{C}$, the selection matrix $\mathbf{S}$ which prohibits from actuating the floating base state directly and the contact Jacobian $\mathbf{J}_c$ which maps Cartesian contact forces $\mathbf{f}$ to joint torques. Although

(3.6) ensures that the required contact forces $\mathbf{f}$ to produce $\ddot{\mathbf{q}}_{ref}$ are physically feasible, the actual value is not a part of the optimization. This prohibits from solving the underdetermined system of equations directly for $\boldsymbol{\tau}$ and $\mathbf{f}$. However, by knowing which feet are in contact we can calculate a projection operator $\mathbf{P} = \mathbf{I} - \mathbf{J}_c^{+}\mathbf{J}_c$ [58], [81]. With this we can eliminate the contact forces $\mathbf{f}$ from (3.12) and calculate the required joint torques through

$$\boldsymbol{\tau} = (\mathbf{P}\mathbf{S}^T)^{+}\ \mathbf{P}(\mathbf{M}\ddot{\mathbf{q}}_{ref} + \mathbf{C}). \tag{3.13}$$

There always exist discrepancies between the used models (cart-table model, inverse dynamics model) and the real system. To cope with these, it is essential to incorporate feedback into the control loop. We do this by adding an operational space [82] PD controller on the base position and velocity and a low gain PD controller on the joint positions and velocities for more accurate reference tracking.

3.3 Results

3.3.1 PC and robot hardware

We use the Interior Point (or Barrier) method solver Ipopt [35] to solve the NLP. The solver first finds values for the decision variables that satisfy all constraints and then gradually increases the importance of the performance metric to obtain optimal solutions. The results were obtained using C++ Code on an Intel Core i7/2.8 GHz Quadcore Laptop. The Jacobian of the constraints and the gradient of the cost function are provided analytically, which is essential for performance. Ipopt iteratively approximates the Hessian of the Lagrangian through the quasi-Newton L-BFGS algorithm.

We evaluate the performance of this locomotion framework on the hydraulically actuated quadruped robot HyQ [5]. The robot weighs approximately 75 kg, is fully torque controlled and equipped with precision joint encoders and an Inertial Measurement Unit (IMU). State estimation is performed on board, fusing IMU and joint encoder values [83]. The lowest level torque control loop runs at 1000 Hz, while new position, velocity, and torque references are set at 250 Hz. The rigid body dynamics model (3.12) was generated with [84].

3.3.2 Experiments

We demonstrate the performance and robustness of the locomotion framework through a consecutive run of walking motions to different goal positions. For close goal positions the phase planner determines no steps are necessary, so only the CoM is shifted to the commanded positions. The other motions each require various optimized motions (forward walk, sideways walk) to reach the target. The time discretization to enforce the dynamics in (3.6) is 0.2 s, whereas kinematic reachability (3.7) is enforced every 0.3 s. We insert an

Table 3.1: *Results of the online optimization for various tasks*

	Walking Direction			
	Stance	Side	Forward	Backward
Goal x,y [cm]	0,0	0,15	40,-20	-30,0
Footholds/Steps	0	4	8	8
Planning horizon [s]	1.5	4.3	7.1	7.1
Optimization Variables	12	68	124	124
Constraints	92	211	327	327
Iterations	2	28	29	29
Objective value	0.09	0.4	0.9	1.59
Nonzeros in Jacobian	708	1756	2844	2844
Solving Time [ms]	**3**	**58**	**72**	**75**

initial stance phase of T_0=1.5 s to allow sufficient time to shift the body before taking the first step. For all tasks, the step duration is $T_i = 0.7$ s, the walking height is h=0.58 m and a margin of m=8 cm to reduce the size of the support polygons is specified. We initialize the solver with the robot standing in default stance for the duration of the trajectory.

A quantitative summary of the optimization results for the performed tasks can be seen in Table 3.1 and a visualization of the optimized footholds and body trajectory for walking 0.5 m forward is shown in Fig. 3.5. Additionally, the reader is strongly encouraged to view the accompanying video at `https://youtu.be/EBW3lpr1tB8`, as it provides the most intuitive way to judge the performance of our framework.

3.3.3 Discussion

The following section analyzes the experimental results, highlighting some features of the proposed framework.

3.3.3.1 Footstep selection towards goal

As observed in the tasks, the user specifies only the goal state. This high-level input is the actual relevant information, as it is often of secondary importance with which footsteps the robot reaches this goal. The only reason the footsteps follow the direction of the goal is because the system knows it needs to move the body there (3.8), but has to maintain stability (3.6) while staying within the kinematic range of the legs (3.7). By not fixing the footsteps beforehand, we allow the optimizer to modify the base of support as needed to perform the desired motion. This allows to perform the side-, diagonal- and backwards-motions without any hand-tuned estimations of where to best place the feet.

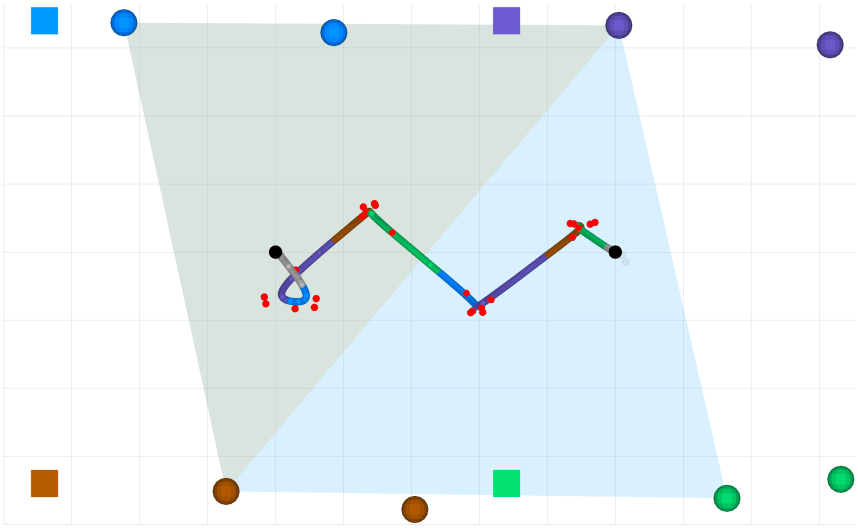


Figure 3.5: *8-step quadruped walking motion from left to right to move the CoM to a goal at x=0.5 m. The colors of the CoM trajectory correspond to the swingleg at that time, gray symbolizes a four-leg support (stance) phase. The initial stance is shown by the squares, the optimized footholds by the circles. The two support areas for swinging the right-front green leg and then the left-hind blue leg are shown. The corresponding ZMP (red dots) stays inside these areas by a margin of m=8 cm. It can be seen that by manipulation of the acceleration, the ZMP accumulates towards the center of support polygons for each phase. Additionally, the footholds automatically extend towards the defined goal position in order to create appropriate support areas for the CoM.*

3.3.3.2 Explicitly enforcing kinematic limits

In traditional approaches, the footsteps are decided by a footstep planner that for instance tries to keep the step length bounded or the footholds close to an *estimated* position of the CoM. However, the actual CoM is only discovered subsequently, when these fixed foothold locations are fed to a CoM planner. This decoupling makes it impossible for the footstep planner to be sure to have chosen footsteps within the kinematic range of the robot. An essential benefit of this approach is that it can explicitly enforce this constraint (3.7), as it combines footstep and CoM optimization. This is demonstrated in the in the video, showing that the chosen footsteps never overextend the legs.

3.3.3.3 Online optimization

The optimization of a reduced dimensional model allows us to obtain solving times of magnitudes lower than most full-body optimization approaches. As seen in Table 3.1 we can generate 7 s body trajectories in less than 100 ms, even though the constraints are nonlinear due to the combined optimization. This is shown in the video by specifying a new goal position on-the-fly and the robot almost immediately starting to approach it.

This online planning is an essential factor in controlling real systems, where plans have to be frequently adjusted to account for changing environments or inaccurate execution. The very short solving times demonstrate that the presented framework can be used in a Model-Predictive Control fashion in the future.

3.4 Conclusions

We presented an approach for online and simultaneous optimization of two core and very connected components of locomotion, namely footholds and CoM motion. The results show that our online optimization provides a variety of feasible walking motions for a quadruped to execute. It is important to note that the demonstrated motions can also be achieved by more hand-tuned traditional ways of specifying footsteps and body movements. However, the generality of the presented approach and its more complete view on the problem has considerable potential for extension and future work.

As the optimization problem is formulated as an NLP, it is possible to include various types of optimization variables, costs and constraints into the formulation (linear, quadratic, nonlinear, etc.). This allows us to optimize also over the remaining degrees of freedom of the body or the phase durations. Using the body orientation, for instance, can help in reaching footholds that otherwise exceed the range of motion. Adapting the phase durations will allow the optimizer to take quicker steps when required (e.g., when pushed). Another feature to be explored is the inclusion of terrain costs such as slope and height deviation in the performance criteria. This can allow the robot to navigate over uneven and challenging terrain, selecting those footholds with the lowest cost to generate steps to take the robot to a desired goal state. Finally, the speed of the optimization makes it possible to re-plan the motions in a Model-Predictive Control fashion at nearly the control loop frequency. Walking to defined goal states as well as recovering from unexpected external pushes or changes in the environment can all emerge from the same controller.

4

Paper II: Vertex-based ZMP constraints

Alexander W. Winkler, Farbod Farshidian, Diego Pardo, Michael Neunert, Jonas Buchli. *Fast Trajectory Optimization for Legged Robots using Vertex-based ZMP Constraints.* In Robotics and Automation Letters (RA-L), pp. 2201–2208, 2017.

Abstract This paper combines the fast ZMP approaches that work well in practice with the broader range of capabilities of a Trajectory Optimization formulation, by optimizing over body motion, footholds and Center of Pressure simultaneously. We introduce a vertex-based representation of the support-area constraint, which can treat arbitrarily oriented point-, line-, and area-contacts uniformly. This generalization allows us to create motions such as quadrupedal walking, trotting, bounding, pacing, combinations and transitions between these, limping, bipedal walking and push-recovery all with the same approach. This formulation constitutes a minimal representation of the physical laws (unilateral contact forces) and kinematic restrictions (range of motion) in legged locomotion, which allows us to generate diverse motions in less than a second. We demonstrate the feasibility of the generated motions on a physical quadruped robot.

Paper: https://doi.org/10.1109/LRA.2017.2723931
Video: https://youtu.be/5WLeQMBuv30
Code: https://github.com/ethz-adrl/towr/tree/0.5.0-ral17

4.1 Introduction

Planning and executing motions for legged systems is a complex task. A central difficulty is that legs cannot pull on the ground, e.g., the forces acting on the feet can only push upwards. Since the motion of the body is mostly generated by these constrained (=unilateral) contact forces, this motion is also restricted. When leaning forward past the tip of your toes, you will fall, since your feet cannot pull down to generate a momentum that counteracts the gravity acting on your CoM. Finding motions that respect these physical laws can be done by various approaches described in the following.

A successful approach to tackle this problem is through full-body TO, in which an optimal body and end-effector motion plus the appropriate inputs are discovered to achieve a high-level goal. This was demonstrated by [7], [55], [60], [67], [69], [70], [73], [85] resulting in an impressive range of motions for legged systems. These TO approaches have shown great performance, but are often time-consuming to calculate and not straight-forward to apply on a real robot. In [64] the authors generate a wide range of quadruped gaits, transitions and jumps based on a parameterized controller and periodic motions. While the resulting motions are similar to ours; the methods are very different: While our approach is based on TO with physical constraints, [64] optimizes controller parameters based mainly on motion capture data.

Previous research has shown that to generate feasible motions to execute on legged systems, non-TO approaches also work well, although the motions cannot cover the range of the methods above. One way is to model the robot as a LIPM and keep the ZMP [79] inside the convex hull of the feet in stance. This approach has been successfully applied to generate motions for biped and quadruped walking [9], [18], [20], [22], [77], [78]. However, these hierarchical approaches use predefined footholds, usually provided by a higher-level planner beforehand that takes terrain information (height, slope) into account. Although this decoupling of foothold planning and body motion generation reduces complexity, it is unnatural, as the primary intention of the footholds is to assist the body to achieve a desired motion. By providing fixed foot-trajectories that the body motion planner cannot modify, constraints such as stability or kinematic reachability become purely the responsibility of the lower-level body motion planner, artificially constraining the solution. A somewhat reverse view of the above are Capture Point (CP) [40] approaches, which have been successfully used to generate dynamic trotting and push recovery motions for quadruped robots [19], [86]. A desired body motion (usually a reference CoM velocity) is given by a high-level planner or heuristic, and a foothold/CoP trajectory must be found that generates it.

Because of the dependency between footholds and body motion, approaches that optimize over both these quantities simultaneously, while still using a simplified dynamics model, have been developed [38], [44], [61], [80], [87], [88]. This reduces heuristics while increasing the range of achievable motions, but still keeps computation time short compared to full body TO approaches. These approaches are most closely related to the work presented in this paper.

The approaches [44], [80], [87], [88] demonstrate impressive performance on biped robots. One common difficulty in these approaches, however, is the nonlinearity of the CoP constraint with respect to the orientation of the feet. In [80], [88] the orientation is either fixed or solved with a separate optimizer beforehand. In [44] the nonlinearity of this constraint is accepted and the resulting nonlinear optimization problem solved. However, although the orientation of the individual feet can be optimized over in these approaches, a combined support-area with multiple feet in contact is often avoided, by not sampling the constraint during the multi-support phase. For biped robots neglecting the constraint in the double-support phase is not so critical, as these take up little time during normal walking. For quadruped robots, however, there are almost always two or more feet in contact at a given time, so the correct representation of the dynamic constraint in this phase is essential.

We, therefore, extend the capabilities of the approaches above by using a vertex-based representation of the CoP constraint, instead of hyperplanes. In [89] this idea is briefly touched; however, the connection between the corners of the foot geometry and the convexity variables is not made and thereby the restriction of not sampling in the multi-support phase remains. Through our proposed formulation, single- and multi-stance support areas can be represented for arbitrary foot geometry, including point-feet. Additionally, it allows representing arbitrarily oriented 1D-support lines, which wasn't possible with the above approaches. Although not essential for biped walk on non-point feet, it is a core necessity for dynamic quadruped motions (trot, pace, bound). This is a reason why ZMP-based approaches have so far only been used for quadrupedal walking, where 2D-support areas are present.

The approach presented in this paper combines the LIPM-based ZMP approaches that are fast and work well in practice with the broader range of capabilities of a TO formulation. A summary of the explicit contributions with respect to the papers above are:

- We reformulate the traditional ZMP-based legged locomotion problem [22] into a standard TO formulation with the CoP as input, clearly identifying state, dynamic model and path- and boundary-constraints, which permits easier comparison with existing methods in the TO domain. Push recovery behavior also naturally emerges from this formulation.
- We introduce a vertex-based representation of the CoP constraint, instead of hyperplanes. which allows us to treat arbitrarily oriented point-, line-, and area-contacts uniformly. This enables us to generate motions that are difficult for other ZMP-based approaches, such as bipedal walk with double-support phases, point-feet locomotion, various gaits as well as arbitrary combinations and transitions between these.
- Instead of the heuristic shrinking of support areas, we introduce a cost term for uncertainties that improve the robustness of the planned motions.

We demonstrate that the problem can be solved for multiple steps in less than a second to generate walking, trotting, bounding, pacing, combinations, and transitions between these,

limping, biped walking and push-recovery motions for a quadruped robot. Additionally, we verify the physically feasibility of the optimized motions through demonstration of walking and trotting on a real 80 kg hydraulic quadruped.

4.2 Method

4.2.1 Physical model

We model the legged robot as a Linear Inverted Pendulum Model (LIPM), with its CoM $\mathbf{r}=(r_x, c_y)$ located at a constant height h. The touchdown position of the pendulum with the ground (also known as ZMP or CoP) is given by $\mathbf{p}_c=(p_{c,x}, p_{c,y})$ as seen in Fig. 4.1. The CoM acceleration $\ddot{\mathbf{r}}$ is predefined by the physics of a tipping pendulum

$$\begin{bmatrix}\dot{\mathbf{r}}\\ \ddot{\mathbf{r}}\end{bmatrix} = \mathbf{F}(\mathbf{x},\mathbf{u}) = \begin{bmatrix}\dot{\mathbf{r}}\\ (\mathbf{r}-\mathbf{p}_c)gh^{-1}\end{bmatrix}. \tag{4.1}$$

The second-order dynamics are influenced by the CoM position $\mathbf{r}$, the CoP $\mathbf{p}_c$ and gravity g. This model can be used to describe a legged robot since the robot can control the torques in the joints, thereby the contact forces and through these the position of the CoP. Looking only at the x-direction (left image in Fig. 4.1), if the robot decides to lift the hind leg, the model describing the system dynamics is a pendulum in contact with the ground at the front foot $\mathbf{p}^{RF}$, so $\mathbf{p}_c=\mathbf{p}^{RF}=(p_x, p_y)^{RF}$. Since this pendulum is nearly upright, the CoM will barely accelerate in x; the robot is balancing on the front leg. However, lifting the front leg can be modeled as placing the pendulum at $\mathbf{p}_c = \mathbf{p}^{LH}$, which is strongly leaning and thereby must accelerate forward in x. By distributing the load between the legs, the robot can generate motions corresponding to a pendulum anchored anywhere between the contact points, e.g., $\mathbf{p}_c \in \mathcal{P}$ (see Fig. 4.1). Therefore, the CoP $\mathbf{p}_c$ is considered the input to the system and an abstraction of the joint torques and contact forces.

4.2.2 Trajectory Optimization problem

We want to obtain the inputs $\mathbf{u}(t)$ that generate a motion $\mathbf{x}(t)$ from an initial state $\mathbf{x}_0$ to a desired goal state $\mathbf{x}_T$ in time T for a robot described by the system dynamics $\mathbf{F}(\mathbf{x},\mathbf{u})$, while respecting some constraints $\mathbf{h}(\mathbf{x},\mathbf{u}) \leq 0$ and optimizing a performance criteria J.

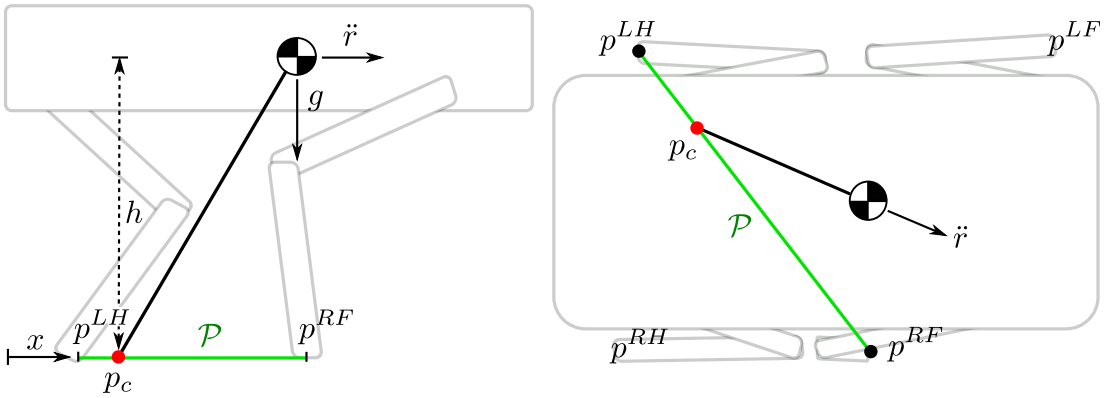


Figure 4.1: *Modeling of a quadruped robot by a LIPM with the right-front* $\mathbf{p}^{RF}$ *and left-hind* $\mathbf{p}^{LH}$ *legs in contact. Through joint torques the robot can control the center of pressure* $\mathbf{p}_c$ *and thereby the motion of the CoM* $\ddot{\mathbf{r}}$*. However,* $\mathbf{p}_c$ *can only lie inside the convex hull (green line) of the contact points.*

This can be formulated as a continuous-time TO problem

$$\begin{aligned}
\text{find} \quad & \mathbf{x}(t), \mathbf{u}(t), \quad \text{for } t \in [0, T] && \text{(4.2a)}\\
\text{subject to} \quad & \mathbf{x}(0) - \mathbf{x}_0 = 0 \quad \text{(given initial state)} && \text{(4.2b)}\\
& \dot{\mathbf{x}}(t) - \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t)) = 0 \quad \text{(dynamic model)} && \text{(4.2c)}\\
& \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0 \quad \text{(path constraints)} && \text{(4.2d)}\\
& \mathbf{x}(T) - \mathbf{x}_T = 0 \quad \text{(desired final state)} && \text{(4.2e)}\\
& \mathbf{x}(t), \mathbf{u}(t) = \arg\min J(\mathbf{x}, \mathbf{u}). && \text{(4.2f)}
\end{aligned}$$

The dynamics are modeled as those of a LIPM (4.1), whereas the state and input for the legged system model are given by

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{r} & \dot{\mathbf{r}} & \mathbf{p}^1, \alpha^1, \ldots, \mathbf{p}^{n_f}, \alpha^{n_f} \end{bmatrix}^T \tag{4.3}$$

$$\mathbf{u}(t) = \mathbf{p}_c, \tag{4.4}$$

which includes the CoM position and velocity and the position and orientation of the n_f feet. The input $\mathbf{u}(t)$ to move the system is the generated CoP $\mathbf{p}_c$, abstracting the usually used contact forces or joint torques.

4.2.3 Specific case: Capture Point

We briefly show that this general TO formulation, using the LIPM model, also encompasses Capture Point methods to generate walking motions. Consider the problem of finding the position to step with a point-foot robot to recover from a push. With the initial position $\mathbf{r}_0$ and the initial velocity $\dot{\mathbf{r}}_0$ generated by the force of the push we have $\mathbf{x}_0 = (\mathbf{r}_0, \dot{\mathbf{r}}_0)$. The

robot should come to, and remain, at a stop at the end of the motion, irrespective of where and when, so we have $\dot{\mathbf{r}}_{T\to\infty}=0$. We parametrize the input by the constant parameter $\mathbf{u}(t)=\mathbf{p}_{c,0}$, as we only allow one step with a point-foot. We allow the CoP to be placed anywhere, e.g. no path constraints (4.2d) and do not have a preference as to how the robot achieves this task, e.g. $J(\mathbf{x},\mathbf{u})=0$.

Such a simple TO problem can be solved analytically, without resorting to a mathematical optimization solver (see Appendix A.3). The point on the ground to generate and hold the CoP in order to achieve a final steady-state maintaining zero CoM velocity becomes

$$\mathbf{u}(t) = \mathbf{p}_{c,0} = \mathbf{r}_0 + \sqrt{hg^{-1}}\dot{\mathbf{r}}_0. \tag{4.5}$$

This is the one-step Capture Point (CP), originally derived by [40] and the solution of our general TO formulation (4.2) for a very specific case (e.g one step/control input, zero final velocity).

4.2.4 General case: legged locomotion formulation

Compared to the above example, our proposed formulation adds the capabilities to represent motions of multiple steps, time-varying CoP, physical restrictions as to where the CoP can be generated and preferences which of the feasible motions to choose. This TO formulation is explained on a high-level in the following, corresponding to Fig. 4.2, whereas more specific details of the implementation are postponed to the next section.

4.2.4.1 Unilateral forces

We differentiate between the CoP $\mathbf{p}_c$ and the feet positions $\mathbf{p}^f$, which only coincide for a point-foot robot with one leg in contact. The footholds affect the input bounds of $\mathbf{u}$. We use $\mathbf{u}$ to control the body, but must at the same time choose appropriate footholds to respect the unilateral forces constraint. Traditional ZMP approaches fix the footholds $\mathbf{p}^f$ in advance, as the combination of both the CoP $\mathbf{p}_c$ and the footholds make this constraint nonlinear. We accept this nonlinearity and the higher numerical complexity associated with it. This gives us a much larger range of inputs $\mathbf{u}$, as we can "customize" our bounds $\mathcal{P}$ by modifying the footholds according to the desired task. Therefore, the first path constraint of our TO problem is given by

$$\mathbf{h}_1(\mathbf{x}(t),\mathbf{u}(t)) \leq 0 \quad \Leftrightarrow \quad \mathbf{p}_c \in \mathcal{P}(\mathbf{p}^f, \alpha^f, c^f), \tag{4.6}$$

where $\mathcal{P}$ represents the convex hull of the feet in contact as seen in Fig. 4.2 and $c^f \in \{0,1\} \in \mathbb{Z}$ is the indicator if foot f is in contact.

We implement this convex hull constraint by weighing the vertices (corners) of each foot in contact. This extends the capabilities of traditional representations by line segments (hyperplanes) to also model point- and line-contacts of arbitrary orientation. We use

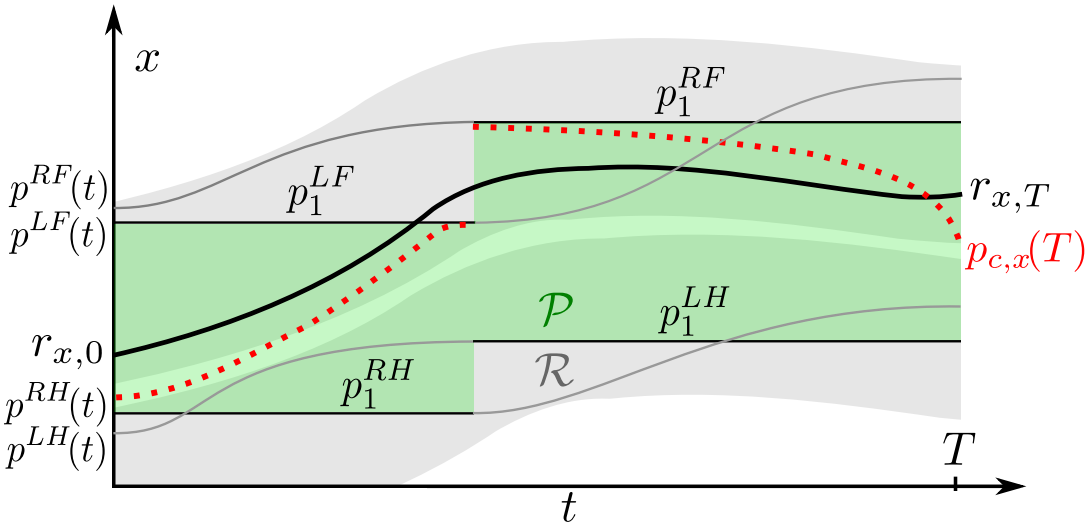


Figure 4.2: *Overview of the TO problem: A point-foot quadruped robot trotting forward in x-direction, first swinging right-front and left-hind legs $f \in \{RF, LH\}$, then left-front and right-hind $f \in \{LF, RH\}$. The CoM motion r_x (black line) is generated by shifting the CoP $p_{c,x}(t)$ (red dots). However, $p_{c,x}$ can only lie in the convex hull $\mathcal{P}$ (green area) of the legs in contact at that time t. Additionally, the position of each leg p^f must always be inside its range of motion $\mathcal{R}$ (gray areas for front and hind legs) relative to the CoM. The optimization problem consist of varying the position of the footholds $p_s^f \in \mathcal{R}$, to allow inputs $p_{c,x} \in \mathcal{P}$ that drive the robot from an initial position $r_{x,0}$ to a desired goal position $r_{x,T}$ in time T.*

predefined contact sequences and timings $c^f(t)$, to only optimize over real-valued decision variables $w \in \mathbb{R}$ and not turn the problem into a mixed-integer NLP. Simply by adapting this contact schedule $c^f(t)$, the optimizer generates various gaits as well as combinations and transitions between these, for which previously separate frameworks were necessary.

4.2.4.2 Kinematic reachability

When modifying the footholds to enclose the CoP, we must additionally ensure that these stay inside the kinematic range $\mathcal{R}$ of the legs (Reachability). This constraint that depends on both the CoM $\mathbf{r}$ and foothold positions $\mathbf{p}^f$ is formulated for every leg f as

$$\mathbf{h}_2(\mathbf{x}(t)) \leq 0 \quad \Leftrightarrow \quad \mathbf{p}^f \in \mathcal{R}(\mathbf{r}). \tag{4.7}$$

Allowing the simultaneous modification of both these quantities characterizes the legged locomotion problem more accurately and reduces heuristics used in hierarchical approaches.

4.2.4.3 Robust motions

With the above constraints, the motion will comply with the physics and the kinematics of the system. This feasible motion is assuming a simplified model, a perfect tracking controller and an accurate initial state. To make solutions robust to real-world discrepancies where these assumptions are violated, it is best to avoid the borders of feasible solutions, where the inequality constraints are tight ($\mathbf{h}=\mathbf{0}$). This can be achieved by artificially shrinking the solutions space by a stability margin (e.g. $\mathbf{h}\leq\mathbf{m}$). For legged locomotion, this is often done by shrinking the support area to avoid solutions were the CoP is placed at the marginally-stable border [18].

We do not restrict the solution space, but choose the more conservative of the feasible motions through a performance criteria J_λ. This soft constraint expresses "avoid boundaries when possible, but permit if necessary". The robot is allowed to be at marginally stable states, but since there are many uncertainties in our model and assumptions, it is safer to avoid them. This cost does not require a hand-tuned stability margin and the solution can still be at the boundaries when necessary. However, especially for slow motions (e.g., walking) where small inaccuracies can accumulate and cause the robot to fall, this cost term is essential to generate robust motions for real systems.

4.3 Implementation

There exist different methods to solve Optimal Control problems (4.2), namely Dynamic Programming (Bellman Optimality Equation), indirect (Maximum Principle) and direct methods [30]. In direct methods, the continuous time TO problem is represented by a finite number of decision variables and constraints and solved by a nonlinear programming solver. If the decision variables $\mathbf{w}$ fully describe the input $\mathbf{u}(t)$ and state $\mathbf{x}(t)$ over time, the method is further classified as a simultaneous direct method, with flavors Direct Transcription and Multiple Shooting. In our approach we chose a Direct Transcription formulation, e.g. optimizing state and controls together. This has the advantage of not requiring an ODE solver, constraints on the state can be directly formulated and the sparse structure of the Jacobian often improves convergence. The resulting discrete formulation to solve the continuous problem in (4.2) is given by

$$
\begin{array}{lll}
\text{find} & \mathbf{w} = (\mathbf{w}_r, \mathbf{w}_p, \mathbf{w}_u) & \\
\text{subject to} & (4.2b), & \text{(given initial state)} \\
& (4.10), (4.12), (4.15), (4.19) & \text{(dyn./path constraints)} \\
& (4.2e), & \text{(desired final state)} \\
& \mathbf{w} = \arg\min(4.21), & \text{(robustness cost)}
\end{array}
$$

where $\mathbf{w}_r$ are the parameters describing the CoM motion, $\mathbf{w}_p$ the feet motion (swing and stance) and $\mathbf{w}_u$ the position of the CoP. This section describes in detail how we parametrize

the state $(\mathbf{w}_r, \mathbf{w}_p)$ and input $\mathbf{w}_u$, formulate the constraints and defined the cost (4.21).

4.3.1 Center of Mass motion

This section explains how the continuous motion of the CoM can be described by a finite number of variables to optimize over, while ensuring compliance with the LIPM dynamics.

4.3.1.1 Center of Mass parametrization

The CoM motion is described by a spline, strung together by n quartic-polynomials as

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{r}(t) \\ \dot{\mathbf{r}}(t) \end{bmatrix} = \sum_{i=1}^{4} \begin{bmatrix} (t-t_k) \\ i \end{bmatrix} \mathbf{a}_{k,i} (t-t_k)^{i-1} + \begin{bmatrix} \mathbf{a}_{k,0} \\ \mathbf{0} \end{bmatrix} \tag{4.8}$$

$$\mathbf{w}_r = \begin{bmatrix} \mathbf{a}_{1,0}, \ldots, \mathbf{a}_{1,4}, \ldots, \mathbf{a}_{n,0}, \ldots, \mathbf{a}_{n,4} \end{bmatrix}, \tag{4.9}$$

with coefficients $\mathbf{a}_{k,i} \in \mathbb{R}^2$ and t_k describing the global time at the start of polynomial k.

We ensure continuity of the spline by imposing equal position and velocity at each of the $n-1$ junctions between polynomial k and $k+1$, so $\mathbf{x}[t_{k+1}^-] = \mathbf{x}[t_{k+1}^+]$. Using $T_k = t_{k+1} - t_k$ we enforce

$$\sum_{i=1}^{4} \begin{bmatrix} T_k \\ i \end{bmatrix} \mathbf{a}_{k,i} T_k^{i-1} + \begin{bmatrix} \mathbf{a}_{k,0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{k+1,0} \\ \mathbf{a}_{k+1,1} \end{bmatrix}. \tag{4.10}$$

4.3.1.2 Dynamic constraint

In order to ensure consistency between the parametrized motion and the dynamics of the system (4.1), the integration of our approximate solution $\ddot{\mathbf{r}}(t)$ must resemble that of the actual system dynamics, so

$$\int_{t_k}^{t_{k+1}} \ddot{\mathbf{r}}(t)\, \mathrm{d}t \approx \int_{t_k}^{t_{k+1}} \mathbf{F}_2(\mathbf{x}(t), \mathbf{u}(t))\, \mathrm{d}t. \tag{4.11}$$

Simpson's rule states that if $\ddot{\mathbf{r}}(t)$ is chosen as a 2^{nd}-order polynomial (which is why $\mathbf{r}(t)$ is chosen as 4^{th}-order) that matches the system dynamics $\mathbf{F}_2$ at the beginning, the center and at the end, then (4.11) is bounded by an error proportional to $(t_{k+1} - t_k)^4$. Therefore we add the following constraints for each polynomial

$$\ddot{\mathbf{r}}[t] = \mathbf{F}_2(\mathbf{x}[t], \mathbf{u}[t]), \quad \forall t \in \left\{ t_k, \tfrac{t_{k+1}-t_k}{2}, t_{k+1} \right\} \tag{4.12}$$

(see Appendix A.4 for a more detailed formulation). By keeping the duration of each polynomial short ($\sim$50 ms), the error of Simpson's integration stays small and the 4^{th}-order polynomial solution $\mathbf{r}(t)$ is close to an actual solution of the ODE in (4.1).

This formulation is similar to the "collocation" constraint [59]. Collocation implicitly enforces the constraints (4.12) at the boundaries through a specific parametrization of the polynomial, while the above formulation achieves this through explicit constraints in the NLP. Reversely, collocation enforces that $\frac{\partial \mathbf{r}(t)}{\partial t} = \dot{\mathbf{r}}(t)$ through the explicit constraint, while our formulation does this through parametrization in (4.8).

4.3.2 Feet motion

4.3.2.1 Feet parametrization

We impose a constant position $\mathbf{p}_s^f \in \mathbb{R}^2$ and orientation $\alpha_s^f \in \mathbb{R}$ if leg f is in stance. We use a cubic polynomial in the ground plane to move the feet between two consecutive contacts

$$\begin{bmatrix} \mathbf{p}^f(t) \\ \alpha^f(t) \end{bmatrix} = \sum_{i=0}^{3} \begin{bmatrix} \mathbf{a}_{s,i}^f \\ b_{s,i}^f \end{bmatrix} (t - t_s)^i, \tag{4.13}$$

where $(t - t_s)$ is the elapsed time since the beginning of the swing motion. The vertical swingleg motion does not affect the NLP and is therefore not modeled. The coefficients $\mathbf{a}_{s,i} \in \mathbb{R}^2$ and $b_{s,i} \in \mathbb{R}$ are fully determined by the predefined swing duration and the position and orientation of the enclosing contacts $\left\{\mathbf{p}_s^f, \alpha_s^f\right\}$ and $\left\{\mathbf{p}_{s+1}^f, \alpha_{s+1}^f\right\}$. Therefore the continuous motion of all n_f feet can be parametrized by the NLP decision variables

$$\begin{aligned} \mathbf{w}_p &= \left[\mathbf{w}_p^1, \ldots, \mathbf{w}_p^{n_f}\right], \\ \text{where } \mathbf{w}_p^f &= \left[\mathbf{p}_1^f, \alpha_1^f, \ldots, \mathbf{p}_{n_s}^f, \alpha_{n_s}^f\right] \end{aligned} \tag{4.14}$$

are the parameters to fully describe the motion of a single leg f taking n_s steps.

4.3.2.2 Range-of-Motion constraint

To ensure a feasible kinematic motion, we must enforce $\mathbf{p}^f \in \mathcal{R}(\mathbf{r})$, which is the gray area in Fig. 4.3. We approximate the area reachable by each foot through a rectangle $[-\mathbf{r}^{xy}, \mathbf{r}^{xy}]$, representing the allowed distance that a foot can move from its nominal position $\mathbf{p}_{nom}^f$ (center of gray area). The foothold position for each foot f is therefore constrained by

$$-\mathbf{r}^{xy} < \mathbf{p}^f[t] - \mathbf{r}[t] - \mathbf{p}_{nom}^f < \mathbf{r}^{xy}. \tag{4.15}$$

Contrary to hierarchical approaches, this constraint allows the optimizer to either move the body to respect kinematic limits *or* place the feet at different positions. A constraint on the foot orientation can be formulated equivalently.

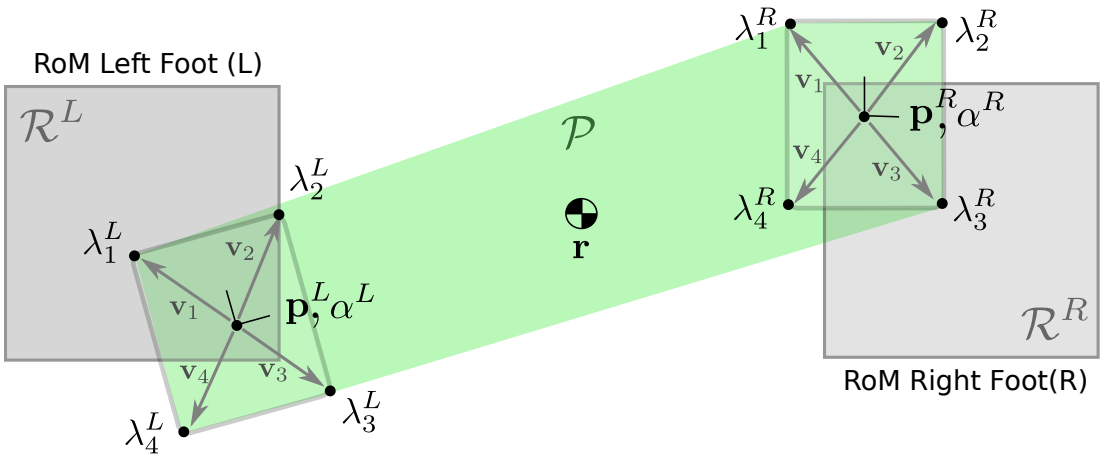


Figure 4.3: *Top-down view of a biped for both feet in contact at* $\mathbf{p}^R, \mathbf{p}^L \in \mathcal{R}$ *inside the range of motion* $\mathcal{R}$ *(gray), which moves with the CoM position. For square feet with corners* $\mathbf{v}_v$*, rotated by* α*, the support area is shown by* $\mathcal{P}$ *(light green area). This is the area to which the CoP* $\mathbf{u}$ *is constrained. If the biped controls its CoP to lie on the tip of the right foot, the corresponding corner carries all the load* ($\lambda_1^R = 1.0$)*, while the other seven lambdas are zero. In case of point-feet the support area is simply a straight line between* $\mathbf{p}^R$ *and* $\mathbf{p}^L$*.*

4.3.3 Center of Pressure motion

To represent the continuous CoP trajectory, we parameterize it through the load carried by each end-effector. This parametrization is used to formulate a novel convexity constraint based on vertices instead of hyperplanes. Finally, this section introduces a cost that keeps the CoP from marginally stable regions and improves the robustness of the motion.

4.3.3.1 Center of Pressure parametrization

The CoP $\mathbf{p}_c(t)$ is not parametrized by polynomial coefficients or discrete points, but by the relative load each corner of each foot is carrying. This load is given by

$$\begin{aligned} \boldsymbol{\lambda}(t) &= \left[\boldsymbol{\lambda}^1(t), \ldots, \boldsymbol{\lambda}^{n_f}(t)\right]^T, \\ \text{where } \boldsymbol{\lambda}^f(t) &= \left[\lambda_1^f(t), \ldots, \lambda_{n_v}^f(t)\right] \in [0,1]^{n_v}. \end{aligned} \tag{4.16}$$

n_v represents the number of vertices/corners of foot f. For the square foot in Fig. 4.3, four lambda values represent one foot and distribute the load amongst the corners. These multipliers represent the percentage of vertical force that each foot is carrying, e.g. $\|\boldsymbol{\lambda}^f(t)\|_1 = 0.9$ implies that leg f is carrying 90% of the weight of the robot at time t. Using these

values, the CoP is parameterized by

$$\mathbf{p}_c(t) = \sum_{f=1}^{n_f} \sum_{v=1}^{n_v} \lambda_v^f(t)(\mathbf{p}^f(t) + \mathbf{R}(\alpha^f(t))\mathbf{v}_v), \tag{4.17}$$

where $\mathbf{R}(\alpha^f) \in \mathbb{R}^{2\times2}$ represents the rotation matrix corresponding to the optimized rotation α^f of foot f (4.13). $\mathbf{v}_v$ represents the fixed position (depending on the foot geometry) of corner v of the foot expressed in the foot frame. For a point-foot robot with $\mathbf{v}_v = \mathbf{0}$, (4.17) simplifies to $\mathbf{p}_c = \sum_{f=1}^{n_f} \lambda^f \mathbf{p}^f$.

We represent $\boldsymbol{\lambda}(t)$ for the duration of the motion by piecewise-constant values $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}(t_i)$ discretized every 20 ms, resulting in n_u nodes. Therefore the CoP $\mathbf{p}_c$ can be fully parameterized by $\mathbf{w}_p$ and the additional NLP decision variables

$$\mathbf{w}_u = \left[\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{n_u}\right]. \tag{4.18}$$

4.3.3.2 Unilateral forces constraint

We represent the essential input constraint (4.6), which ensures that only physically feasible forces inside the convex hull of the contacts are generated, for $i = 1, \ldots, n_u$ as

$$\|\boldsymbol{\lambda}_i\|_1 = 1, \tag{4.19a}$$

$$0 \leq \lambda_v^f[t_i] \leq c^f[t_i], \tag{4.19b}$$

where $c^f \in \{0, 1\} \in \mathbb{Z}$ is the indicator if foot f is in contact. The constraints (4.17) and (4.19a) allow $\mathbf{p}_c$ to be located anywhere inside the convex hull of the vertices of the current foot positions, independent of whether they are in contact. However, since only feet in contact can carry load, (4.19b) enforces that a leg that is swinging ($c^f = 0$) must have all the corners of its foot unloaded. These constraints together ensure that the CoP lies inside the green area shown in Fig. 4.3.

4.3.3.3 Robust walking cost

To keep the CoP away from the edges of the support-area we could constrain λ_v^f of each leg in stance to be greater than a threshold, causing these legs in contact to never be unloaded. This conceptually corresponds to previous approaches that heuristically shrink support areas and thereby reduce the solution-space for *all* situations. We propose a cost that has a similar effect but still permits the solver to use the limits of the space if necessary.

The most robust state to be in, is when the weight of the robot is equally distributed

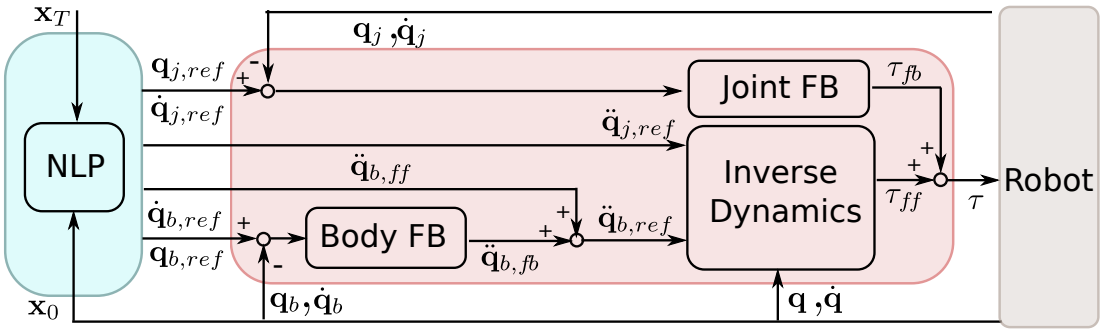


Figure 4.4: *The controller that generates the required torques to execute a planned motion. Given the current state of the system* $\mathbf{x}_0$ *and a user-defined goal state* $\mathbf{x}_T$*, the optimizer generates a reference motion. We augment this reference through a body feedback acceleration based on how much the body deviates from the desired motion. Inverse dynamics is used to generate the torques to achieve the reference base and joint accelerations.*

amongst all the corners in contact, so

$$\lambda_v^{f*}(t) = \frac{c^f(t)}{n_v(t)}, \tag{4.20}$$

where $n_v(t) = n_v \sum_{f=1}^{n_f} c^f(t)$ is the total number of vertices in contact at time t, predefined by the contact sequence $c(t)$. This results in the CoP to be located in the center of the support areas. The deviation of the input values from the optimal values λ^* over the entire discretized trajectory (4.18) is then given by

$$J_\lambda(\mathbf{w}_u) = \sum_{i=1}^{n_u} \|\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_i^*\|_2^2 . \tag{4.21}$$

For a support triangle ($\lambda_v^{f*} = \frac{1}{3}$) this cost tries to keep the CoP in the center and for a line ($\lambda_v^{f*} = \frac{1}{2}$) in the middle. For quadruped walking motions this formulation generates a smooth transition of the CoP between diagonally opposite swing-legs, while still staying away from the edges of support-areas whenever possible.

4.4 Tracking the motion

The motion optimization part of our approach is mostly robot independent. The only robot specific information needed to run the framework is the robot height, the number of feet, their geometry, and their kinematic range. For execution, however, the optimized motion must be translated into joint torques $\boldsymbol{\tau}$ using a fully-body dynamics model. This section discusses this generation summarized by Fig. 4.4.

4.4.1 Generating full-body reference accelerations

The 6–DoF base pose is reconstructed using zero desired orientation (in Euler angles x,y,z), the optimized CoM motion $\mathbf{r}$ (assuming the geometric center of the base coincides with the CoM), and the constant base height h as

$$\mathbf{q}_{b,ref}(t) = \begin{bmatrix} 0 & 0 & 0 & r_x(t) & r_y(t) & h \end{bmatrix}^T.$$

In order to cope with uncertainties it is essential to incorporate feedback into the control loop. We do this by adding an operational space PD-controller on the base that creates desired 6D base accelerations according to

$$\ddot{\mathbf{q}}_{b,ref} = \ddot{\mathbf{q}}_{b,ff} + K_p(\mathbf{q}_b - \mathbf{q}_{b,ref}) + K_d(\dot{\mathbf{q}}_b - \dot{\mathbf{q}}_{b,ref}).$$

The derivate of the pose, the base twist $\dot{\mathbf{q}}_b \in \mathbb{R}^6$ represents the base angular and linear velocities and $\ddot{\mathbf{q}}_{b,ff}$ is the optimized CoM acceleration from the NLP. This controller modifies the planned body motion if the current state deviation from the reference state.

In order to obtain the desired joint accelerations that correspond to the planned Cartesian motion of the feet we can use the relationship $\ddot{\mathbf{p}}(t) = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$, where $\dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^{6+n}$ represent the full body state (base + joints) and $\mathbf{J} = \begin{bmatrix} \mathbf{J}_b & \mathbf{J}_j \end{bmatrix} \in \mathbb{R}^{3n_f \times (6+n)}$ the Jacobian that maps full-body velocities to linear foot velocities in world frame. Rearranging this equation, and using the Moore–Penrose pseudoinverse $\mathbf{J}_j^+$, gives us the reference joint acceleration

$$\ddot{\mathbf{q}}_{j,ref} = \mathbf{J}_j^+ \left(\ddot{\mathbf{p}} - \dot{\mathbf{J}}\dot{\mathbf{q}} - \mathbf{J}_b \ddot{\mathbf{q}}_{b,ref} \right). \tag{4.22}$$

4.4.2 Inverse Dynamics

The inverse dynamics controller is responsible for generating required joint torques $\boldsymbol{\tau}$ to track the reference acceleration $\ddot{\mathbf{q}}_{ref}$, which is physically feasible based on the LIPM model. This is done based on the rigid body dynamics model of the system, which depends on the joint torques, but also the unknown contact forces. To eliminate the contact forces from the equation, we project it into the space of joint torques by $\mathbf{P} = \mathbf{I} - \mathbf{J}_c^+ \mathbf{J}_c$, where $\mathbf{J}_c^T$ is the contact Jacobian that maps Cartesian contact forces to joint torques [58], [81]. This allows us to solve for the required joint torques through

$$\boldsymbol{\tau} = (\mathbf{P}\mathbf{S}^T)^+ \, \mathbf{P}(\mathbf{M}\ddot{\mathbf{q}}_{ref} + \mathbf{C}), \tag{4.23}$$

where $\mathbf{M}$ is the joint space inertia matrix, $\mathbf{C}$ the effect of Coriolis forces on the joint torques and $\mathbf{S}$ the selection matrix which prohibits from actuating the floating base state directly. We found it beneficial to also add a low-gain PD-controller on the joint position and velocities. This can mitigate the effects of dynamic modeling errors and force tracking imperfections.

4.5 Results

We demonstrate the performance of this approach on the hydraulically actuated quadruped robot HyQ [5]. The robot weighs approximately 80 kg, moves at a height of about 0.6 m and is torque controlled. Base estimation [83] is performed on-board, fusing Inertial Measurement Unit (IMU) and joint encoder values. Torque tracking is performed at 1000 Hz, while the reference position, velocity and torque set-points are provided at 250 Hz. The C++ dynamics model is generated by [84].

4.5.1 Discussion of generated motions

This section analyses the different motions generated by changing the sequence and timings of contacts $c(t)$. There is no high-level footstep planner; the footholds are chosen by the optimizer to enable the body to reach a user-defined goal state $\mathbf{x}_T$. The results were obtained using C++ code interfaced with Interior Point Method (Ipopt [35]) or Sequential Quadratic Programming (Snopt [36]) solvers on an Intel Core i7/2.8 GHz Quadcore laptop. The Jacobians of the constraint and the gradient of the cost function are provided to the solver analytically, which is essential for performance. We initialize the decision variables $\mathbf{w}$ with the quadruped standing in default stance for a given duration. The shown motions correspond to the first columns (e.g. 16 steps) in Table 4.1. The reader is encouraged to view the video[1], as it very intuitively demonstrates the performance of this approach. Apart from the basic gaits, the video shows the capability of the framework to generate gradual transitions between them, bipedal walking, limping and push-recovery.

4.5.1.1 Walk

Fig. 4.5(a) shows a walk of multiple steps, with the two support areas highlighted for swinging RF→LH. The effect of the cost term J_λ is visible, as the CoP is accumulated away from the support area borders by left-right swaying of the body. Only when switching diagonally opposite legs the CoP lies briefly at the marginally stable border, but then immediately shifts to a more conservative location. Without the cost term, the CoM motion is a straight line between $\mathbf{x}_0$ and $\mathbf{x}_T$, causing the real system to fail.

4.5.1.2 Trot

Fig. 4.5(b) shows a completely different pattern of support areas and CoP distribution. During trotting only line-contacts exist, so the possible places to generate the CoP is extremely restricted compared to walking. Notice how the CoP lies close to the CoM trajectory during the middle of the motion, but deviates quite large back/forward during the start/end of the motion (e.g., the robot pushing off from the right-front (green) leg

[1] Video of generated motions: https://youtu.be/5WLeQMBuv30.

Table 4.1: *Specs of the NLP for 16- and 4-step motions*

	(16 steps, 1 m) | (4 steps, 0.2 m)			
	Walk	Trot	Pace	Bound
Horizon T [s]	6.4 | 1.6	2.4 | 0.6	3.2 | 0.8	3.2 | 0.8
Variables [-]	646 | 202	387 | 162	1868 | 728	1868 | 728
Constraints [-]	850 | 270	548 | 255	2331 | 939	2331 | 939
$t_{k+1} - t_k$ [s]	0.1	0.05	0.02	0.02
Cost term	J_λ	-	-	-
Time Ipopt [s]	0.25 | 0.06	0.02 | 0.01	0.21 | 0.12	0.17 | 0.04
Time Snopt [s]	0.35 | 0.04	0.04 | 0.01	0.54 | 0.18	0.42 | 0.29

in the second to last step). This is because the distance between the CoP and the CoM generates the acceleration necessary for starting and stopping, whereas in the middle the robot is moving with nearly constant velocity.

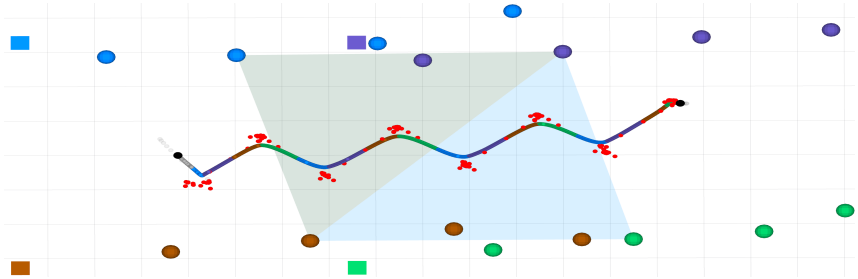
4.5.1.3 Pace/Bound/Biped Walk

Specifying legs on the same side to be in contact, with a short four-leg transition period between them produces the motion shown in Fig. 4.5(c). This can also be viewed as a biped walking with line-feet (e.g., skis), with the constraint enforced also *during* the double-stance phase. The first observation is the sideways swaying motion of the CoM. This is necessary because the support areas do not intersect (as in the trot) the CoM trajectory. Since the CoP always lies inside these left and right support areas, they will accelerate the body away from that side until the next step, which then reverses the motion. We found that the LIPM model with fixed zero body orientation does not describe such a motion very well, as the inherent rotation (rolling) of the body is not taken into account. To also demonstrate these motions on hardware, the LIPM model must be extended by the angular body motion. Specifying the front and hind legs to alternate between contact generates a bound Fig. 4.5(d). The lateral shifting motion of the pace is now transformed to a forward-backward motion of the CoM due to support areas. In case of an omnidirectional robot, a bounding gait can merely be considered a side-ways pace.

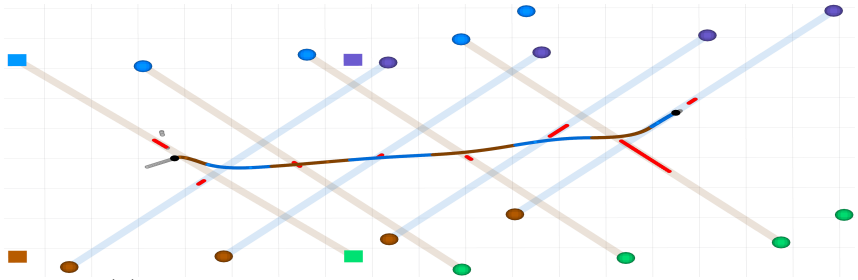
4.6 Conclusion

This paper presented a TO formulation using vertex-based support-area constraints, which enables the generation of a variety of motions for which previously separate methods were necessary. In the future, more decision variables (e.g., contact schedule, body orientation, foothold height for uneven terrain), constraints (e.g., friction cone, obstacles) and more

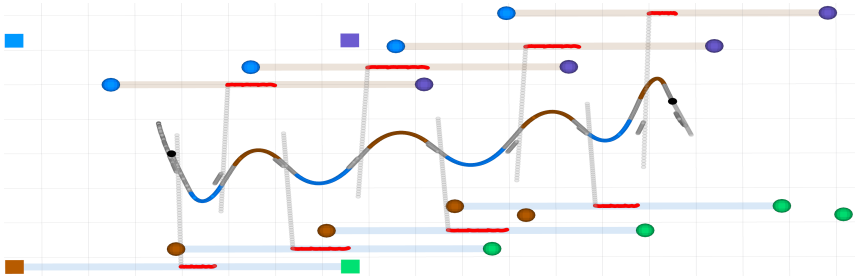
sophisticated dynamic models can be incorporated into this formulation. Additionally, we plan to utilize the speed of the optimization for MPC.



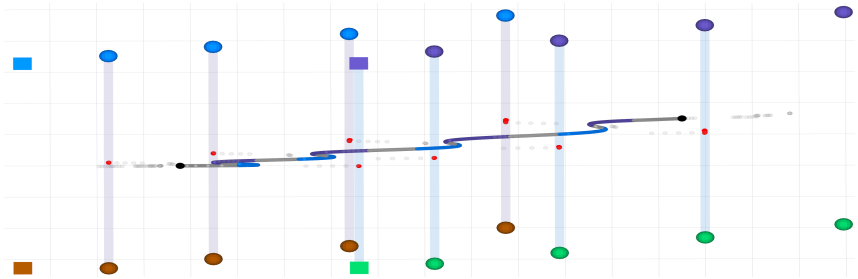
(a) Quadruped Walk: swinging one leg at a time.



(b) Quadruped Trot: swinging diagonally opposite legs.



(c) Quadruped Pace: swinging left, then right legs.



(d) Quadruped Bound: swinging front, then hind legs.

Figure 4.5: *Top-down view of the generated motions for a quadruped robot moving from left to right, swinging the legs f left-hind (blue), left-front (purple), right-hind (brown), right-front (green) in the sequence shown. The initial stance is shown by the squares, the optimized steps by the circles. The CoM motion* $\mathbf{r}(t)$ *is shown by the solid line, where the color corresponds to the swingleg(s) at that moment. If all legs are in contact, the CoM motion and corresponding CoP are shown in gray. The support area for each phase is shown by the transparent areas. The optimized CoP positions* $\mathbf{p}_c(t)$ *that drive the system are shown in red and always lie inside the support area.*

5

Paper III: Gait and trajectory optimization

Alexander W. Winkler, Farbod Farshidian, Diego Pardo, Michael Neunert, Jonas Buchli. *Fast Trajectory Optimization for Legged Robots using Vertex-based ZMP Constraints.* In Robotics and Automation Letters (RA-L), pp. 2201–2208, 2017.

Abstract We present a single Trajectory Optimization formulation for legged locomotion that automatically determines the gait-sequence, step-timings, footholds, swing-leg motions and 6D body motion over non-flat terrain, without any additional modules. Our phase-based parameterization of feet motion and forces allows optimizing over the discrete gait sequence using only continuous decision variables. The system is represented using a Single Rigid Body Dynamics (SRBD) model that is influenced by the feet's location and forces. We explicitly enforce friction cone constraints, depending on the shape of the terrain. The NLP solver generates highly dynamic motion-plans with full flight-phases for a variety of legged systems with arbitrary morphologies in an efficient manner. We validate the feasibility of the generated plans in simulation and on the quadruped robot ANYmal. Additionally, the entire software Towr used to generate these motions is made freely available.[1]

Paper: https://doi.org/10.1109/LRA.2018.2798285
Video: https://youtu.be/0jE46GqzxMM
Code: https://github.com/ethz-adrl/towr/tree/1.1.0

[1]The dynamic model described in Section 5.3.2 has been slightly adapted to conform to the terms, e.g. SRBD, used in this thesis.

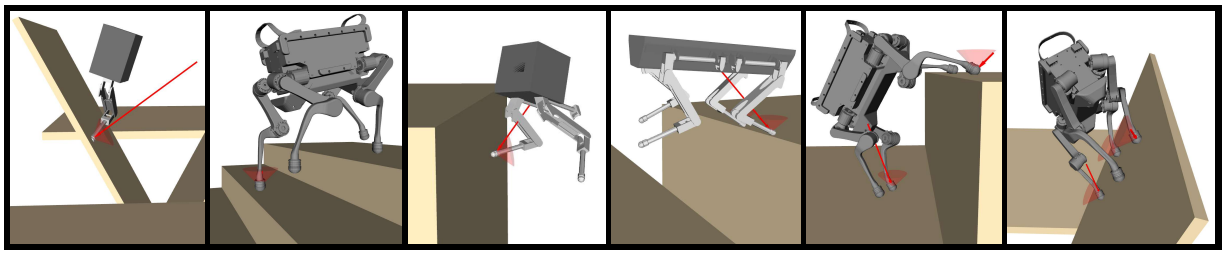


Figure 5.1: *Motions produced by the solver* TOWR *[72] for single-legged hoppers, bipeds and quadrupeds seen in video at* https://youtu.be/0jE46GqzxMM.

5.1 Introduction

Planning physically feasible motions for legged systems is difficult. A core difficulty is that base movement cannot be directly generated but results from contact of the feet with the environment. Therefore, the generated forces acting at these contact points must be carefully planned to achieve a desired behavior. Unfortunately, there are substantial restrictions on these forces, e.g. a force can only be generated if the foot is touching the environment or feet can only push into the ground, not pull on it.

Due to the complexity of these restrictions, hand-crafting valid trajectories for all these interdependent quantities (body, feet, forces) is tedious. Instead, TO [31] can be used to generate motions in a more general, automated way. The user specifies only the high-level task, while the optimizer determines the motions and forces given these locomotion specific restrictions. The research problem is how to transcribe this continuous-time optimization problem into one with a finite number of decision variables and constraints to be solvable by a NLP solver. This approach is attractive, because once the problem has been appropriately modeled, the program would, in an ideal case, produce motions for *any* high-level task, solving legged locomotion planning on a general level.

5.1.1 Related Work

In the following, we categorize existing approaches to legged locomotion by their used physical model and by which aspects of the motion (e.g., body height and orientation, step sequence, timings) are fixed in advance and which are determined by an optimizer.

5.1.1.1 Dynamic Models

There exist a wide variety of approaches using the LIPM model, that optimize only over the CoM position, while using predefined footholds and step timings. By modeling the robot as an inverted pendulum, the position of the CoP, or Zero Moment Point (ZMP) [79], can be used as a substitute for the contact forces and is used to control the motion of the CoM. This fast and efficient approach has been successfully applied in bipedal and

quadrupedal locomotion on real hardware [20], [22], [23], [78]. This demonstrates that even such drastic model simplifications can be valid and useful. However, imposing *where* these contact forces will be acting (by predefining the footholds) strongly restricts the possible base motions. A slight relaxation is to still define *when* each foot is in contact, but allow the algorithm to determine the best location for the foothold. Together with a simplified model, this results in a very fast solver that can also be used online [38], [39]. It is also possible to adapt step timings or foothold locations for robust real robot execution. Many other variations of using these simplified models to generate impressive results have been shown by [19], [44], [61], [62], [65], [80], [88], [90].

To generate more complex motions, involving vertical movement and changes in body orientation, a more sophisticated dynamics model becomes necessary. The (6+n)-dimensional full rigid-body dynamics consider the mass and inertia of each link and defines the relation between joint torques and base- and joint-accelerations. This model takes into account changing CoM positions and inertia properties based on leg configurations and Coriolis forces generated by leg motions and has been used by [55], [60], [64], [73].

Another common dynamic model used in TO is the Centroidal dynamics [10], which projects the effects of all link motions onto the 6-dimensional base. The idea is that once a physically correct motion for the unactuated base has been found, the leg torques can be readily calculated using standard, fully-actuated inverse dynamics. The input that drives this system is the contact forces, as opposed to the CoP in the LIPM or the joint torques in the full rigid-body dynamics model. Variations of this model have been successfully used by [8], [9], [46], [49] to optimize for a wide variety of dynamic motions for biped robots, including demonstrations on real bipeds.

Common to all these approaches is that some part of the motion is specified beforehand. The different levels include specifying (i) only order of feet in contact (ii) order and times when each foot is in contact (iii) order, times and position of each foot in contact. This decoupling can increase optimization speed; however, it often introduces handcrafted heuristics to link these separated problems. These can become hard to tune for more complex problems and often limit the range of achievable motions. The following discusses approaches how (i)–(iii) can be automatically determined.

5.1.1.2 Contact Schedule Optimization

Before searching for the optimal forces at a given time, it is necessary to know if a foot is touching the environment and therefore can even exert *any* force. This reasoning about which of the feet should be in contact at a given time, or stated differently, when it is necessary to generate forces at which foot is the problem of finding a "contact schedule".

Since feet are modeled as discrete variables $\mathbb{Z}$ (e.g. left-foot, right-foot), Integer Programming can be used to optimize the contact-schedule and footholds, independent from the dynamics of the system [49], [91], [92]. While this decoupling increases speed and allows quick solutions for each separate problem, it also necessitates heuristics that limit

the range of achievable motions. Additionally, Integer Programming becomes computationally expensive as the number of decision variables increases. Another conventional approach is to use a soft-contact model, which approximates the inherently hard contact surfaces as spring-damper systems [67]. The downside to this approach is that these virtual spring-damper models must be very stiff to most accurately resemble the real surface. These abrupt changes in force from a foot hitting this stiff surface hinder convergence of the optimizer. To avoid this, the problem can also be solved by formulating a LCP, which enforces that either the foot is zero distance from the contact surface (touching the environment), or the force is zero [7], [69], [70]. These approaches produced impressive results and are closest to the work presented in this paper.

5.1.2 Contributions

We extend the above works with a *single* TO formulation for legged locomotion that automatically determines the gait-sequence, step timings, footholds, swing-leg motions, 6D body motion and required contact forces over non-flat and inclined terrain. No prior footstep planning is necessary since our formulation directly generates the complete motion given only a desired goal position and the number of steps. We directly enforce friction constraints, which allow the algorithm to use inclined surfaces in physically feasible ways to complete desired tasks. Our algorithm extends existing algorithms that have the above capabilities in the following ways:

1. We generate motions for multiple steps in only a few seconds while still optimizing over the gait sequence. This is due to our novel phase-based parameterization of the feet and forces that keep the optimization variables continuous, and thereby the problem solvable by an NLP solver.

2. Our NLP formulation can automatically generate motions with full-flight phases, which are essential for highly dynamics motions.

5.2 Trajectory optimization formulation

The complete TO formulation presented in this paper can be seen in Fig. 5.2. The initial and desired final state of the system, the total duration T and the amount of steps $n_{s,i}$ per foot i is provided. With this information the algorithm finds a trajectory for the linear CoM position $\mathbf{r}(t)$, its orientation $\boldsymbol{\theta}(t)$, the feet motion $\mathbf{p}_i(t)$ and the contact force $\mathbf{f}_i(t)$ for each foot, while automatically discovering an appropriate gait pattern defined by $\Delta T_{i,j}$.

To ensure physically correct behavior, a simplified Centroidal dynamics model is used that relates feet position and forces with the CoM motion. Additionally, kinematic restriction between base and foot position are enforced in Cartesian space. This robot model is explained in Section 5.3.

find	$\mathbf{r}(t) \in \mathbb{R}^3$	(CoM linear position)
	$\boldsymbol{\theta}(t) \in \mathbb{R}^3$	(base Euler angles)
	for every foot i :	
	$\Delta T_{i,1} \ldots, \Delta T_{i,2n_{s,i}} \in \mathbb{R}$	(phase durations)
	$\mathbf{p}_i(t, \Delta T_{i,1}, \ldots) \in \mathbb{R}^3$	(foot position)
	$\mathbf{f}_i(t, \Delta T_{i,1}, \ldots) \in \mathbb{R}^3$	(force at foot)
s.t.	$[\mathbf{r}, \boldsymbol{\theta}](t{=}0) = [\mathbf{r}_0, \boldsymbol{\theta}_0]$	(initial state)
	$\mathbf{r}(t{=}T) = \mathbf{r}_g$	(desired goal)
	$[\ddot{\mathbf{r}}, \dot{\boldsymbol{\omega}}]^T = \mathbf{F}(\mathbf{r}, \mathbf{p}_1, \ldots, \mathbf{f}_1, \ldots)$	(dynamic model)
	for every foot i :	
	$\mathbf{p}_i(t) \in \mathcal{R}_i(\mathbf{r}, \boldsymbol{\theta})$,	(kinematic model)
	if foot i in contact :	
	$\dot{\mathbf{p}}_i(t \in \mathcal{C}_i) = \mathbf{0}$	(no slip)
	$p_i^z(t \in \mathcal{C}_i) = h_{terrain}(\mathbf{p}_i^{xy})$	(terrain height)
	$\mathbf{f}_i(t \in \mathcal{C}_i) \cdot \mathbf{n}(\mathbf{p}_i^{xy}) \geq 0$	(pushing force)
	$\mathbf{f}_i(t \in \mathcal{C}_i) \in \mathcal{F}(\mu, \mathbf{n}, \mathbf{p}_i^{xy})$	(friction cone)
	if foot i in air :	
	$\mathbf{f}_i(t \notin \mathcal{C}_i) = \mathbf{0}$	(no force in air)
	$\sum_{j=1}^{2n_{s,i}} \Delta T_{i,j} = T$	(total duration)

Figure 5.2: *Decision variables and constraints defining the legged locomotion TO problem. The brown quantities show those aspects of the formulation that are related to environmental contact (Section 5.4), while the other rows apply in general to floating base systems.*

Independent from the base motion and dynamics, there are various restrictions on feet motions and forces described in Section 5.4. To ensure that feet do not slip and forces are produced only when touching the terrain, additional constraints are necessary. We introduce a novel parameterization of the variables based on each foot's swing and stance durations $\Delta T_{i,j}$, which allows to automatically determine the gait sequence and timings.

The presented NLP formulation allows generating dynamic motions with full flight-phases for systems with various numbers of feet in just a few seconds. It can handle non-flat terrain, e.g., walking over stairs and jumping over gaps. In Section 5.5 we analyze selected motions, as well as validate the feasibility of the generated motion plans on a real quadruped.

5.2.1 Parameterization of optimization quantities

The 6D base motion is represented by the linear CoM position $\mathbf{r}(t) \in \mathbb{R}^3$, while the orientation is parameterized by Euler angles $\boldsymbol{\theta}(t) \in \mathbb{R}^3$. We use fourth-order polynomials of fixed durations strung together to create a continuous spline and optimize over the polynomial coefficients. For each foot's motion $\mathbf{p}_i(t) \in \mathbb{R}^3$, we use multiple third-order polynomials per swing-phase, and a constant value $\in \mathbb{R}^3$ for the stance phase. For each foot's force profile $\mathbf{f}_i(t) \in \mathbb{R}^3$, multiple polynomials represent each stance phase, and zero force is set during swing-phase. The duration of each phase, and with that the duration of each foot's polynomial, is changed based on the optimized phase durations $\Delta T_{i,j} \in \mathbb{R}$. Since the decision variables fully describe both the input (forces) and the state evolution (base and feet motion), the optimization can be considered a "simultaneous direct" method (as, e.g., Collocation) [30].

5.3 Robot model

5.3.1 Kinematic model

Instead of directly constraining joint angles, as is done in full-body joint-space TO, we consider how the joint limits constrain the Cartesian foot position. We approximate each foot's workspace by a cube of edge length $2\mathbf{b}$ (see Fig. 5.3), centered at the nominal position of each foot $\overline{\mathbf{p}}_i$ relative to the CoM. We assume the joint limits are not violated if every foot i lies inside a cube, given by

$$\begin{aligned} & \mathbf{p}_i(t) \in \mathcal{R}_i(\mathbf{r}, \boldsymbol{\theta}) \\ \Leftrightarrow \quad & |\mathbf{R}(\boldsymbol{\theta})[\mathbf{p}_i(t) - \mathbf{r}(t)] - \overline{\mathbf{p}}_i| < \mathbf{b}, \end{aligned} \tag{5.1}$$

where $\mathbf{R}(\boldsymbol{\theta})$ is the rotation matrix from the world frame to the base frame. This condition is enforced at regularly sampled states along the trajectory.

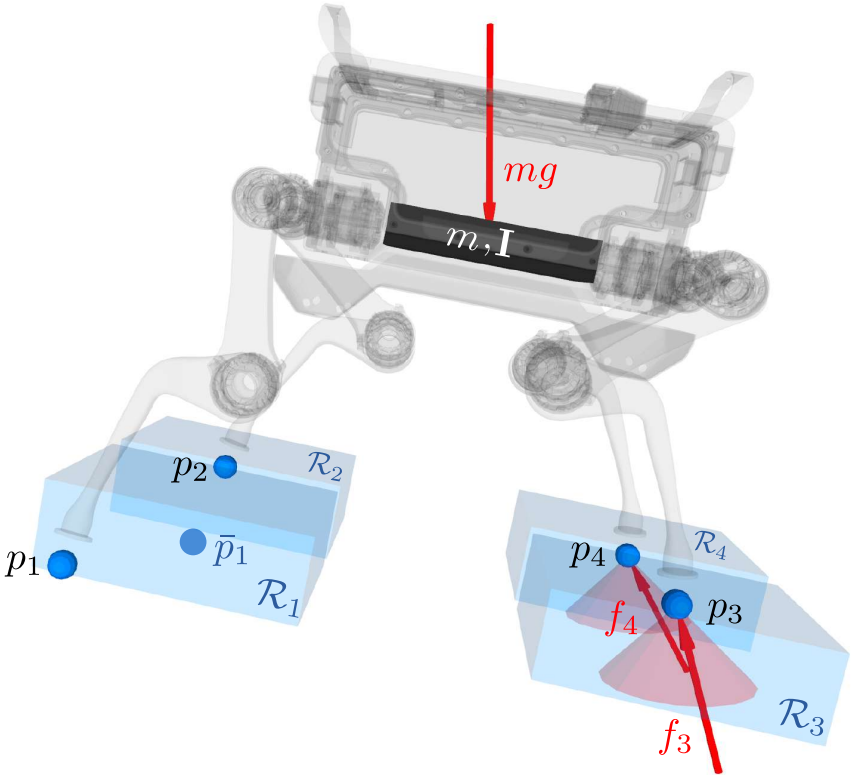


Figure 5.3: *The robot model known by the optimizer. The robot kinematic model is conservatively approximated by keeping the respective foot* $\mathbf{p}_i$ *inside the range of motion* $\mathcal{R}_i$ *of each foot. The dynamics are approximated by a single rigid-body with mass* m *and inertia* $\mathbf{I}$ *located at the robots CoM (Centroidal dynamics). This can be controlled by the contact forces* $\mathbf{f}_i$ *of the feet in contact with the environment while keeping these forces inside the friction cone. The gray overlayed ANYmal model [11] is only for visualization and not known by the optimizer.*

5.3.2 Dynamic model

If we restrict the base orientation to be fixed, $\dot{\boldsymbol{\theta}}(t) = 0$, as well as the walking height to remain constant, $\mathbf{r}_z(t) = h$, it is possible to represent the system as a LIPM, driven by the CoP $\mathbf{p}_c$ as $\ddot{\mathbf{r}}_{xy} = (\mathbf{r}_{xy} - \mathbf{p}_c)gh^{-1}$. However, keeping the base at a constant height and orientation restricts the range of achievable motions, especially on rough terrain where some footholds can only be reached when also tilting the base. Additionally, by replacing the effect of individual contact forces with a single CoP, important information is lost, e.g., keeping each individual force inside the corresponding friction cone cannot be enforced anymore. Finally, situations with all feet in the air cannot be represented with this model, as a CoP $\mathbf{p}_c$ must always exist. For the above reasons, we decide this model is not expressive enough to represent the motions we wish to generate.

Another possibility is using the accurate joint-space RBD (Section 1.2.1), applying joints

torques as input. However, the main difficulty of legged locomotion remains in finding a physically feasible motion for the under-actuated base given the locomotion specific restrictions seen in Fig. 5.2. These physical constraints can be most intuitively expressed in Cartesian-, not joint-space, as they relate to the environment. Once a physically feasible motion for the base has been found the joint torques can easily be obtained using inverse dynamics.

We choose to remain in Cartesian space since this allows to formulate the relevant variables and constraints in a more simple way and model the legged robot using Single Rigid Body Dynamics (SRBD) (see Fig. 5.3). The CoM linear $\ddot{\mathbf{r}}$ and angular $\dot{\boldsymbol{\omega}}$ acceleration are determined by

$$\begin{aligned} m\ddot{\mathbf{r}}(t) &= \sum_{i=1}^{n_i} \mathbf{f}_i(t) - m\mathbf{g} \\ \mathbf{I}\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}(t) \times \mathbf{I}\boldsymbol{\omega}(t) &= \sum_{i=1}^{n_i} \mathbf{f}_i(t) \times (\mathbf{r}(t) - \mathbf{p}_i(t)), \end{aligned} \tag{5.2}$$

where m is the mass of the robot, n_i the number of feet, $\mathbf{g}$ is the gravity acceleration and $\boldsymbol{\omega}(t)$ represents the angular velocity that can be calculated from the optimized Euler angles $\boldsymbol{\theta}(t)$ and rates $\dot{\boldsymbol{\theta}}(t)$ (see Appendix A.6). We use a combined rotational inertia $\mathbf{I} \in \mathbb{R}^{3\times3}$ calculated for the robot in nominal joint configuration. This assumes that either the limb masses are negligible compared to the torso or that the limbs do not move significantly or quickly from their default pose. These assumptions make the dynamics of the robot independent of the joint configuration and express them solely in Cartesian space. For the presented robots and motions the above assumptions introduce only negligible modeling error while keeping the formulation simpler and the solver fast. For more information on this model, see Section 1.2.3.

To ensure physical behavior of the motion, we enforce (5.2) at regular time intervals along the trajectory. Additionally, we constrain the acceleration at the junction between two base polynomials to be equal, as jumps in acceleration would imply jumps in force or foot position, which we do not allow.

5.3.3 Contact independent dynamic model

Until now, the dynamic model (5.2) can just as well represent a flying drone. What makes it specific to legged systems is that the restriction on the forces and feet positions abruptly change depending on whether a foot is in contact with the environment or not. These discretely switching contact configuration and therefore discretely switching constraints are difficult to handle. For example, NLP formulations don't naturally allow constraints to merely be turned on or off arbitrarily during the iterations. This is why the sequence and duration of contacts are often specified in advance when using an NLP to solve the legged locomotion problem.

To partially simplify the problem, the robot model can be viewed independently from concepts such as contacts or phases and the discontinuities can be handled where they

actually occur – in the individual foot motion and forces. As a consequence of treating every foot separately, concepts that described multiple feet at once, such as *phases* or *contact configurations* can be simplified to binary *in contact* or *not*.

The following section does not include any robot dependent quantities (kinematic, dynamic) anymore, nor is it dependent on the base motion. From now on each foot is treated separately and is only affected by the terrain and the physical constraints coming from non-slip, frictional contact of rigid bodies.

5.4 Contact model

In this section, we first explain how arbitrary gaits can be generated by modifying the duration of each individual foot's swing and stance phase. We then describe how we exploit this knowledge to formulate an NLP with continuous optimization variables, that is still able to optimize over the gait sequence. Finally, we describe how we model the physical constraints between the terrain and the foot motion and forces.

5.4.1 Contact schedule optimization

A biped walk can be characterized by phases (L, R, D, F) as seen in Fig. 5.4. If the number of possible phases is low, these can possibly be predefined in a sensible, intuitive way for a given task. However, as the number of contact points increases (e.g., quadruped, bipeds with hands, allowing other body parts in contact), it becomes highly complicated to determine which of feet $\in \mathbb{Z}$ should make contact in what order to achieve a desired task.

However, we can observe that more than two phases only exist when viewing multiple feet simultaneously. When looking at a single-legged hopper, there exist *exactly* two phases – a contact phase $\mathcal{C}$ and a flight phase. Furthermore, these two phases always alternate: After the foot is in contact, it will be in a flight phase, then again in contact, etc.

Analogously, we can view multi-legged robots as having independent feet, each alternating between contact and flight. What varies to generate the different gaits are the durations of each foot's swing and stance phase. Figure 5.4 shows that solely by changing the phase durations $\Delta T_{i,j} \in \mathbb{R}$ of the right foot, a completely different gait can be generated. Since the phase durations are continuous, these can be readily optimized by NLP solvers and Integer Programming can be avoided.

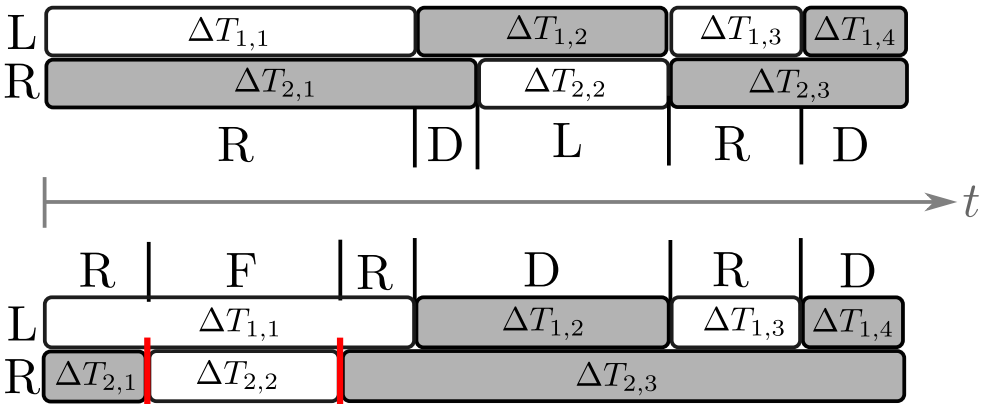


Figure 5.4: *Two different contact schedules for a bipedal robot. L = left foot in contact, R = right foot in contact, D = both feet standing, F = flight phase. The change of gait is achieved solely by adapting the stance durations (red delimiters) of the right foot.*

5.4.2 Feet motion and forces parameterization

To exploit this regularity, each dimension of the quantities $\mathbf{p}_i(t), \mathbf{f}_i(t)$ is described by alternating sequences of constant values and cubic polynomials

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \quad a_i = f(\Delta T_{,}, x_0, \dot{x}_0, x_1, \dot{x}_1) \tag{5.3}$$

as shown in Fig. 5.5. Instead of optimizing over polynomial coefficients, we use the value x and derivative $\dot{x}$ at the end-points ("nodes") and its duration ΔT to fully define each polynomial (see Appendix A.5). This so-called "Hermite" parameterization is more intuitive, since the optimization variables directly describe the state. Furthermore, the node used as the end of the previous polynomial can also be used as the starting node of the next, which ensures continuous foot velocity and force changes over the trajectory.

In Fig. 5.5 we use three polynomials of equal duration $\Delta T_{i,j}/3$ to represent each swing phase of the foot motion and each stance phase of the foot force. These can represent typically varying force and motion profiles while still keeping the problem as small as possible. The other phases are represented by a constant value for a duration of $\Delta T_{i,j}$. We predefine the maximum number of steps $n_{s,i}$ each foot can take. Note that this is not a strong restriction, as phase durations can always be set close to zero if fewer steps are required.

The times at which foot i is in contact for the s^{th} time are given by

$$\mathcal{C}_{i,s} = \left\{ t \,\middle|\, 0 < t - \textstyle\sum_{j=1}^{2s-1} \Delta T_{i,j} < \Delta T_{i,2s} \right\}. \tag{5.4}$$

This uses the intuition that every new foothold s is preceded by exactly two phases j (swing and stance). The set of all times that foot i is in contact is denoted by $\mathcal{C}_i = \cup_{s=1}^{n_{s,s}} \mathcal{C}_{i,s}$.

The individual polynomials carry information about whether they represent a swing or

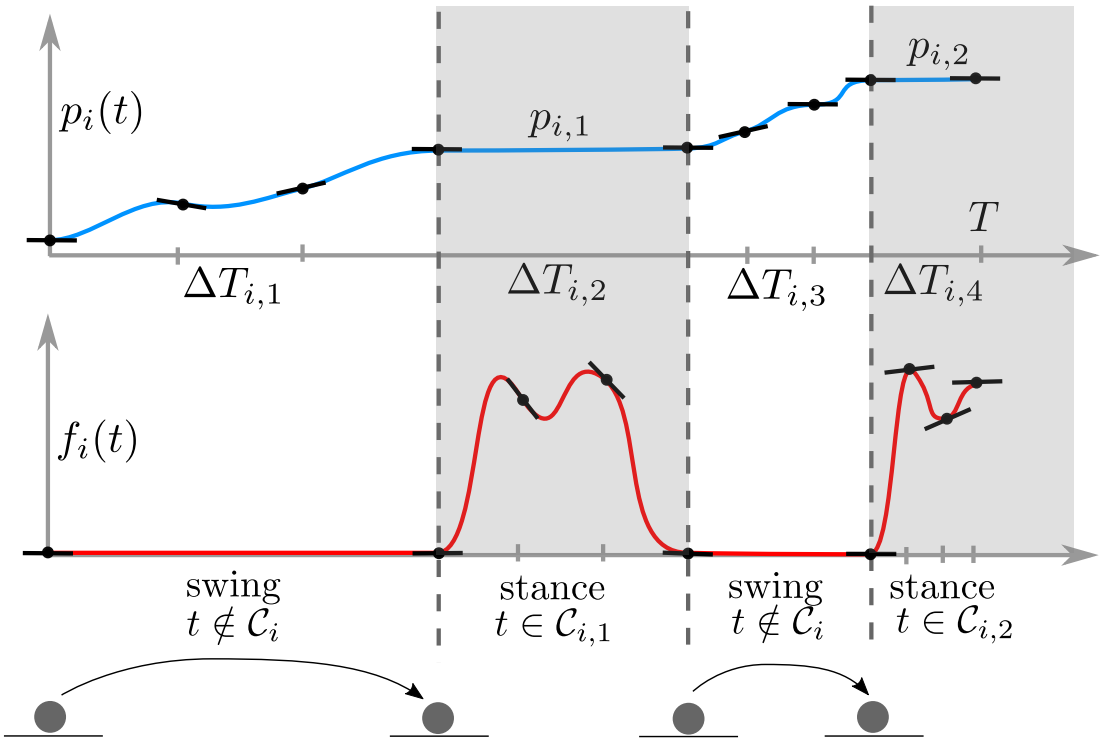


Figure 5.5: *Phase-based parameterization of foot i's motion $\mathbf{p}_i$ and force $\mathbf{f}_i$. Each phase (swing or stance) is represented by either a constant value or a sequence of cubic polynomials with continuous derivatives at the junctions. The optimizer is able to modify the phase durations $\Delta T_{i,j}$, thereby changing the shape of the functions. Performing this for all feet allows to generate arbitrary gait patterns, while still using continuous decision variables $\Delta T_{i,j}$.*

stance phase, and this never changes. However, the algorithm still has the flexibility to change the contact state at a given time by adapting the relevant phase durations and thereby make this time fall onto a polynomial of different contact state. Therefore, by changing these durations, all contact schedules/gaits can be generated. Since we are changing the durations, we must ensure that the total duration of each foot's motion and force spline ends at the specified total time. Therefore, we have the additional constraint for every foot i that $\Delta T_{i,1} + \cdots + \Delta T_{i,2n_{s,i}} = T$.

With this parameterization we directly impose that a foot in contact does not slip, more specifically, that the velocity of the foot motion during stance phase is zero. This is not a constraint of the optimization problem, but is ensured directly by our parameterization through a single, constant position variable $\mathbf{p}_{i,s}$ as

$$\dot{\mathbf{p}}_i(t \in \mathcal{C}_{i,s}) = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{p}_i(t \in \mathcal{C}_{i,s}) = \mathbf{p}_{i,s} = \text{const.} \tag{5.5}$$

If a foot is not in contact, no force can be produced. Therefore, we set each constant value

representing the force in the flight-phase to zero as

$$\mathbf{f}_i(t \notin \mathcal{C}_i) = \mathbf{0}. \tag{5.6}$$

The above restrictions (5.5), (5.6) are handled *before* starting the optimization and are equivalent to the LCP constraint $\dot{\mathbf{p}}_i(t)\mathbf{f}_i(t) = 0$ used in other TO formulations with automatic gait discovery. However, instead of checking this conditions at every sampling time t along the trajectory during the optimization, our phase-duration based optimization allows us to predefine this condition a-priori. This simplifies the problem for the solver and decreases computation time.

As seen in Fig. 5.5, we additionally ensure that the foot motion and force profiles are smooth at phase junctions (continuously differentiable) and thereby easier for the gradient-based solver to handle. Physically this is not required as contact with the environment can be impulsive, which abruptly zeros the foot velocity and spikes the contact force.

5.4.3 Terrain height constraint

A foot is only in contact if it is touching the terrain. Therefore, the height of the foot during contact must match the terrain at that 2D foot position $\mathbf{p}_{i,s}^{xy} = (p_{i,s}^x, p_{i,s}^y)$. The continuous height map $h_{terrain}(x, y)$ can be either manually specified if the objects in the environment are known or be generated from stereo camera data. We constrain the variable representing the constant foothold height of foot i during stance phase s by

$$p_i^z(t \in \mathcal{C}_{i,s}) = p_{i,s}^z = h_{terrain}(\mathbf{p}_{i,s}^{xy}). \tag{5.7}$$

5.4.4 Stance force constraints

For physically correct locomotion it is necessary that forces can only push into the contact surface, and not pull on it. For flat ground and an LIPM model this can be equivalently formulated as keeping the CoP inside the area spanned by the feet in contact. Since our formulation has explicit values for the contact forces we can directly constrain these as

$$f_n(t \in \mathcal{C}_{i,s}) = \mathbf{f}^T(t)\mathbf{n}(\mathbf{p}_{i,s}^{xy}) \geq 0, \tag{5.8}$$

where $\mathbf{n}(x, y)$ denotes the normal vector defining the slope of the terrain at position x, y. The scalar product extracts the component of the force that is orthogonal to the terrain. For flat ground, $\mathbf{n}(x, y) = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$ and the constraint simplifies to $f^z(t) \geq 0$.

It follows from Coulomb's law that pushing stronger into a surface allows exerting larger side-ways forces without slipping. This is equivalent to keeping tangential forces f_{t1}, f_{t2} inside the friction cone defined by the friction coefficient μ as $\sqrt{f_{t1}^2 + f_{t2}^2} < \mu f_n$. We approximate this friction cone by a friction pyramid, enforcing an upper and lower bound for the force in both tangential directions $\mathbf{t}_1, \mathbf{t}_2$. This pyramid approximation introduces

only negligible error but linearizes this constraint, simplifying the problem for the NLP solver. The constraint is given by

$$\begin{aligned} & -\mu f_n < f_{\{t_1,t_2\}} < \mu f_n \\ \Leftrightarrow \quad & |\mathbf{f}^T(t)\mathbf{t}_{\{1,2\}}(\mathbf{p}_{i,s}^{xy})| < \mu \mathbf{f}^T(t)\mathbf{n}(\mathbf{p}_{i,s}^{xy}). \end{aligned} \tag{5.9}$$

5.5 Results

This section discusses the variety of motions generated with the presented algorithm for a single-legged hopper, a biped robot and the quadruped robots ANYmal [11] and HyQ [5]. First, the motion plans, fulfilling all the specified physical constraints, are analyzed and discussed. Secondly, we demonstrate a subset of motions in the realistic physics simulator Gazebo as well as on the quadruped robot ANYmal [11].

This requires a controller that can reliably track the generated motion-plans by incorporating current sensor data to calculate the appropriate joint torques. This is not a trivial task, and just as much a research topic as generating the plans. Our controller solves a hierarchy of tasks using optimization to most accurately track the plans and is described in detail in [21]. In simulation, we demonstrate highly dynamic and full-body walking, trotting, pacing and galloping, all produced by the same method and tracked with the same controller. Additionally, we show that despite model mismatches, sensor noise, torque tracking inaccuracies and delays the motion plans are robust enough to be tracked on a real system. A quadruped trot and walk with optimized full 6D-body motion and directly planned contact forces are executed on a real system. These examples are another form of validation that our formulation produces motion plans that are physically feasible.

The accompanying video[2] shows a visualization of the generated motion plans using inverse kinematics and simulation and real robot experiments. The biped gap crossing example below can be seen in the video at 01:09 and is shown in Fig. 5.6. The motions are optimized with Towr [72] and visualized with Xpp [93], which also provide ROS bag files of some optimized motions.

5.5.1 Example: biped gap crossing

5.5.1.1 Dynamic consistency

A main focus in TO is to generate motions plans that are physically feasible. For shooting methods that optimize only over the inputs and integrate to get the state the dynamics are always fulfilled. For the used simultaneous method the dynamic constraint (5.2) is enforced only at discretized times (red nodes). We therefore calculate the true base vertical acceleration $\ddot{r}^z_{\text{true}} = m^{-1}(f^z_L + f^z_R) - g$ (black dashed line) from (5.2) using the optimized

[2]Video of generated motions: https://youtu.be/0jE46GqzxMM.

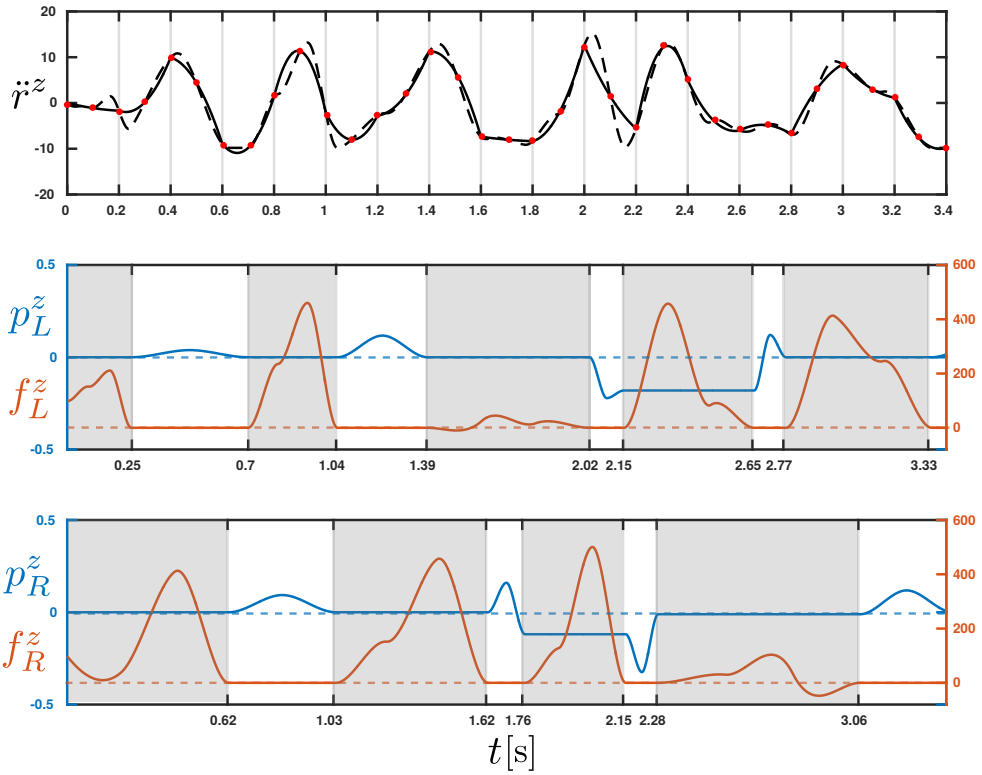


Figure 5.6: *A generated motion plan for a 20 kg-bipedal robot crossing a 1 m wide gap (see video at 01:09). The plots show the base vertical acceleration $\ddot{r}^z(t)$ as well as the vertical position and vertical force of the left (L) and right (R) leg. The planned base vertical acceleration is compared to the vertical body acceleration that results from evaluating (5.2) with the current footholds and forces. In this example we use fourth-order polynomials of duration 0.2 s for the base motion parameterization. The red nodes, spaced 0.1 s apart, show the times at which the dynamic constraint is enforced in the NLP. We use two third-order polynomials to parameterize each foot motion in swing-phase (white area), and three third-order polynomials for each force profile per foot in stance-phase (shaded area).*

forces and compare it to the optimized result $\ddot{r}^z$ (black solid line). The same evaluation can be done for the other five base coordinates $\ddot{r}^x, \ddot{r}^y, \dot{\omega}^x, \dot{\omega}^y, \dot{\omega}^z$. As can be seen, these values coincide exactly at the node values, as these are enforced by hard constraints, and deviate only slightly in between, e.g. during the dynamic sequence at t=2.0-2.2 s. The Root-Mean-Squared-Error (RMSE) for the base vertical acceleration is 1.8433 $\frac{m}{s^2}$.

In case more accuracy is required, dynamic constraints can be enforced at a finer grid, e.g., every 50 ms. However, one must keep in mind that (i) the dynamic model is an approximation of the true dynamic system and will also never be perfectly accurate (ii) enforcing the dynamics exactly might be unnecessary since the controller cannot even track the desired motions exactly, due to sensor noise, inaccurate force tracking and delays. Taking the above into account, sensible model accuracy must be chosen for each hardware, controller

Table 5.1: *NLP specs for bipedal gap crossing*

Horizon T: 4.4 s	dt-kinematic: 0.05 s	Variables: 926
Goal x: 3.7 m	dt-dynamic: 0.1 s	Constraints: 1543
Steps per foot $n_{s,L/R}$: 5	Iterations: 21	T-solve (Ipopt): **4.1 s**

and problem individually.

5.5.1.2 Foot contact constraints

The foot height $p^z(t)$ for the left and right foot is shown by the blue lines. We notice that the constant segments during stance phase (gray areas) are mostly at the tableau height $h_{terrain} = 0$ (blue dotted line) as required by (5.7). The z-positions lower than zero are where the biped steps into the sides of the gap while traversing it. In order the have gradient information available, we model the gap $h_{\text{gap}}(x, y)$ as a 5 m deep parabola, instead of a discretely changing ground height. This helps the NLP solver to converge to a solution.

Vertical forces $f^z(t)$ only exist whenever the corresponding foot is in stance phase (gray area). This is enforced by the parameterization of the force profile given by (5.6). During the stance phase, the force can only push into the terrain as required by (5.8). Since the normal direction of the terrain changes at the side of the gaps, negative z-forces are also physically feasible, as seen at $t = 2.8s$.

5.5.1.3 Automatic gait discovery

The algorithm can automatically change the initially provided gait sequence and timings depending on the terrain and desired task. The motion was initialized with a walking gait, with short two-leg support phases between every step. As can be seen in Fig. 5.6, flight-phases have been automatically inserted at e.g. $t = 0.62s - 0.7s$. We constrain each phase duration variable $\Delta T_{i,j}$ to be greater than 0.1 s in this example to avoid rapid swing- or extremely short stance phases. Flight-phases allow the solver to respect kinematic limits while still covering significant distance with few steps. Capabilities such as these show the advantages of optimizing all the aspects of the motion simultaneously.

5.5.1.4 Fast solver

As seen in Table 5.1 our formulation is able to generate the sample biped motion in 4.1 s. This includes optimization over the contact sequence, finding 6D-body, foothold and swing-leg motions as well as enforcing friction constraints. The kinematic constraint (5.1) is enforced every 0.05 s, while the dynamic constraint (5.2) is checked every 0.1 s.

Generating a motion of two steps per leg for a quadruped robot takes ∼300 ms. For the other motions shown in the video, with time horizons around 5 s involving dozens of steps, the algorithm takes ∼20 s. These values vary depending on the problem definition, terrain, and other parameters. Nonetheless, other approaches that can generate similarly complex motions often take orders of magnitude longer. The results were obtained using C++ code interfaced with Interior Point Method solver Ipopt [35] on an Intel Core i7/2.8 GHz Quadcore laptop. The Jacobians of the constraints are provided to the solver analytically, which is important for performance. Another factor that speeds up the optimization is that we do not use a cost function, as observed in Fig. 5.2. This also reduces the amount of tuning parameters, as the relative importance of the constraints does not have to be quantified – they all have to be fulfilled for a motion to be physically feasible.

5.5.2 Limitations and future work

For humanoid robots with heavy limbs deviating far from its nominal configuration the simplified Centroidal dynamics model (5.2) might not be sufficient. If a more accurate dynamic representation is necessary, the model can be refined by calculating the joint angles $\mathbf{q}$ from the optimized foot positions $\mathbf{p}$ using inverse kinematics and updating $\mathbf{I}(\mathbf{q})$. Likewise, optimized foot velocities $\dot{\mathbf{p}}$ can be converted into angular velocities $\boldsymbol{\omega}_i$ of each leg link (e.g., using the leg Jacobian) and used in (5.2) explicitly. These refinements will increase the nonlinearity of the dynamic constraint and possibly computation time, but can still be handled by the proposed formulation and NLP solver.

The solver enforces the terrain constraints (terrain collision, unilateral force, friction cone) only at the junctions of the 3^{rd}-order feet polynomials. Since these polynomials are usually short, violation of the constraints in between is often negligible. However, especially when a foot motion polynomial enforces the terrain constraint (5.7) only at the borders and an obstacle is in between, undesired terrain collision can occur.

Finally, for highly uneven terrain the solver is sometimes trapped in local minima. One way to simplify the problem for the solver is to fix the step timings, thereby not optimizing over the gait sequence. As long as the range of motion is large enough to reach the desired goal in the specified number of steps the solver consistently finds solutions to the problem and the solution time rapidly decreases.

5.6 Conclusion

We presented a TO formulation that is able to efficiently generate complex, highly dynamic motions for a variety of legged systems over non-flat terrain, while also optimizing over the contact sequence. The feasibility of the motion plan is demonstrated in simulation and on a real quadruped. In the future, we will transfer even more motions to the real system and also use this fast motion planner in an MPC fashion.

6 Conclusions and outlook

6.1 Summary

A variety of incremental contributions listed in Section 2.4 have led to the efficient motion-planning algorithm for legged robots presented in this thesis.

The premise of the investigation was that footholds and body motion are important quantities in legged locomotion and tightly coupled. Therefore, traditional body motion planning algorithms have been extended to generate optimal footholds and body motion plans simultaneously. This reduces heuristics and allows the footholds to be placed precisely where they are needed to guide the system to a specified goal.

Following this, we observed that different traditional planning approaches for quadruped robots, either based on the ZMP or the Capture Point, can be represented in a uniform TO formulation. We propose a vertex-based formulation of the support area constraint, which allows treating point-, line-, and area contacts in a uniform way. This LIPM-based TO formulation allows to efficiently generate a variety of also unconventional quadruped gaits (e.g., limping) that can be useful for more complex tasks or terrains.

However, these gaits were still limited to flat ground and the restricting assumptions of the LIPM. To increase the repertoire of motions, we model the robot as a 6D rigid body controlled directly by 3D external forces according to the SRBD. We remove the restriction of fixing the contact sequence in advance by a novel phase-based parameterization of the end-effector variables in the NLP. The final result of this thesis is an efficient motion-planning algorithm for legged robots that automatically determines the gait-sequence, step-timings, footholds, contact forces, swing-leg motions and 6-dimensional body motion over non-flat terrain. This generates motions for monopeds, bipeds and also for a physical quadruped robot.

6.2 Future directions

We see large potential in using TO methods to generate motions for legged robots. Without numerical assistance through mathematical optimization, it can become unfeasible to hand-design solutions with such complex interplay of components. This might become more evident as legged robots start to tackle increasingly difficult terrain and are required to perform higher dynamic motions.

The correct translation of the legged locomotion problem into a mathematical optimization problem has been tackled by this thesis. This algorithm can still be improved in a number of ways, which are explicit listed in Section 5.5.2. By transferring more of the motions onto hardware, possible shortcomings of the dynamic model or other formulations can be detected. The goal is for the algorithm to efficiently and reliably generate correct physical motions that can be tracked on the physical system.

It is worth working on the core algorithm and improving its capabilities according to Section 2.2, as it is a primary building block that MPC formulation depend on. In an MPC formulation the motion-plan is repeatedly generated starting from the current state of the system. This is an attractive approach, as without any additional methods, only by solving a problem repeatedly and fast, the system gains a variety of capabilities. If the control loop is fast enough, the robot will be reactive in case a foot slips, take corrective steps if pushed or re-plan its motion completely when the plan is not executing as predicted. Even further benefits of MPC include that, since motions are continuously re-optimized, modeling errors of the dynamics are implicitly corrected.

This direction of future research, improving the underlying motion-planning algorithm and following that embedding it in an MPC formulation on a physical system, seems promising. With some algorithmic work, efficient software implementations, as well as capable hardware, we might soon find it unremarkable when one of these highly remarkable technological systems crosses our daily path.

A

Appendix

A.1 Derivation of SRBD from Centroidal Dynamics

We derive the SRBD (1.5) for a single rigid body from the Centroidal Dynamics. The change of total momentum defined though the CMM $\mathbf{A}$ [10] as

$$\frac{\mathrm{d}(\mathbf{A}\dot{\mathbf{q}})}{\mathrm{d}t} = \mathbf{A}\ddot{\mathbf{q}} + \dot{\mathbf{A}}\dot{\mathbf{q}}. \tag{A.1}$$

From Assumption A2 follows that the joint velocities can be neglected. Therefore we consider only the linear and angular velocity of the base $\dot{\mathbf{q}}_u = [\dot{\mathbf{r}}, {}_I\boldsymbol{\omega}_{IB}]^T$ and define the first 6 columns of the CMM as

$$\mathbf{A}_u = \begin{bmatrix} m\mathbf{1}_3 & \mathbf{0} \\ \mathbf{0} & {}_I\mathbf{I}_r(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} m\mathbf{1}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{IB}\, {}_B\mathbf{I}_r\, \mathbf{R}_{IB}^T \end{bmatrix}. \tag{A.2}$$

Let m be the total of all link masses combined, $\mathbf{1}_3 \in \mathbb{R}^{3\times3}$ the identity matrix and ${}_I\mathbf{I}_r(\boldsymbol{\theta}) \in \mathbb{R}^{3\times3}$ the combined inertia of all individual limbs in nominal joint configuration, anchored at the CoM and expressed in coordinate axis parallel to the inertial frame (I). If this inertia matrix is expressed in a coordinate frame (B) rotating together with the rigid body, it is constant over time and denoted by ${}_B\mathbf{I}_r$. The base orientation $\boldsymbol{\theta}$ can be equivalently described through a rotation $\mathbf{R}_{IB} \in \mathbb{R}^{3\times3}$ from base (B) to inertial (I) frame.

From Assumption A3 follows that the full-body inertia remains constant, so ${}_B\dot{\mathbf{I}}_r = \mathbf{0}$. We introduce the skew-symmetric matrix $\mathbf{S}(\mathbf{a}) \in \mathbb{R}^{3\times3}$, which is a generalization of the cross product and satisfies $\mathbf{S}(\mathbf{a})\mathbf{b} = \mathbf{a} \times \mathbf{b}$. The time derivative of a rotation matrix can be expressed through $\dot{\mathbf{R}}_{IB} = \mathbf{S}({}_I\boldsymbol{\omega}_{IB})\mathbf{R}_{IB}$, where ${}_I\boldsymbol{\omega}_{IB}$ denotes the angular velocity between the inertial frame and the base frame, expressed in the inertial frame. With this we can

rewrite the derivative of the bottom right-corner of $\dot{\mathbf{A}}_u$ as

$$\begin{aligned}\frac{\mathrm{d}(\mathbf{R}_{IB}\,{}_B\mathbf{I}_r\,\mathbf{R}_{IB}^T)}{\mathrm{d}t} &= \dot{\mathbf{R}}_{IB\,B}\mathbf{I}_r\mathbf{R}_{IB}^T + \mathbf{R}_{IB\,B}\dot{\mathbf{I}}_r\mathbf{R}_{IB}^T + \mathbf{R}_{IB\,B}\mathbf{I}_r\dot{\mathbf{R}}_{IB}^T \\ &= \mathbf{S}({}_I\boldsymbol{\omega}_{IB})\mathbf{R}_{IB\,B}\mathbf{I}_r\mathbf{R}_{IB}^T + \mathbf{R}_{IB\,B}\mathbf{I}_r(\mathbf{S}({}_I\boldsymbol{\omega}_{IB})\mathbf{R}_{IB})^T \\ &= \mathbf{S}({}_I\boldsymbol{\omega}_{IB})\mathbf{R}_{IB\,B}\mathbf{I}_r\mathbf{R}_{IB}^T + \mathbf{R}_{IB\,B}\mathbf{I}_r\mathbf{R}_{IB}^T\mathbf{S}^T({}_I\boldsymbol{\omega}_{IB}) \\ &= \mathbf{S}({}_I\boldsymbol{\omega}_{IB})\,{}_I\mathbf{I}_r(\boldsymbol{\theta}) + {}_I\mathbf{I}_r(\boldsymbol{\theta})\mathbf{S}^T({}_I\boldsymbol{\omega}_{IB})\end{aligned} \tag{A.3}$$

Inserting this into back into (A.1) and using $\mathbf{S}^T({}_I\boldsymbol{\omega}_{IB})\,{}_I\boldsymbol{\omega}_{IB} = -{}_I\boldsymbol{\omega}_{IB} \times {}_I\boldsymbol{\omega}_{IB} = 0$ we get

$$\begin{aligned}\mathbf{A}_u\ddot{\mathbf{q}} + \dot{\mathbf{A}}_u\dot{\mathbf{q}} &= \begin{bmatrix} m\ddot{\mathbf{r}} \\ {}_I\mathbf{I}_r(\boldsymbol{\theta})\,{}_I\dot{\boldsymbol{\omega}}_{IB} + [\mathbf{S}({}_I\boldsymbol{\omega}_{IB})\,{}_I\mathbf{I}_r(\boldsymbol{\theta}) + {}_I\mathbf{I}_r(\boldsymbol{\theta})\mathbf{S}^T({}_I\boldsymbol{\omega}_{IB})]\,{}_I\boldsymbol{\omega}_{IB} \end{bmatrix} \\ &= \begin{bmatrix} m\ddot{\mathbf{r}} \\ {}_I\mathbf{I}_r(\boldsymbol{\theta})\,{}_I\dot{\boldsymbol{\omega}}_{IB} + {}_I\boldsymbol{\omega}_{IB} \times {}_I\mathbf{I}_r(\boldsymbol{\theta})\,{}_I\boldsymbol{\omega}_{IB} \end{bmatrix},\end{aligned} \tag{A.4}$$

which is the left-hand side of the SRBD equations given in (1.5).

A.2 Derivation of LIPM from SRBD

Here we derive the LIPM model (1.6) from assumptions applied to the SRBD. The first assumption A5 for the LIPM states that the angular velocity $\boldsymbol{\omega}$ and acceleration $\dot{\boldsymbol{\omega}}$ of the base must be zero. Substituting this into (1.5b) gives

$$\mathbf{0} = \sum_{i=1}^{n_i} \mathbf{f}_i \times (\mathbf{r} - \mathbf{p}_i) \quad = \sum_{i=1}^{n_i} \begin{bmatrix} f_{i,y}(r_z - p_{i,z}) - f_{i,z}(r_y - p_{i,y}) \\ f_{i,z}(r_x - p_{i,x}) - f_{i,x}(r_z - p_{i,z}) \\ f_{i,x}(r_y - p_{i,y}) - f_{i,y}(r_x - p_{i,x}) \end{bmatrix}, \tag{A.5}$$

where $\mathbf{p}_i$ and $\mathbf{f}_i$ denote the position and force of foot i, r the CoM position and x the index for the horizontal motion (y accordingly).

With our second assumption A6 that the foothold height $p_{i,z} = p_z$ is constant we rewrite the second row of (A.5) as

$$\begin{aligned}&\sum^{n_i} f_{i,x}(r_z - p_{i,z}) = \sum^{n_i} f_{i,z}(r_x - p_{i,x}) \\ \Leftrightarrow \quad &\sum^{n_i} f_{i,x} = \frac{1}{r_z - p_z}\left(r_x \sum^{n_i} f_{i,z} - \sum^{n_i} f_{i,z}p_{i,x}\right) \\ &= \frac{\sum^{n_i} f_{i,z}}{r_z - p_z}\left(r_x - \frac{\sum^{n_i} f_{i,z}p_{i,x}}{\sum^{n_i} f_{i,z}}\right)\end{aligned} \tag{A.6}$$

The last assumption A4 for the LIPM is that the CoM height stays constant, and thereby

the vertical acceleration zero. Using the linear z-dimension of (1.5a) we get

$$0 = m\ddot{r}_z = mg - \sum_{i=1}^{n_i} f_{i,z} \quad \Leftrightarrow \quad \sum_{i=1}^{n_i} f_{i,z} = mg. \tag{A.7}$$

With this we can rewrite (1.5a) as

$$\begin{aligned} m\ddot{r}_x = \sum_{i=1}^{n_i} f_{i,x} &= \frac{\sum^{n_i} f_{i,z}}{r_z - p_z} \left(r_x - \frac{\sum_{i=1}^{n_i} f_{i,z} p_{i,x}}{\sum_{i=1}^{n_i} f_{i,z}} \right) \\ &\overset{A.7}{=} \frac{mg}{r_z - p_z} \left(r_x - \frac{\sum_{i=1}^{n_i} f_{i,z} p_{i,x}}{mg} \right) \\ \Leftrightarrow \quad \ddot{r}_x &= \frac{g}{h}(r_x - p_{c,x}), \end{aligned} \tag{A.8}$$

where $h = r_z - p_z$ is the constant height of the LIPM and $\mathbf{p}_c$ the CoP produced by the vertical forces. This is the dynamic relationship of the LIPM in (1.6).

A.3 Derivation of Capture Point

This section derives the one-step Capture Point listed in (4.5). Consider the differential equation describing a LIPM (linear, constant coefficients, second order) in x-direction

$$\ddot{r}(t) - \frac{g}{h} r(t) = -\frac{g}{h} p_c, \tag{A.9}$$

where r denotes the CoM, p_c the CoP, g gravity acceleration and h the walking height. The general solution to the homogeneous part of the equation can be construct by the Ansatz $r(t) = e^{\alpha t}$ which leads to the characteristic equation $\alpha^2 e^{\alpha t} - \frac{g}{h} e^{\alpha t} = 0$, resulting in $\alpha = \pm\sqrt{\frac{g}{h}}$. Assuming constant input $p_{c,0}$ leads to the partial solution $r_p(t) = p_{c,0}$, and the space of solutions for the entire ODE is given by

$$r(t) = \beta_1 e^{\alpha t} + \beta_2 e^{-\alpha t} + p_{c,0} \tag{A.10}$$

where $\beta_1, \beta_2 \in \mathbb{R}$ are the free parameters describing the motion. Imposing the initial position $r(0) = \beta_1 + \beta_2 + p_{c,0} \overset{!}{=} r_0$ and velocity $\dot{r}(0) = \alpha\beta_1 - \alpha\beta_2 \overset{!}{=} \dot{r}_0$ we obtain

$$\beta_{1,2} = \frac{1}{2}(r_0 \pm \frac{\dot{r}_0}{\alpha} - p_{c,0}). \tag{A.11}$$

As $t \to \infty$ we require the velocity $\dot{r}(t)$ to remain at zero (pendulum at rest). With $\alpha \neq 0$ follows that $\lim_{t\to\infty} e^{-\alpha t} = 0$, so we must only ensure

$$\begin{aligned} \lim_{t\to\infty} \dot{r}(t) = \alpha\beta_1 \lim_{t\to\infty} e^{\alpha t} \overset{!}{=} 0 \quad &\Leftrightarrow \quad \beta_1 = 0 \\ \overset{A.11}{\Rightarrow} p_{c,0} &= r_0 + \alpha^{-1}\dot{r}_0, \end{aligned} \tag{A.12}$$

which is known as the one-step Capture Point originally derived in [40].

A.4 Dynamic constraint

The system dynamics constraint (4.11) enforced through $\ddot{\mathbf{r}}[t] = \mathbf{F}_2(\mathbf{x}[t], \mathbf{u}[t])$, with the local polynomial time $\underline{t} = (t - t_k)$, are formulated as

$$\begin{aligned} \ddot{\mathbf{r}}[t] &= \sum_{i=2}^{4} i(i-1)\mathbf{a}_{k,i}\underline{t}^{i-2} = \frac{g}{h}(\mathbf{r}(t) - \mathbf{p}_c(t)) \\ \Leftrightarrow \sum_{i=2}^{4} \mathbf{a}_{k,i}\underline{t}^{i-2} &\left(i(i-1) - \frac{g}{h}\underline{t}^2\right) = \frac{g}{h}(\mathbf{a}_{k,0} + \mathbf{a}_{k,1}\underline{t} - \mathbf{p}_c(t)). \end{aligned} \tag{A.13}$$

A.5 Hermite parameterization

Each cubic polynomial $x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$ introduced in (5.3) can either be parameterized by its coefficients *or* the value and first derivative and the end points x_0, x_1 and the duration ΔT as

$$\begin{aligned} a_0 &= x_0 \\ a_1 &= \dot{x}_0 \\ a_2 &= -\Delta T^{-2}[3(x_0 - x_1) + \Delta T(2\dot{x}_0 + \dot{x}_1)] \\ a_3 &= \ \Delta T^{-3}[2(x_0 - x_1) + \Delta T(\dot{x}_0 + \dot{x}_1)]. \end{aligned} \tag{A.14}$$

A.6 Euler angles and rates to angular velocities

The transformation from the optimized Euler angles $\boldsymbol{\theta}$ (order of application: yaw, pitch, roll) and rates $\dot{\boldsymbol{\theta}}$ to the angular velocities in world frame used in (5.2) are [94]

$$\begin{aligned} \boldsymbol{\omega} = \mathbf{C}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} &= \begin{bmatrix} \cos(\theta_y)\cos(\theta_z) & -\sin(\theta_z) & 0 \\ \cos(\theta_y)\sin(\theta_z) & \cos(\theta_z) & 0 \\ -\sin(\theta_y) & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix} \\ \dot{\boldsymbol{\omega}} &= \dot{\mathbf{C}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}}. \end{aligned} \tag{A.15}$$

Tables

Figures

Bibliography

[1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: stability and optimality", *Automatica*, vol. 36, no. 6, pp. 789–814, 2000. DOI: 10.1016/S0005-1098(99)00214-9 (cit. on p. 2).

[2] Flaticon, Freepik, and D. Gandy, http://www.freepik.com, https://www.flaticon.com, 2017 (cit. on p. 2).

[3] L. Righetti, J. Buchli, M. Mistry, and S. Schaal, "Inverse dynamics control of floating-base robots with external constraints: A unified view", *IEEE International Conference on Robotics and Automation*, pp. 1085–1090, 2011. DOI: 10.1109/ICRA.2011.5980156 (cit. on pp. 2, 10).

[4] F. Farshidian, E. Jelavic, A. W. Winkler, and J. Buchli, "Robust whole-body motion control of legged robots", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. DOI: 10.1109/IROS.2017.8206328 (cit. on p. 2).

[5] C Semini, N. G. Tsagarakis, E Guglielmino, M Focchi, F Cannella, and D. G. Caldwell, "Design of hyq - a hydraulically and electrically actuated quadruped robot", *Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011. DOI: 10.1177/0959651811402275 (cit. on pp. 4, 6, 32, 50, 66).

[6] A. Herzog, "Optimization-based motion generation for multiped robots in contact scenarios", PhD thesis, ETH Zurich, 2017. DOI: 10.3929/ethz-b-000199099 (cit. on pp. 4, 19).

[7] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics", *IEEE-RAS International Conference on Humanoid Robots*, 2014. DOI: 10.1109/HUMANOIDS.2014.7041375 (cit. on pp. 5, 20, 24, 37, 57).

[8] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation", in *IEEE International Conference on Intelligent Robots and Systems*, 2016, pp. 2703–2710. DOI: 10.1109/IROS.2016.7759420 (cit. on pp. 5, 19, 56).

[9] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion", *IEEE International Conference on Robotics and Automation*, no. 3, pp. 3555–3561, 2016. DOI: 10.1109/ICRA.2016.7487538 (cit. on pp. 5, 19, 24, 37, 56).

[10] D. E. Orin, A. Goswami, and S. H. Lee, "Centroidal dynamics of a humanoid robot", *Autonomous Robots*, vol. 35, no. 2-3, pp. 161–176, 2013. DOI: 10.1007/s10514-013-9341-4 (cit. on pp. 5, 56, 72).

[11] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "ANYmal - A highly mobile and dynamic quadrupedal robot", *IEEE International Conference on Intelligent Robots and Systems*, pp. 38–44, 2016. DOI: 10.1109/IROS.2016.7758092 (cit. on pp. 6, 60, 66).

[12] S. Seok, A. Wang, M. Y. M. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, and S. Kim, "Design principles for energy-efficient legged locomotion and implementation on the mit cheetah robot", *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1117–1129, 2015. DOI: 10.1109/TMECH.2014.2339013 (cit. on p. 6).

[13] Boston Dynamics, *Spotmini*, "Youtube: https://youtu.be/tf7IEVTDjng", 2016 (cit. on p. 6).

[14] Agility Robotics, *Cassie*, "Youtube: https://youtu.be/Is4JZqhAy-M", 2017 (cit. on p. 6).

[15] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot hrp-2", in *IEEE International Conference on Robotics and Automation*, vol. 2, 2004, pp. 1083–1090. DOI: 10.1109/ROBOT.2004.1307969 (cit. on pp. 6, 19).

[16] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah", *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4102–4109, 2017. DOI: 10.1109/IROS.2017.8206268 (cit. on pp. 6, 20).

[17] A. W. Winkler, D. C. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization", *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, pp. 1560–1567, 2018. DOI: 10.1109/LRA.2018.2798285 (cit. on pp. 6, 13, 22).

[18] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain", *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, Nov. 2010. DOI: 10.1177/0278364910388677 (cit. on pp. 7, 14, 24, 28, 29, 37, 43).

[19] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain", *IEEE International Conference on Robotics and Automation*, pp. 2554–2561, 2013. DOI: 10.1109/ICRA.2013.6630926 (cit. on pp. 7, 37, 56).

[20] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. Caldwell, and C. Semini, "Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5148–5154. DOI: 10.1109/ICRA.2015.7139916 (cit. on pp. 7, 24, 37, 56).

[21] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots", *IEEE Robotics and Automation Letters*, 2018. DOI: 10.1109/LRA.2018.2794620 (cit. on pp. 7, 19, 66).

[22] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point", *International Conference on Robotics and Automation*, pp. 1620–1626, 2003. DOI: 10.1109/ROBOT.2003.1241826 (cit. on pp. 7, 10, 19, 24, 37, 38, 56).

[23] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain", *IEEE International Conference on Robotics and Automation*, pp. 2665–2670, 2010. DOI: 10.1109/ROBOT.2010.5509805 (cit. on pp. 7, 56).

[24] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic Locomotion and Whole-Body Control for Quadrupedal Robots", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017. DOI: 10.3929/ethz-b-000174751 (cit. on pp. 7, 19).

[25] H. Hirukawa, S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, and M. Morisawa, "A universal stability criterion of the foot contact of legged robots - adios zmp", *IEEE International Conference on Robotics and Automation*, pp. 1976–1983, 2006. DOI: 10.1109/ROBOT.2006.1641995 (cit. on p. 7).

[26] Alphabet Waymo, *Firefly car*, "Image: https://waymo.com", 2016 (cit. on p. 9).

[27] DJI, *Phantom 2 drone*, "Image: https://www.dji.com/phantom-2", 2016 (cit. on p. 9).

[28] ANYbotics, *Anymal bear*, "Image: https://www.anybotics.com/anymal", 2018 (cit. on p. 9).

[29] Boston Dynamics, *Atlas*, "Image: https://www.bostondynamics.com/atlas", 2016 (cit. on p. 9).

[30] M. Diehl, H. G. Bock, H. Diedam, and P. B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control", *Lecture Notes in Control and Information Sciences*, vol. 340, pp. 65–93, 2006. DOI: 10.1007/978-3-540-36119-0_4 (cit. on pp. 11, 12, 43, 59).

[31] J. T. Betts, "Survey of numerical methods for trajectory optimization", *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998. DOI: 10.2514/2.4231 (cit. on pp. 11, 55).

[32] M. P. Kelly, "Transcription Methods for Trajectory Optimization: a beginners tutorial", *ArXiv e-prints*, Jul. 2017. arXiv: 1707.00284 [math.OC] (cit. on pp. 11, 12).

[33] S. Mehrotra, "On the implementation of a primal-dual interior point method", *SIAM Journal on optimization*, vol. 2, no. 4, pp. 575–601, 1992. DOI: 10.1137/0802028 (cit. on p. 11).

[34] J. Nocedal and S. J. Wright, *Sequential quadratic programming*. Springer, 2006. DOI: 10.1007/978-0-387-40065-5 (cit. on p. 11).

[35] A. Waechter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006. DOI: 10.1007/s10107-004-0559-y (cit. on pp. 11, 32, 50, 69).

[36] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*, 2002. DOI: 10.1137/S1052623499350013 (cit. on pp. 11, 50).

[37] D. Pardo, L. Moeller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating direct transcription and nonlinear optimization methods for robot motion planning", *IEEE Robotics and Automation Letters (RA-L)*, pp. 946–953, 2016. DOI: 10.1109/LRA.2016.2527062 (cit. on pp. 12, 24).

[38] A. W. Winkler, F. Farshidian, M. Neunert, D. Pardo, and J. Buchli, "Online walking motion and foothold optimization for quadruped locomotion", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5308–5313. DOI: 10.1109/ICRA.2017.7989624 (cit. on pp. 13, 21, 37, 56).

[39] A. W. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli, "Fast trajectory optimization for legged robots using vertex-based zmp constraints", *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, pp. 2201–2208, Oct. 2017. DOI: 10.1109/LRA.2017.2723931 (cit. on pp. 13, 22, 56).

[40] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery", in *IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 200–207. DOI: 10.1109/ICHR.2006.321385 (cit. on pp. 14, 19, 37, 41, 75).

[41] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004, pp. 2149–2154. DOI: 10.1109/IROS.2004.1389727 (cit. on p. 18).

[42] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control based on divergent component of motion", *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 355–368, 2015. DOI: 10.1109/TRO.2015.2405592 (cit. on p. 19).

[43] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement", *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010. DOI: 10.1163/016918610X493552 (cit. on pp. 19, 25).

[44] M Naveau, M Kudruss, O Stasse, C Kirches, K Mombaur, and P Souères, "A Reactive Walking Pattern Generator Based on Nonlinear Model Predictive Control", *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 10–17, 2017. DOI: 10.1109/LRA.2016.2518739 (cit. on pp. 19, 37, 38, 56).

[45] C. Gehring, C. D. Bellicoso, P. Fankhauser, S. Coros, and M. Rutter, "Quadrupedal locomotion using trajectory optimization and hierarchical whole body control", in *2017 IEEE International Conference on Robotics and Automation*, 2017, pp. 4788–4794. DOI: 10.1109/ICRA.2017.7989557 (cit. on p. 19).

[46] M Kudruss, M Naveau, O Stasse, N. Mansard, C Kirches, P Soueres, and K. Mombaur, "Optimal Control for Multi-Contact, Whole-Body Motion Generation using Center-of-Mass Dynamics for Multi-Contact Situations", *IEEE-RAS International Conference on Humanoid Robots*, pp. 684–689, 2015. DOI: 10.1109/HUMANOIDS.2015.7363428 (cit. on pp. 19, 56).

[47] S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon, and J. Pettré, "A reachability-based planner for sequences of acyclic contacts in cluttered environments", in *Robotics Research: Volume 2*, A. Bicchi and W. Burgard, Eds. Springer International Publishing, 2018, pp. 287–303. DOI: 10.1007/978-3-319-60916-4_17 (cit. on p. 19).

[48] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, "Trajectory generation for multi-contact momentum control", in *IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2015, pp. 874–880. DOI: 10.1109/HUMANOIDS.2015.7363464 (cit. on p. 19).

[49] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation", *IEEE-RAS International Conference on Humanoid Robots*, pp. 842–849, 2016. DOI: 10.1109/HUMANOIDS.2016.7803371 (cit. on pp. 19, 56).

[50] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems", *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966. DOI: 10.1080/00207176608921369 (cit. on p. 19).

[51] F. Romano, A. Del Prete, N. Mansard, and F. Nori, "Prioritized optimal control: a hierarchical differential dynamic programming approach", in *IEEE International Conference on Robotics and Automation*, 2015, pp. 3590–3595. DOI: 10.1109/ICRA.2015.7139697 (cit. on p. 19).

[52] A. Sideris and J. E. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems", in *American Control Conference*, 2005, pp. 2275–2280. DOI: 10.1109/ACC.2005.1470308 (cit. on p. 19).

[53] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems", in *American Control Conference*, 2005, pp. 300–306. DOI: 10.1109/ACC.2005.1469949 (cit. on p. 19).

[54] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4906–4913. DOI: 10.1109/IROS.2012.6386025 (cit. on p. 19).

[55] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 93–100. DOI: 10.1109/ICRA.2017.7989016 (cit. on pp. 19, 20, 24, 37, 56).

[56] M. Giftthaler, F. Farshidian, T. Sandy, L. Stadelmann, and J. Buchli, "Efficient kinematic planning for mobile manipulators with non-holonomic constraints using optimal control", in *IEEE International Conference on Robotics and Automation*, 2017, pp. 3411–3417. DOI: 10.1109/ICRA.2017.7989388 (cit. on p. 19).

[57] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: a model predictive control approach", in *IEEE-RAS International Conference on Humanoid Robotics*, 2017, pp. 577–584. DOI: 10.1109/HUMANOIDS.2017.8246930 (cit. on pp. 19, 20).

[58] F. Aghili, "A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: Applications to control and simulation", *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 834–849, 2005. DOI: 10.1109/TRO.2005.851380 (cit. on pp. 19, 32, 49).

[59] C. Hargraves and S. Paris, "Direct trajectory optimization using nonlinear programming and collocation", *Journal of Guidance, Control, and Dynamics*, 1987. DOI: 10.2514/3.20223 (cit. on pp. 19, 45).

[60] D. Pardo, M. Neunert, A. W. Winkler, R. Grandia, and J. Buchli, "Hybrid direct collocation and control in the constraint- consistent subspace for dynamic legged robot locomotion", in *Robotics, Science and Systems (RSS)*, 2017. DOI: 10.15607/RSS.2017.XIII.042 (cit. on pp. 20, 37, 56).

[61] I. Mordatch, M. de Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning", *ACM Transactions on Graphics*, vol. 29, no. 4, p. 1, 2010. DOI: 10.1145/1778765.1778808 (cit. on pp. 20, 24, 37, 56).

[62] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, "Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion", in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1096–1103. DOI: 10.1109/ICRA.2017.7989131 (cit. on pp. 20, 56).

[63] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernandez-Lopez, and C. Semini, "Simultaneous contact, gait and motion planning for robust multi-legged locomotion via mixed-integer convex optimization", *IEEE Robotics and Automation Letters*, 2017. DOI: 10.1109/LRA.2017.2779821 (cit. on p. 20).

[64] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne, "Locomotion skills for simulated quadrupeds", *ACM SIGGRAPH*, p. 1, 2011. DOI: 10.1145/1964921.1964954 (cit. on pp. 20, 24, 37, 56).

[65] C. Gehring, S. Coros, M. Hutter, M. Bloesch, P. Fankhauser, M. A. Hoepflinger, and R. Siegwart, "Towards automatic discovery of agile gaits for quadrupedal robots", *IEEE International Conference on Robotics and Automation*, pp. 4243–4248, 2014. DOI: 10.1109/ICRA.2014.6907476 (cit. on pp. 20, 56).

[66] C. Gehring, S. Coros, M. Hutter, C. Dario Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, and R. Siegwart, "Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot", *IEEE Robotics and Automation Magazine*, vol. 23, no. 1, pp. 34–43, 2016. DOI: 10.1109/MRA.2015.2505910 (cit. on pp. 20, 24).

[67] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds", *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, pp. 1502–1509, 2017. DOI: 10.1109/LRA.2017.2665685 (cit. on pp. 20, 24, 37, 57).

[68] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds", *IEEE Robotic and Automation Letters*, 2018. DOI: 10.1109/LRA.2018.2800124 (cit. on p. 20).

[69] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization", *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 1–8, 2012. DOI: 10.1145/2185520.2335394 (cit. on pp. 20, 24, 37, 57).

[70] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact", *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2013. DOI: 10.1177/0278364913506757 (cit. on pp. 20, 24, 37, 57).

[71] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Hierarchical planning of dynamic movements without scheduled contact sequences", in *IEEE International Conference on Robotics and Automation*, 2016, pp. 4636–4641. DOI: 10.1109/ICRA.2016.7487664 (cit. on p. 20).

[72] A. W. Winkler, *TOWR – An open-source Trajecetory Optimizer for Legged Robots in C++*, http://wiki.ros.org/towr (cit. on pp. 22, 55, 66).

[73] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running", *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 5, pp. 783–792, 2010. DOI: 10.1109/TMECH.2009.2035112 (cit. on pp. 24, 37, 56).

[74] K. H. Koch, K. Mombaur, and P. Soueres, "Optimization-based walking generation for humanoid robot", *IFAC Proceedings*, vol. 45, no. 22, pp. 498–504, 2012. DOI: 10.3182/20120905-3-HR-2030.00189 (cit. on p. 24).

[75] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions", *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, 2013. DOI: 10.1177/0278364913478990 (cit. on p. 24).

[76] P. Hämäläinen, S. Eriksson, E. Tanskanen, V. Kyrki, and J. Lehtinen, "Online motion synthesis using sequential monte carlo", *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 51, 2014. DOI: 10.1145/2601097.2601218 (cit. on p. 24).

[77] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain", *IEEE International Conference on Robotics and Automation*, pp. 811–818, 2008. DOI: 10.1109/ROBOT.2008.4543305 (cit. on pp. 24, 37).

[78] A. W. Winkler, I. Havoutis, S. Bazeille, J. Ortiz, M. Focchi, R. Dillmann, D. Caldwell, and C. Semini, "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)*, 2014, pp. 6476–6482. DOI: 10.1109/ICRA.2014.6907815 (cit. on pp. 24, 37, 56).

[79] M. Vukobratović and B. Borovac, "Zero-moment point - thirty five years of its life", *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004. DOI: 10.1142/S0219843604000083 (cit. on pp. 24, 37, 55).

[80] H. Diedam, D. Dimitrov, P.-b. Wieber, K. Mombaur, and M. Diehl, "Online walking gait generation with adaptive foot positioning through linear model predictive control", *IEEE International Conference on Intelligent Robots and Systems*, 2009. DOI: 10.1109/IROS.2008.4651055 (cit. on pp. 25, 37, 38, 56).

[81] M. Mistry, J. Buchli, and S. Schaal, "Inverse dynamics control of floating base systems using orthogonal decomposition", *IEEE International Conference on Robotics and Automation*, no. 3, pp. 3406–3412, 2010. DOI: 10.1109/ROBOT.2010.5509646 (cit. on pp. 32, 49).

[82] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation", *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987. DOI: 10.1109/JRA.1987.1087068 (cit. on p. 32).

[83] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State Estimation for Legged Robots - Consistent Fusion of Leg Kinematics and IMU", *Robotics: Science and Systems*, p. 2005, 2012. DOI: 10.15607/RSS.2012.VIII.003 (cit. on pp. 32, 50).

[84] M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini, "RobCoGen : a code generator for efficient kinematics and dynamics of articulated robots , based on Domain Specific Languages", *Journal of Software Engineering for Robotics*, vol. 7, no. July, pp. 36–54, 2016 (cit. on pp. 32, 50).

[85] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems", *IEEE International Conference on Robotics and Automation*, no. 724454, pp. 1366–1373, 2016. DOI: 10.1109/ICRA.2016.7487270 (cit. on p. 37).

[86] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Control of dynamic gaits for a quadrupedal robot", *IEEE International Conference on Robotics and Automation*, pp. 3287–3292, 2013. DOI: 10.1109/ICRA.2013.6631035 (cit. on p. 37).

[87] B. J. Stephens and C. G. Atkeson, "Push recovery by stepping for humanoid robots with force controlled joints", in *IEEE-RAS International Conference on Humanoid Robots*, 2010, pp. 52–59. DOI: 10.1109/ICHR.2010.5686288 (cit. on pp. 37, 38).

[88] A. Herdt, N. Perrin, and P. B. Wieber, "Walking without thinking about it", *International Conference on Intelligent Robots and Systems*, pp. 190–195, 2010. DOI: 10.1109/IROS.2010.5654429 (cit. on pp. 37, 38, 56).

[89] D. Serra, C. Brasseur, A. Sherikov, D. Dimitrov, and P.-b. Wieber, "A Newton method with always feasible iterates for Nonlinear Model Predictive Control of walking in a multi-contact situation", *IEEE-RAS International Conference on Humanoid Robots*, no. 1, pp. 6–11, 2016. DOI: 10.1109/HUMANOIDS.2016.7803384 (cit. on p. 38).

[90] H.-W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds", in *Robotics: Science and Systems*, 2015. DOI: 10.15607/RSS.2015.XI.047 (cit. on p. 56).

[91] A. Ibanez, P. Bidaud, and V. Padois, "Emergence of humanoid walking behaviors from mixed-integer model predictive control", *IEEE International Conference on Intelligent Robots and Systems*, pp. 4014–4021, 2014. DOI: 10.1109/IROS.2014.6943127 (cit. on p. 56).

[92] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization", *IEEE-RAS International Conference on Humanoid Robots*, pp. 279–286, 2015. DOI: 10.1109/HUMANOIDS.2014.7041373 (cit. on p. 56).

[93] A. W. Winkler, *XPP – A collection of ROS packages for the visualization of legged robots*, http://wiki.ros.org/xpp (cit. on p. 66).

[94] C. Gehring, D. Bellicoso, M. Bloesch, H. Sommer, P. Fankhauser, M. Hutter, and R. Siegwart, *Kindr Library - Kinematics and Dynamics for Robotics*, https://docs.leggedrobotics.com/kindr/cheatsheet_latest.pdf, 2016 (cit. on p. 75).

Curriculum Vitae

Alexander W. Winkler (born in 1988 in Germany) is a PhD student at the Agile and Dexterous Robotics Lab and the Robotic Systems Lab at ETH Zurich under the supervision of Prof. Dr. Jonas Buchli and Prof. Dr. Marco Hutter. He received his Bachelor's and Master's Degree (with distinction) in Mechanical Engineering from the Karlsruhe Institute of Technology (KIT) in 2012 and 2013. Before starting his PhD in 2014, he was a researcher at the Dynamic Legged Systems Lab headed by Dr. Claudio Semini at the Italian Institute of Technology (IIT). His research focuses on the optimal planning and control of dynamic motions for legged systems.

Alexander W. Winkler
www.awinkler.me

List of publications

The following presents selected publications (including videos) contributing to the results in this thesis. For an up-to-date list, please see Google Scholar.

1. **Alexander W. Winkler**, C. Dario Bellicoso, Marco Hutter, Jonas Buchli. *Gait and Trajectory Optimization for Legged Systems through Phase-based Endeffector Parameterization.* In Robotics and Automation Letters (**RA-L**). 2018. (Video)

2. **Alexander W. Winkler**, Farbod Farshidian, Diego Pardo, Michael Neunert, Jonas Buchli. *Fast Trajectory Optimization for Legged Robots using Vertex-based ZMP Constraints.* In Robotics and Automation Letters (**RA-L**). 2017. (Video)

3. **Alexander W. Winkler**, Farbod Farshidian, Michael Neunert, Diego Pardo, Jonas Buchli. *Online Walking Motion and Foothold Optimization for Quadruped Locomotion.* In IEEE International Conference on Robotics and Automation (**ICRA**). 2017. (Video)

4. **Alexander W. Winkler**, Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin Caldwell, Claudio Semini. *Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain.* In IEEE International Conference on Robotics and Automation (**ICRA**), 2015. (Video)

5. **Alexander W. Winkler**, Ioannis Havoutis, Stephane Bazeille, Jesus Ortiz, Michele Focchi, Ruediger Dillmann, Darwin Caldwell, Claudio Semini. *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**), 2014. (Video)

6. Diego Pardo, Michael Neunert, **Alexander W. Winkler**, Ruben Grandia, Jonas Buchli. *Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion.* In Robotics, Science and Systems (**RSS**), 2017. (Video)

7. Michael Neunert, Farbod Farshidian, **Alexander W. Winkler**, Jonas Buchli. *Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds.* In IEEE Robotics and Automation Letters (**RA-L**). 2017. (Video)

8. Diego Pardo, Lukas Moeller, Michael Neunert, **Alexander W. Winkler**, Jonas Buchli. *Evaluating direct transcription and nonlinear optimization methods for robot motion planning.* IEEE Robotics and Automation Letters (**RA-L**), 2016. (Video)

9. Farbod Farshidian, Michael Neunert, **Alexander W. Winkler**, Gonzalo Rey, Jonas Buchli. *An Efficient Optimal Planning and Control Framework For Quadrupedal Locomotion.* In IEEE International Conference on Robotics and Automation (**ICRA**). 2017. (Video)

10. Farbod Farshidian, Edo Jelavić, **Alexander W. Winkler**, Jonas Buchli. *Robust Whole-Body Motion Control of Legged Robots.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (**IROS**), 2017. (Video)

11. Jonas Buchli, Farbod Farshidian, **Alexander W. Winkler**, Timothy Sandy, Markus Gifthaler, *Optimal and Learning Control for Autonomous Robots*, arXiv:1708.09342, 2017.

12. Carlos Mastalli, Ioannis Havoutis, **Alexander W. Winkler**, Darwin Caldwell, Claudio Semini. *On-line and on-board planning for quadrupedal locomotion using practical, on-board perception.* IEEE International Conference on Practial Robot Applications, 2017.

List of software

The following software has been developed and open-sourced under the BSD3 license as part of this thesis. For an up-to-date list, visit https://github.com/awinkler.

TOWR generates physically feasible motions for legged robots by solving an optimization problem. Physical constraints as well as a desired goal position are given to the solver that then generates the motion plan. TOWR generates 4 step monoped hopping, biped walking, or a complete quadruped trotting cycle, while optimizing over the gait and step durations, in less than 100 ms.

https://github.com/ethz-adrl/towr

IFOPT is a unified Eigen-based interface to use Nonlinear Programming solvers, such as Ipopt and Snopt. The user defines the solver independent optimization problem by set of C++ classes resembling variables, cost and constraints. Subsequently, the problem can then be solved with either solver. This package can also be dropped in your catkin workspace.

https://github.com/ethz-adrl/ifopt

XPP is a collection of ROS-packages for the visualization of motion plans for floating-base robots. Apart from drawing support areas, contact forces and motion trajectories in RVIZ, it also displays these plans for specific robots. Current robots include a one-legged, a two-legged hopper, HyQ and a quadrotor.

https://github.com/leggedrobotics/xpp