Mixed-Integer Convex Optimization for Planning Aggressive Motions of Legged Robots Over Rough Terrain

by

Andrés Klee Valenzuela

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2016

Author .....
Department of Mechanical Engineering
November 16, 2015

Certified by .....
Russ Tedrake
Associate Professor
Thesis Supervisor

Accepted by .....
Rohan Abeyaratne
Chairman, Department Committee on Graduate Students

Mixed-Integer Convex Optimization for Planning Aggressive Motions of Legged Robots Over Rough Terrain

by

Andrés Klee Valenzuela

Submitted to the Department of Mechanical Engineering
on November 16, 2015, in partial fulfillment of the
requirements for the degree of
Doctor of Science in Mechanical Engineering

Abstract

Planning dynamic motions for a legged robot entails addressing both the continuous question of how its joints should move and the combinatorial question of which hand or foot should touch which surface and in what order. Fortunately, these two questions are linked by the centroidal dynamics of the robot, which we can express either in terms of its joint angle trajectories or in terms of its foot placements and applied forces. Based on this insight, I formulate a pair of mathematical programs for planning highly dynamic motions for legged robots. The first is a mixed-integer convex program, specifically, a mixed-integer quadratic program (MIQP), that yields a sequence of footholds/handholds as well as center of mass (COM) and angular momentum trajectories. The second is a trajectory optimization, formulated as a nonlinear program (NLP), that returns trajectories for the COM, angular momentum, and joint angles subject to the footholds/handholds chosen by the MIQP step. While any number of trajectory optimization schemes could be used here, we present one which is particularly useful in this case, as it enforces the system's dynamics directly in terms of its COM motion and angular momentum. As a result, the solution to the MIQP provides constraints (where each end-effector is required to make contact with the environment) for the NLP and also gives seeds for the decision variables corresponding to the robot's centroidal motion. Thus, the three primary contributions of this thesis are: an MIQP-based approach to gait selection over irregular terrain, a trajectory optimization formulation for floating-base systems subject to external forces and kinematic constraints, and a planning methedology that integrates both of those to generate highly dynamic motions in challenging environments. I apply these techniques to models of a quadruped and a humanoid (Boston Dynamics' LittleDog and Atlas respectively) to generate motion plans for running, jumping, and other dynamic behaviors.

Thesis Supervisor: Russ Tedrake
Title: Associate Professor

Acknowledgments

Sure playing with robots every day is fun, but it's the people that make coming into work every day worth it. I've been blessed to work with so many brilliant, generous, and just plain fun people during my time at MIT. Many thanks to all my colleagues in the Biomimetic Robotics Lab and the Robot Locomotion Group. I'd like to especially thank my fellow VRC/DRC planners, Hongkai Dai and Robin Deits for their sharing their insights, expertise, and sense of humor.

Being part of the MIT DRC Team was an amazing experience, and I'm incredibly grateful to Russ and Seth for letting me be a part of it. Thanks to the whole team for all their hard work, fantastic engineering, and the countless Foodler dinners we shared.

Thank you to Russ for giving me the chance to join the Robot Locomotion Group — I can't imagine what my MIT experience would have been like otherwise. You opened my eyes to a new world of robotics research and I'm eternally grateful. Thanks also to the rest of my committee — Nevile Hogan, Tomás Lozano-Pérez, and Alberto Rodriguez — for your comments, direction, and advice.

Finally, my most profound thanks to my family. Without them I definitely wouldn't be here today. Thank you, Mom and Dad, for your love, generosity, and endless support for me, Christina and the kids. Thank you, kiddos, for putting up with all of Daddy's "working late" days and "working at home" days. Thank you for your drawings, your dances, your stories and your joy. And finally, thank you, Christina, my rock, my muse, my bride. Thank you for keeping me talking me down, cheering me up, and making me laugh so many times over the last six years.

+AMDG+

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

A humanoid robot has been deployed to shut down a nuclear power plant that is in danger of melting down after an earthquake. To reach the controls it must cross the rubble-strewn control room. How would we choose a path for the robot to navigate this environment that it has never encountered previously?

A quadrupedal robot stands on one side of a rocky plain. On the other side are a group of soldiers who need the supplies that the robot is carrying. To reach them, however, the robot must not only cross the irregular terrain — it must do so at high speed to avoid exposing itself to enemy fire. How would we choose a trajectory for the robot to follow across the terrain? If the terrain includes a wide gap, how would we determine whether the robot can simply step over it or whether it needs to jump?

Current motion-planning techniques do not address these sorts of scenarios, in which careful selection of footsteps needs to be considered alongside the dynamics of the robot. Both of these scenarios demonstrate a need for planning techniques that let us determine the appropriate course of action for a robot in situations where "appropriateness" depends on the dynamics of the robot and the geometry of obstacles and available supports in its environment. This is what I will refer to as "dynamic motion planning." It is an extremely challenging problem.

In this thesis, rather than try to find a universal solution method for the dynamic

motion planning problem, I treat the special case of legged robots. While still a broad class of robots, legged robots share certain characteristics that we can leverage to our advantage. These allow us to solve simplified versions of the planning problem that still provide useful results. Moreover, we can use our domain knowledge to split the problem into pieces more easily digested by numerical methods. These simplifications and decompositions form the primary theme of this thesis.

1.2 What makes this problem hard?

At first glance, getting a legged robot over rough terrain does not appear to be such a hard problem. We see robots like Boston Dynamics' BigDog, LS3, Spot, and Atlas walk and even run outdoors. The MIT Cheetah 2 can clear obstacles up to 80% of its leg-length [1]. Many contestants in the DARPA Robotics Challenge walked over slanted cinderblocks and stairs [2, 3]. Several other groups have demonstrated quadrupedal locomotion over rough terrain [4–10]. Despite how impressive these accomplishments are, however, they do not fully address the scenarios raised in the previous section.

One approach to legged locomotion on irregular terrain, perhaps best demonstrated by BostonDynamics' robots, is to rely on a robust controller — start out across the terrain and give little consideration to foot placement. If the terrain is complex though, this may lead us to dead ends where the robot cannot make any further progress. This is inadequate for the proposed scenarios, as we will not know that the robot is going to fail until it is in the middle of executing its task. We would like our planning methods to give us an a priori indication of whether the task is achievable.

Alternatively, we can specialize our planning to the point of performing a single maneuver. The MIT Cheetah team has done this to great success [1]. For running jumps over obstacles, their method provides excellent solutions very quickly. In order to do this, however, they restrict their approach to that specific task. Our goal here, however is to provide an approach that is more general — that works across a wider array of terrains and robots.

A third, more general approach, is to enforce conservative constraints on the motions

that the robot is allowed to perform. This allows for planning approaches that can treat a variety of terrains. Some of these approaches use quasi-static motion planning, in which we require the robot's center of mass (COM) to lie within the support polygon of its feet at all times [4, 5, 11], while others use a dynamic stability criterion which requires that the Zero Moment Point (ZMP) exist and lie inside the support polygon at all times [6, 7, 9, 10]. In more challenging scenarios, however, there may simply be no path that satisfies those constraints.

One of the key difficulties in treating this problem in a general way is that it combines both discrete and continuous aspects. Consider the scenario shown in Fig. 1-1. The choice

Figure 1-1: A quadrupedal robot prepares to cross a set of staggered platforms

of which platform to step on and with which foot will be instrumental in determining whether our robot makes it to the other side. Such discrete choices do not fit well into the continuous techniques usually used for trajectory optimization. Some approaches treat this difficulty by solving the discrete problem of end-effector placement prior to kinematic or dynamic planning for the rest of the robot [9, 10]. However, for more difficult terrain, we cannot rely on a purely discrete approach to choosing footholds. Which support sequences are feasible depends on the robot's kinematics and dynamics. Unfortunately, the dynamics and kinematics of a legged robot are nonlinear and even non-convex. These considerations place the dynamic motion planning problem for legged robots into mixed-integer non-convex programming — the farthest reaches of the Mathematical Programming realm, a place where there be dragons.

1.3 Approach

In this introduction, I have showed that there is a gap between the problems solvable with the current state-of-the-art in locomotion planning and the general dynamic motion planning problem for legged robots. This thesis demonstrates how we can diminish that gap by using some of the characteristic elements of that problem to our advantage. Chapter 2 gives a more thorough review of existing planning techniques for legged robots. In Chapter 3 I set out the general form of the planning problem we wish to solve. This general form is a large MINCP, that does not readily admit solution. In the chapters that follow, I present a series of approximations and decompositions that break this problem into more tractable chunks. In Chapter 4, we focus our attention on the discrete problem of searching for an appropriate sequence of contacts between the robot and the environment, and only as much of the dynamics as is necessary to inform that search. The result is a smaller MINCP, whose mixed-integer quadratic programming (MIQP) relaxation can be used to resolve the discrete planning problem.

I then take the results of that optimization and use them to formulate a continuous non-linear program (NLP) that fills in the joint-angle trajectories. This problem is continuous because the footstep assignment problem was solved in the previous step — that result is incorporated into the NLP as kinematic constraints on the robot's end-effectors. Chapter 5 presents a solution method that I developed with H. Dai for trajectory optimization on legged-robots. In Chapter 6, I detail how the solutions found in Chapter 4 can be used to formulate a problem of the form treated in Chapter 5. Examples using the humanoid robot Atlas and the quadrupedal robot LittleDog are given throughout.

Chapter 2

Related Work

I frame this discussion of the related work in terms of the flowchart shown in Fig. 2-1. On the left of Fig. 2-1 are the components that make up the planning scenario: the

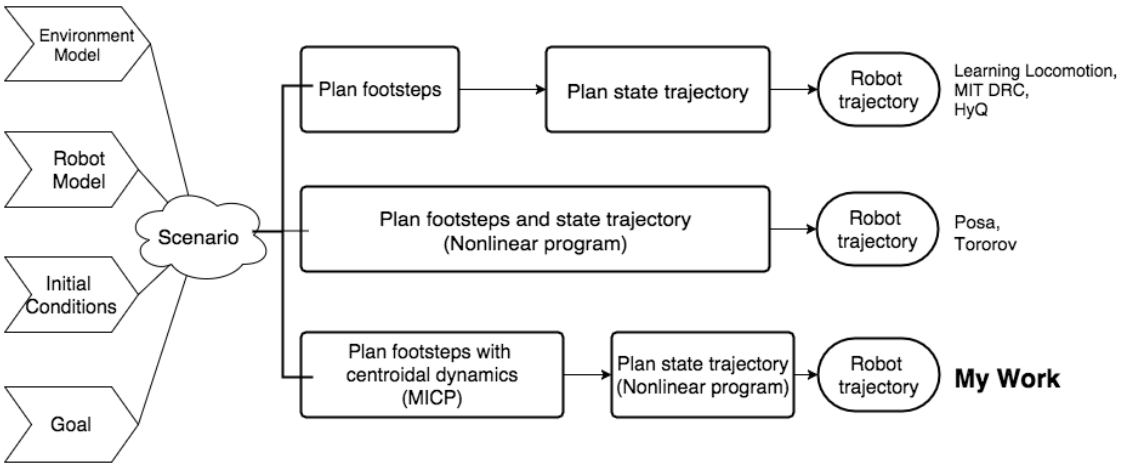


Figure 2-1: Approaches to planning for legged robots

models of the robot and its environment, the robot's initial state, and the goals for the planned motion (e.g. final position, total duration, minimizing energy expenditure). The planning problem is then to generate a robot trajectory based on the given scenario. I divide approaches to this planning problem into three tracks:

- Kinematic footstep-planning followed by dynamically feasible whole-body planning
- Simultaneous solution of whole-body planning and footstep planning

- Footstep-planning with centroidal dynamics + dynamically feasible whole-body planning

The first two of these tracks cover most of the strategies for legged-robot motion planning in the literature.

2.1 Kinematic footstep-planning followed by dynamically feasible whole-body planning

Methods following the first track start by computing end-effector placements without considering the dynamics of the robot. They then use those end-effector placements to generate dynamically feasible motion, either through another planning step or through a controller. To ensure that the end-effector placements chosen in the first step can be achieved, these methods typically enforce conservative constraints on the relative position of those placements. This restricts the set of motions that such methods can generate. We now consider approaches to each of the two steps on this track.

2.1.1 Kinematic footstep planning

As noted by Deits and Tedrake [12], these can be roughly divided into two camps: graph-based approaches that plan over a discrete set of actions (e.g. [5, 7, 9–11, 13–16]), and continuous approaches that optimize end-effector placements within a continuous space (e.g. [12, 17, 18]). Another crucial distinction is between methods that seek to satisfy a quasi-static criterion [4, 5, 11, 13], and those that use a dynamic balancing criterion based on the ZMP [6, 7, 9, 10, 17].

Graph-based approaches

Beyond the distinctions mentioned above, the graph-based approaches differ from one another in their action sets and the search algorithm used. For some, each action is a transition between complete configurations of the robot [13, 14, 16]. For others, each action is a transition between poses of the robot's floating base, in which case the remaining

configuration variables are filled in either by a post-processing step or by the controller [9–11]. For still others, each action is a transition between "stances" — tuples containing the position of each foot [15]. Most of the graph-based approaches use a heuristic-guided search, such as A* [13, 14] or one of its sub-optimal variants, such as ARA* [7, 9, 10] or R* [15]. Other search algorithms used include value iteration [5] and RRT [16].

Optimization-based approaches

In contrast to these graph-based techniques, some approaches optimize the positions of the robot's end-effectors in continuous space. Fallon et al. [18] present a footstep planning method that uses local non-convex optimization. Herdt et al. [17] present a method that employs linear model-predictive control (LMPC), but in order to do so they assume the orientations of the robot's feet are pre-specified, as including the orientation of the feet makes the planning problem non-convex. Herdt, Perrin, and Wieber [19] propose an additional quadratic program (QP) to solve for those foot orientations given a desired yaw-rate. Solving this QP followed by the LMPC problem gives a footstep plan that includes rotations of the feet. This line of work [17, 19] is an outlier in this section, as it *does* consider the dynamics of the robot (through a simplified model). However, like the method presented by Fallon et al. [18], it *does not* consider footstep placement in the context of obstacle avoidance — it presumes that the entire ground-plane can be stepped on safely. The approach I present in Chapter 4 also considers a simplified model of the robot's dynamics, but does so in a way that accounts for obstacles in the environment.

Deits and Tedrake [12] incorporate both foot orientation and obstacle avoidance into their method by formulating the problem as a mixed-integer convex program (MICP). They treat the orientation of the feet by approximating $\sin(\cdot)$ and $\cos(\cdot)$ with piecewise-linear functions. To ensure that the footsteps avoid obstacles, they restrict them to lie in the union of a pre-computed set of convex configuration space regions. My approach to footstep planning in Chapter 4 builds on both of these ideas, while adding consideration of the robots' dynamics.

2.1.2 Dynamically feasible whole-body planning

This thesis treats motion planning as it relates to dynamical systems. As such, we are interested in planning algorithms whose outputs are dynamically feasible trajectories. We discuss the three main classes of approaches to this problem in the literature: sampling-based, trajectory-optimization-based, and simplified-model-based.

Sampling-based approaches

Sampling-based planners vary widely both in their applications and the particulars of their algorithms, but the common thread is exploration of a search space through random sampling. These planners can be split into two primary groups: single-query and multi-query. Members of the former group attempt to solve a single planning problem, while members of the latter attempt to maintain data-structures that allow multiple planning problems to be solved. In situations where the environment may change substantially between planning problems, the former may be more efficient.

Variations on the rapidly-exploring random tree (RRT) algorithm introduced by LaValle and Kuffner [20] form the basis of many single-query probabilistic planners, e.g. [21–26]. All of these algorithms generate a tree in the configuration space of the robot. In each iteration a sample point is chosen in the configuration space. If the sample point is feasible (not in collision with the environment and satisfying all applicable constraints) then an attempt is made to extend the tree in the direction of that point. First the node in the tree closest to the sample point is chosen as the parent for the new node. Then a new node is added at some point between the parent and the sample point. The distinction between the different variants of RRT lies mainly in the methods of selecting samples and generating the new point between the parent and the sample.

In the case of systems with differential constraints, LaValle and Kuffner propose finding the new node by simulating the system forwards from the parent node with some control input [20]. While this approach successfully solves some kinodynamic planning problems, as noted by several authors [22–24], this can cause difficulties when only a small portion of the configuration space is accessible by the robot subject to dynamics. In that case, the

majority of sample points chosen will be in portions of the configuration space that the tree cannot reach [22]. Shkolnik, Walter, and Tedrake address this by adding reachability nodes to the tree [23]. New sample points are accepted only if they are closer to a reachability node than to one of the primary nodes. This focuses the search on portions of the state space towards which the tree can actually expand. Yershova et al. [22, 24] present a similar approach, the Dynamic Domain RRT (DDRRT) which maintains a representation of the explored portion of the free space. Sample points are drawn from this region, which is constructed such that there is more space at the edges of the tree. Though this approach was originally implemented to address kinematic constraints, the authors of [22] state that it could be used for differential constraints as well. Jaillet and Porta have taken this idea further, constructing an explicit map of the accessible portion of the space with techniques from differential geometry [26].

Trajectory optimization-based approaches

Another approach to planning dynamically-feasible trajectories is through trajectory optimization [27–30], in which the planning problem is formulated as the minimization of some objective function subject to a set of constraints. This set includes, at a minimum, the differential and algebraic constraints encoded in the equations of motion. The objective function depends on the state and control trajectories of the robot. This approach to motion planning is closely tied to optimal control, and trajectory optimization methods have their roots in that discipline.

A basic distinction between trajectory optimization methods is whether they are direct or indirect [27, 29]. Indirect methods formulate the optimality conditions of the optimal control problem (OCP) as a set of nonlinear equations which are then solved numerically [31, 32]. In contrast, direct methods discretize the OCP itself and then solve the nonlinear resulting program that corresponds to the discretized OCP [27, 33]. Direct methods possess several advantages over indirect methods that have led them to become the norm in trajectory optimization in the past decades[29]. Indirect methods require the formulation of the adjoint equations for the system under consideration which can require an expert user. Furthermore, if the OCP contains path inequality constraints,

indirect methods require a priori knowledge of when those constraints are active. Direct methods have neither of these restrictions. For these reasons, indirect methods are no longer widely used in trajectory optimization and therefore we will restrict our attention to direct methods for the remainder of this section.

One can also categorize trajectory optimization methods by the solution technique that they use. By "solution technique" we mean how the infinite dimensional OCP is reduced to a finite dimensional nonlinear program (NLP) and solved.

Shooting methods parameterize the differential equations in the OCP or BVP and use numerical integration of the ODEs (or DAEs) in the problem to evaluate any functions that depend on the state, control, and adjoint variable trajectories. In single-shooting, the parameters include only the initial conditions and whatever parameters are necessary for the control. Thus, the differential equations must be integrated over the entire time span. This produces an unfortunate numerical effect, wherein small variations in the control input at the beginning of the trajectory cause large variations in state in the latter parts of the trajectory. In an effort to avoid this phenomenon, multiple-shooting methods split the time span into segments, the initial conditions of which are added to the list of parameters. Additional constraints are introduced to ensure that the end-points of adjacent segments match. The resulting NLPs are larger than those for single-shooting, but are sparse and have significant structure that solvers can exploit [34].

Transcription methods can be thought of as the extension of multiple-shooting to the point where each segment contains exactly one integration step. The state and control trajectories are discretized, and their values at each knot-point form the decision variables of the NLP. The differential equations defining the system's dynamics are converted to constraints on these variables [29]. The form of these constraints differs according to the discretization scheme used. This discretization scheme can be thought of as the definition of the continuous trajectory associated with the discrete knot-points. Various discretization schemes, analogous to different integration methods, are employed in the trajectory optimization community [29]. In direct transcription methods, path constraints can be enforced directly on the state and control values at the knot-points.

While most of the theoretical underpinnings of trajectory optimization have their roots

in the aerospace community, there have been many applications to motion planning for legged robots. Mombaur [28] optimizes self-stable running for a planar humanoid model using direct multiple-shooting, while Schultz and Mombaur [35] optimize running gaits for a three-dimensional humanoid model with the same method. Remy, Buffinton, and Siegwart [36] use direct collocation to find energetically efficient gaits for planar hoppers. Some researchers have applied these techniques to humanoids, but with substantially simplified dynamics. El Khoury, Lamiraux, and Taïx [30] use direct multiple-shooting to generate minimum jerk trajectories for HRP-2. They treat the joint-space dynamics of the robot as a triple integrator and impose a constraint on the Zero Moment Point (ZMP). The planning techniques I present in Chapters 4 and 5 are direct transcription methods applied to specific simplifications of the robot model.

Simplified-model based approaches

The complications arising from the multi-body dynamics of legged robots can be mitigated by planning on a simplified model of the robot's dynamics, e.g. [37–40]. The best known approaches in this vein employ the concept of the Zero Moment Point (ZMP) — the point inside the robot's support polygon through which the resultant ground reaction force passes [41]. The ZMP can be used as a criterion to ensure dynamic balance during a robot's motion: as long as the ZMP exists (or equivalently, as long as the robot's center of pressure lies in the interior of its support polygon) the robot is dynamically-balanced [42]. Thus, dynamically-balanced motion plans can be synthesized by specifying a trajectory of the robot's ZMP that lies in the interior of its support polygon at all times. For example, Kajita et al. combine such ZMP trajectories with a simplified model of a robot's dynamics (the cart-table model) and preview control to generate a corresponding COM trajectory [37]. This COM trajectory is then converted to a full-body trajectory through inverse kinematics.

The simplified models used in these ZMP-based approaches treat the robot as a point mass and ignore angular momentum about the robot's center of mass, which limits the situations in which they can be employed. In response to this limitation, other criteria for dynamic balance have been proposed, such as maintaining the total wrench on the robot within the cone of wrenches that can be generated by the robot's contacts with the

environment [39]. This approach summarizes the robot's dynamic state by its momenta (linear and angular). While this and other approaches based on momentum [38, 40] allow for the generation of a wider class of motions, they still lack the ability to incorporate arbitrary kinematic constraints in the planning process, as the reference trajectories they output are for body-level properties only (momenta and center of mass positions). The method proposed in Chapter 5 can be seen as an augmentation of these momentum-based approaches to incorporate the full kinematics of the robot during planning.

2.2 All-at-once approaches

In contrast to the methods described in the previous section, methods following the second track in Fig. 2-1 do not separate the selection of end-effector placements and whole-body planning. Rather, they treat both aspects of the planning problem simultaneously [43, 44].

Posa, Cantu, and Tedrake [44] incorporate end-effector placement into the whole-body planning problem through complementarity constraints. The quintessential example of such a constraint is that the fact that the product of the distance between an end-effector and a support surface, ϕ, and the contact force, λ must always be zero:

$$\phi\lambda = 0.$$

With this and other similar constraints, Posa, Cantu, and Tedrake express the combined problem of whole-body planning with dynamics and end-effector placement as a single, constrained NLP. This allows them to plan walking and running motions without specifying the sequence of footfalls a priori.

Mordatch, Todorov, and Popović [43] take another approach and express the combined problem as an unconstrained NLP. They do this by allowing, but penalizing, contact forces between end-effectors and contact surfaces that are still separated. By pushing this and all other constraints (such as the robot's kinematics) into the objective function, they obtain a single, unconstrained NLP for the combined planning problem. They use this approach to plan a variety of dynamic maneuvers for humanoid and non-humanoid characters.

The downsides of these methods stem from the fact that they require the solution of large nonlinear programs. The size and non-convexity of these problems means that local methods are the only effective techniques for solving them. As a result, proper initialization of the optimization is very important — what answer the solver gives will depend on the initial guess, or seed, that the user supplies. In addition to the difficulties of working with large NLPs, this formulation presents the challenge of choosing appropriate weights for the various terms of the objective function. Additionally, local NLP solvers may return with an "infeasible" status in cases where the solver cannot make further progress, but where the actual problem is not infeasible. This makes interpreting the solver's results more difficult, as one is never entirely sure whether the issue is with the planning problem itself or merely with the seed. The unconstrained method presented by Mordatch, Todorov, and Popović [43] is further complicated by the fact that all constraints in the problem are replaced with terms in the objective function. This introduces additional parameters (weights between the different terms in the objective) that the user must tune.

Chapter 3

General Formulation of Legged Locomotion Planning Problem

In this chapter we develop a mathematical description of the sort of problem described in Section 1.1 in which a robot must move quickly through its environment, using different portions of the environment for support. This problem has several components:

- a model of the robot's kinematics
- a model of the robot's dynamics
- a model of the environment
- a cost function to define performance.

3.1 Kinematics of legged robots

The purpose of a kinematic model is to convert between joint angles and link positions. In this work we treat all legged robots as kinematic trees with a floating base. This lets us use a single model to represent all of the contact modes of the system, rather than having to formulate a separate model for each mode. For a robot with n_{joint} joints, the generalized position of the robot, $\mathbf{q}$, lies in the space $\text{SE}(3) \times \mathbb{R}^{n_{\text{joint}}}$, where SE(3) is the space of three-dimensional rigid body transformations. Let $\mathcal{Q} \subseteq \text{SE}(3) \times \mathbb{R}^{n_{\text{joint}}}$ be the set

of allowable positions. When dealing with algorithms that work on vector spaces, we will often use an alternative definition: $\mathbf{q} \in \mathcal{Q} \subseteq \mathbb{R}^{n_\mathbf{q}}$ where $n_\mathbf{q}$ is the total number of variables needed to describe the configuration of the robot.

The generalized velocity of a floating-base robot consists of the angular and linear velocity of the floating-base and the joint-velocity for each joint. We denote this as $\mathbf{v} \in \mathcal{V} \subseteq \mathbb{R}^{n_\mathbf{v}}$, where $n_\mathbf{v} = 6 + n_{\text{joint}}$ is the number of velocity variables and $\mathcal{V}$ is the set of allowable velocities.

The kinematics of such robots are well understood - see for example [45]. Each link has an associated coordinate frame. We denote the expression of a vector, $\mathbf{v} \in \mathbb{R}^3$, in coordinate frame A by a leading super script: ${}^{\text{A}}\mathbf{v}$. The transform of A relative to B, ${}^{\text{B}}T_{\text{A}}$ maps a vector expressed in frame A to the same vector expressed in frame B

$$ {}^{\text{B}}\mathbf{v} = {}^{\text{B}}T_{\text{A}} \, {}^{\text{A}}\mathbf{v}. $$

Each link is connected to its parent by a joint, which constrains the relative motion of the two links. Joints have associated generalized positions that parameterize that constraint. To find the transformation between the coordinate frames of arbitrary links A and B, one concatenates the joint transforms from link A up to the common ancestor of the two links and then back to link B. This defines the forward kinematics

$$ \text{FK}(\mathbf{q}, \text{A}, \text{B}) = {}^{\text{B}}T_{\text{A}}(\mathbf{q}) $$

The primary kinematic quantities of interest are the location of the robot's end-effectors. We denote these positions as $\mathbf{r}_i(\mathbf{q})$ for $i = 1, \ldots, n_{\text{ee}}$.

$$ \text{FK}_{\text{ee},i}(\mathbf{q}) = \text{FK}(\mathbf{q}, \text{A}_i, O) \, {}^{\text{A}_i}\mathbf{r}_i \tag{3.1} $$

where, A_i is the link to which the i-th end-effector belongs, O is the world link, and ${}^{\text{A}_i}\mathbf{r}_i$ is the fixed position of the i-th end-effector in body A_i.

3.2 Dynamics of a Floating-base Multi-body System

The dynamics of a floating-base robot can be expressed by the well-known manipulator equation

$$\mathbf{H}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{C}(\mathbf{q}, \mathbf{v}) = \mathbf{B}\mathbf{u} + \sum_{i=1}^{n_{\text{ee}}} \mathbf{J}_i(\mathbf{q})'\mathbf{f}_i, \tag{3.2}$$

where $\mathbf{q}$ is the generalized position vector, $\mathbf{v}$ is the vector of generalized velocities, and $\mathbf{u}$ is the vector of generalized inputs. $\mathbf{H}$ is the generalized mass-matrix, $\mathbf{C}$ is the joint-space bias force, composed of the joint-friction, Coriolis, and gravitational forces on the robot, and $\mathbf{B}$ maps the generalized inputs to joint torques. We consider n_{ee} end-effectors of the robot each of which has an associated external force, $\mathbf{f}_i$. In general, each end-effector is also subject to an external torque, but in this thesis we will consider point contacts, in which case that external torque is identically zero. The end-effector Jacobian, $\mathbf{J}_i$, which maps $\mathbf{f}_i$ into joint coordinates is defined by

$$\mathbf{J}_i(\mathbf{q}) = \begin{bmatrix} \frac{\partial \mathbf{r}_{i,1}}{\partial \mathbf{q}_1} & \cdots & \frac{\partial \mathbf{r}_{i,1}}{\partial \mathbf{q}_n} \\ \frac{\partial \mathbf{r}_{i,2}}{\partial \mathbf{q}_1} & \cdots & \frac{\partial \mathbf{r}_{i,2}}{\partial \mathbf{q}_n} \\ \frac{\partial \mathbf{r}_{i,3}}{\partial \mathbf{q}_1} & \cdots & \frac{\partial \mathbf{r}_{i,3}}{\partial \mathbf{q}_n} \end{bmatrix}$$

We can use efficient, recursive techniques to compute $\mathbf{H}$, $\mathbf{C}$, and $\mathbf{B}$ for any robot defined by a kinematic tree [46]. However, (3.2) is a nonlinear system of ordinary differential equations (ODEs) due to the complicated dependence of $\mathbf{H}$, $\mathbf{C}$ and $\mathbf{J}_{\text{ee}}$ on $\mathbf{q}$ and $\mathbf{v}$.

3.3 Environment model

Let the available contact regions in the environment be given by $\mathcal{R}_j \subset \mathbb{R}^3$, for $j = 1, \ldots n_{\text{contact}}$. Similarly, let the available obstacle-free regions in the environment be given by $\mathcal{R}_j \subseteq \mathbb{R}^3$, for $j = n_{\text{contact}} + 1, \ldots, n_{\text{regions}}$ where $n_{\text{regions}} = n_{\text{contact}} + n_{\text{free}}$. In this chapter, we place no restrictions on the $\mathcal{R}_j$ — they may be non-convex. Thus, in this general formulation of the problem, one could have a single region comprised of all the obstacle-free space, in which case n_{free} would be one. Given these definitions, $\bigcup_{j=1}^{n_{\text{regions}}} \mathcal{R}_j$ is the set of

all admissible positions for the robot's end-effectors — at any time, each end-effector must lie in at least one $\mathcal{R}_j$. Thus,

$$\mathbf{r}_i(t) \in \bigcup_{j=1}^{n_{\text{regions}}} \mathcal{R}_j \;\forall\, i \in \{1, \ldots, n_{\text{ee}}\}, t \in [0, T]. \tag{3.3}$$

3.4 Contact Constraints

Since (3.2) imposes no constraints on the end-effector forces, it is not, by itself, sufficient to specify the motion of the robot. In order to complete this picture, we must connect the end-effector forces and positions (3.2) and (3.3). To that end, consider the space of end-effector positions and forces together: $X = \mathbb{R}^3 \times \mathbb{R}^3$. Each $\mathcal{R}_j$ has an associated set $\mathcal{K}_j \subseteq \mathbb{R}^3$ which is the set of admissible forces for an end-effector that lies only in that region. For example, if $\mathcal{R}_j$ is a non-contact region, $\mathcal{K}_j = \{\mathbf{0}\}$, whereas if $\mathcal{R}_j$ represents a unilateral contact with normal, $\mathbf{n}$, and coefficient of friction, μ, $\mathcal{K}_j$ is the friction cone defined by

$$\mathcal{K}_j = \left\{ \mathbf{f} \in \mathbb{R}^3 \mid \|\mathbf{f}\| \leq \left(1 + \mu^2\right)^{\frac{1}{2}} \mathbf{f} \cdot \mathbf{n} \right\}. \tag{3.4}$$

Thus, for each j, we can define the set of positions and forces that are admissible for an end-effector contained in the j-th region as

$$\mathcal{S}_j = \mathcal{R}_j \times \mathcal{K}_j \subseteq X. \tag{3.5}$$

The full admissible set for end-effector positions and contact forces is given by $\bigcup_{j=1}^{n_{\text{regions}}} \mathcal{S}_j$. Therefore, the constraint that the contact forces be consistent with the robot's end-effector positions can be written as

$$(\mathbf{r}_i, \mathbf{f}_i) \in \bigcup_{j=1}^{n_{\text{regions}}} \mathcal{S}_j \;\forall\, i \in \{1, \ldots, n_{\text{ee}}\} \tag{3.6}$$

3.5 Legged robot planning problem

Let

$$\mathfrak{q}\colon [0, t_f] \to \mathbb{R}^{n_\mathfrak{q}}, \qquad \mathfrak{v}\colon [0, t_f] \to \mathbb{R}^{n_\mathfrak{v}}, \qquad \mathfrak{u}\colon [0, t_f] \to \mathbb{R}^{n_\mathfrak{u}},$$

and

$$\mathfrak{f}_i\colon [0, t_f] \to \mathbb{R}^3 \text{ for } i = 1, \ldots, n_{\text{ee}},$$

be the configuration, velocity, input and end-effector force trajectories respectively. Define the cost functional

$$\mathfrak{q}, \mathfrak{v}, \mathfrak{u}, \mathfrak{f}_1, \ldots, \mathfrak{f}_{n_{\text{ee}}} \mapsto J(\mathfrak{q}, \mathfrak{v}, \mathfrak{u}, \mathfrak{f}_1, \ldots, \mathfrak{f}_{n_{\text{ee}}}). \tag{3.7}$$

Then we can combine (3.2), (3.6), and (3.7) to yield a formulation of the legged robot dynamic motion planning problem:

$$\begin{aligned}
\underset{\mathfrak{q},\mathfrak{v},\mathfrak{u},\mathfrak{f}_i}{\text{minimize}} \quad & J(\mathfrak{q}, \mathfrak{v}, \mathfrak{u}, \mathfrak{f}_i) \\
\text{subject to} \quad & \left.\begin{aligned}
& \mathbf{H}(\mathfrak{q}(t))\dot{\mathfrak{v}}(t) + \mathbf{C}(\mathfrak{q}(t), \mathfrak{v}(t)) = \mathbf{B}\mathfrak{u}(t) + \sum_{i=1}^{n_{\text{ee}}} \mathbf{J}_i(\mathfrak{q}(t))' \mathfrak{f}_i(t) \\
& (\mathbf{r}_i(\mathfrak{q}(t)), \mathfrak{f}_i(t)) \in \bigcup_{j=1}^{n_{\text{regions}}} \mathcal{S}_j \;\forall\, i \in \{1, \ldots, n_{\text{ee}}\} \\
& \mathfrak{q}(t) \in \mathcal{Q} \\
& \mathfrak{v}(t) \in \mathcal{V} \\
& \mathfrak{u}(t) \in \mathcal{U}
\end{aligned} \right\} \forall\, t \in [0, T]
\end{aligned} \tag{P0}$$

We can apply existing trajectory optimization techniques to (P0), but the resulting problem is likely to be intractable because of the complexity of the constraint relating end-effector positions and forces, (3.6). To address this problem, I develop an approach that uses the combinatorial structure of (3.6) and a simplified model of the robot to formulate a relaxation of (P0) that can be solved through mixed-integer convex programming in Chapter 4.

Chapter 4

Mixed-Integer Convex Planning for Locomotion Over Irregular Terrain

Equation (3.3) specifies that each end-effector of our legged robot must always lie in the union of a collection of (possibly) disjoint regions. This is the combinatorial aspect of the planning problem — which region should each end-effector be in at each point in time? As discussed in Section 2.1.1, footstep-planning methods choose a sequence of positions for the stance-feet of the robot subject to kinematic constraints. Because these constraints do not consider the robot's dynamics, they must be conservative to ensure that the generated footstep plan can be achieved by the robot. In this chapter, I develop an approximation to (P0) that, given constraints on the initial and final position of the robot's COM, lets us solve for end-effector placements consistent with those constraints while also considering a key-subset of the robot's dynamics. This lets us address the combinatorial aspect of legged-locomotion planning in cases that require aggressive maneuvers, such as running or jumping. The results of the method I present here will then be used to guide whole-body planning as described in Chapter 6.

4.1 End-effector placement as a mixed integer constraint

My approach to this problem is inspired by Deits and Tedrake's [12] treatment of footstep planning for a humanoid robot. By segmenting the space of safe footholds into large

convex regions and then explicitly assigning each footstep to a particular region, they convert the nonlinear constraint

$$\mathbf{x} \in \bigcup_{r=1}^{R} \{\mathbf{x} \mid A_r \mathbf{x} \leq b_r\} \tag{4.1}$$

to the mixed-integer implications

$$H_r \implies A_r \mathbf{x} \leq b_r \text{ for } r = 1, \ldots, R \tag{4.2a}$$

$$\sum_{r=1}^{R} H_r = 1 \tag{4.2b}$$

$$H_r \in \{0, 1\} \; \forall \, r \in \{1, \ldots, R\} \tag{4.2c}$$

where the notation $H \implies c$ means that if $H = 1$, constraint c must hold. The binary variables $H_1, \ldots, H_R$ are called indicator variables — they indicate to which region the footstep belongs.

The constraints in (4.2) can be converted to a set of mixed-integer constraints through either a Big-M or convex-hull reformulation [47, 48]. The Big-M reformulation requires that $\mathbf{x}$ belong to a bounded set, X. It converts (4.2a) to

$$A_r \mathbf{x} \leq b_r + M_r \cdot (1 - H_r) \cdot \mathbb{1}^{m \times 1} \text{ for } r = 1, \ldots, R, \tag{4.3}$$

where $\mathbb{1}^{m \times 1}$ is an m-element column vector of all ones and M_r is a constant large enough that (4.3) holds for any $\mathbf{x} \in X$ when $H_r = 0$. In contrast, the convex-hull reformulation replaces (4.2a) with the constraints

$$\mathbf{x} = \sum_{r=1}^{R} \mathbf{x}_r \tag{4.4a}$$

$$A \mathbf{x}_r \leq H_r b_r \tag{4.4b}$$

$$H_r \mathbf{x}_{\text{lb}} \leq \mathbf{x}_r \leq H_r \mathbf{x}_{\text{ub}}. \tag{4.4c}$$

This formulation disaggregates $\mathbf{x}$ into the new auxiliary variables $\mathbf{x}_1, \ldots, \mathbf{x}_R$. Equa-

tions (4.4b) and (4.4c) constrain $\mathbf{x}_r$ either to be zero, if $H_r = 0$, or to lie in the r-th region, if $H_r = 1$. The Big-M approach yields a problem with fewer decision variables, but the convex-hull approach yields a tighter formulation [48]. Deits and Tedrake [12] use a Big-M reformulation, which yields a problem with fewer decision variables, but we will use the convex-hull reformulation later in this chapter, as it yields a tighter formulation and does not require selecting M_r values.

Deits and Tedrake then solve the footstep planning problem with a mixed-integer convex quadratically-constrained quadratic programming (MIQCQP) solver [12]. The ability to frame the problem as a mixed-integer *convex* program is significant, as solvers for mixed-integer non-convex programs have fewer theoretical guarantees and are computationally more expensive [49]. Deits and Tedrake can use a mixed-integer convex approach here because the safe regions were chosen to be convex, which makes the continuous relaxation of either (4.3) or (4.4) a set of convex constraints.

What we would like to do here is very much akin to footstep planning, indeed (4.1) is reminiscent of (3.3). We therefore set out to use a mixed-integer convex quadratic programming (MIQP) based approach to tackle the question of end-effector placement. Now, however, we need to consider not only the placement of each stance-foot, but also the position of all the robot's end-effectors at each point in time, as well as the external forces acting on those end-effectors. If we consider the robot's end-effector positions as decision variables rather than nonlinear functions of the robot's generalized position, then (3.3) becomes

$$(\mathbf{r}_i, \mathbf{f}_i) \in \bigcup_{j=1}^{n_{\text{regions}}} \mathcal{S}_j$$

If the $\mathcal{S}_j$ are convex, then the mixed-integer convex reformulations available for the footstep planning problem will also be available here. Recall that $\mathcal{S}_j = \mathcal{R}_j \times \mathcal{K}_j$, and therefore, $\mathcal{S}_j$ is convex if $\mathcal{R}_j$ and $\mathcal{K}_j$ are convex. The set of admissible forces for a free region, $\{\mathbf{0}\}$, and for a unilateral contact, (3.4), are both convex. Thus, for these classes of regions (as well as some others) the $\mathcal{K}_j$ are convex. Only the $\mathcal{R}_j$ are potentially non-convex. However, these regions were chosen as part of an initial segmentation step. If, as in Deits and Tedrake's [12] work, that segmentation is restricted to finding convex polyhedral regions, then the

$\mathcal{R}_j$ have the form

$$\mathcal{R}_j = \left\{ \mathbf{x} \in \mathbb{R}^3 \mid A_j \mathbf{x} \leq b_j \text{ where, } A_j \in \mathbb{R}^{m \times 3}, b_j \in \mathbb{R}^m \right\} \text{ for } j = 1, \ldots, n_{\text{regions}}.$$

Thus all the $\mathcal{S}_j$ are convex, and (3.3) is equivalent to the following disjunction over convex constraints

$$\bigvee_{j=1}^{n_{\text{regions}}} \left[(\mathbf{r}_i, \mathbf{f}_i) \in \mathcal{S}_j \right], \tag{4.5}$$

We can apply a convex-hull reformulation [47, 48] to (4.5) to convert it to constraints that can be added to an MIQP. Define the indicator variables $z_{ij} \in \{0, 1\}$. We will enforce that if $z_{ij} = 1$ then $(\mathbf{r}_i, \mathbf{f}_i) \in \mathcal{S}_j$. We introduce n_{regions} copies of $\mathbf{r}_i$ and $\mathbf{f}_i$ which we denote $\hat{\mathbf{r}}_{ij}$ and $\hat{\mathbf{f}}_{ij}$ respectively. These new decision variables are subject to the following constraints, which enforce the disjunction (4.5):

$$\left(\hat{\mathbf{r}}_{ij}, \hat{\mathbf{f}}_{ij}\right) \in \mathcal{S}_j \; \forall \, i \in \{1, \ldots, n_{\text{ee}}\}, \; j \in \{1, \ldots, n_{\text{regions}}\} \tag{4.6a}$$

$$z_{ij} \in \{0, 1\} \; \forall \, i \in \{1, \ldots, n_{\text{ee}}\}, \; j \in \{1, \ldots, n_{\text{regions}}\} \tag{4.6b}$$

$$\sum_{j=1}^{n_{\text{regions}}} \hat{\mathbf{r}}_{ij} = \mathbf{r}_i \; \forall \, i \in \{1, \ldots, n_{\text{ee}}\} \tag{4.6c}$$

$$\sum_{j=1}^{n_{\text{regions}}} \hat{\mathbf{f}}_{ij} = \mathbf{f}_i \; \forall \, i \in \{1, \ldots, n_{\text{ee}}\} \tag{4.6d}$$

$$\sum_{j=1}^{n_{\text{regions}}} z_{ij} = 1 \; \forall \, i \in \{1, \ldots, n_{\text{ee}}\} \tag{4.6e}$$

$$z_{ij} \mathbf{r}_{i,\text{lb}} \leq \hat{\mathbf{r}}_{ij} \leq z_{ij} \mathbf{r}_{i,\text{ub}} \; \forall \, i \in \{1, \ldots, n_{\text{ee}}\}, \; j \in \{1, \ldots, n_{\text{regions}}\} \tag{4.6f}$$

$$z_{ij} \mathbf{f}_{i,\text{lb}} \leq \hat{\mathbf{f}}_{ij} \leq z_{ij} \mathbf{f}_{i,\text{ub}} \; \forall \, i \in \{1, \ldots, n_{\text{ee}}\}, \; j \in \{1, \ldots, n_{\text{regions}}\} \tag{4.6g}$$

Note that these constraints to not preclude $(\mathbf{r}_i, \mathbf{f}_i)$ from lying in more than one region. Rather, they require that $(\mathbf{r}_i, \mathbf{f}_i)$ satisfy the constraints for *at least* one region. We can add basic collision avoidance for an end-effector by additionally requiring that the line segment defined by the end-effector's position at adjacent knot-points lie entirely within one region. This is similar to the approach used by Deits and Tedrake [50] for collision

avoidance with un-manned aerial vehicles.

4.2 MIQP formulation of dynamic motion planning

We showed in Section 4.1 that the constraints on end-effector position and forces could be formulated as part of an MIQP. However, for a quickly moving robot, which sequences of footsteps are feasible depends not only on the distribution of potential support surfaces and the robot's kinematics but also on its dynamics — the robot must be able to reach the next footstep at the time it is expected to make contact. The dynamics of a legged robot, (3.2), are decidedly non-convex. Our goal, therefore, is to include enough of the dynamics for our planning problem to yield reasonable end-effector placements while ensuring that it remains an MIQP.

4.2.1 Centroidal Dynamics

We first need to find an approximation of the robot's dynamics that fits into the MIQP framework. The mass-matrix, $\mathbf{H}$, and bias force, $\mathbf{C}$, both contain trigonometric functions of $\mathbf{q}$, and therefore cannot be used. However, (3.2) is not the only available representation of the robot's dynamics. An alternative that captures key elements of the robot's dynamics is its centroidal dynamics, that is, the evolution of its linear and centroidal angular momentum over time. This captures the relationship between the momenta of the whole-robot system and the external forces and moments acting on that system. The centroidal momentum of a multi-body system is a vector, $\mathbf{h}_G \in \mathbb{R}^6$ representing the system's linear momentum and angular momentum about its center of mass, expressed in an inertial frame O.

$$\mathbf{h}_G = \begin{bmatrix} {}^O\mathbf{k} \\ {}^O\boldsymbol{\ell} \end{bmatrix} \tag{4.7}$$

While the multi-body dynamics of a legged robot are complicated, the centroidal dynamics of that system are quite simple. Newton's Second Law applied to the robot as a whole

yields

$$\dot{\boldsymbol{\ell}} = \mathbf{f}_{\text{ext}} \tag{4.8}$$

$$\dot{\mathbf{k}} = \mathbf{T}_{\text{ext}}, \tag{4.9}$$

where $\mathbf{T}_{\text{ext}}$ is the total external moment exerted on the system by its environment. In the framework presented in Chapter 3, $\mathbf{f}_{\text{ext}}$ and $\mathbf{T}_{\text{ext}}$ can be expanded to yield

$$\dot{\boldsymbol{\ell}} = \sum_i \mathbf{f}_i + m\mathbf{g} \tag{4.10a}$$

$$\dot{\mathbf{k}} = \sum_i (\mathbf{r}_i - \mathbf{r}) \times \mathbf{f}_i. \tag{4.10b}$$

The definition of linear momentum allows us to rewrite (4.10a) as

$$\ddot{\mathbf{r}} = \frac{1}{m} \sum_i \mathbf{f}_i + \mathbf{g}. \tag{4.11}$$

Thus, the translational portion of the robot's centroidal dynamics is linear.

The rotational portion is more problematic. The cross product in (4.10b) expands to

$$(\mathbf{r}_i - \mathbf{r}) \times \mathbf{f}_i = \begin{bmatrix} (r_{i,y} - r_y) f_{i,z} - (r_{i,z} - r_z) f_{i,y} \\ (r_{i,z} - r_z) f_{i,x} - (r_{i,x} - r_x) f_{i,z} \\ (r_{i,x} - r_x) f_{i,y} - (r_{i,y} - r_y) f_{i,x} \end{bmatrix}. \tag{4.12}$$

The bilinear terms in (4.12) are non-convex. We can reduce the number of such terms by introducing new variables, $\mathbf{s}_i$, that encode the positions of the end-effectors relative to the robot's COM:

$$\mathbf{s}_i = \mathbf{r}_i - \mathbf{r} \; \forall \, i \in \{1, \ldots, n_{\text{ee}}\}. \tag{4.13}$$

This still leaves us with six bilinear terms. Fortunately, there are multiple techniques available for approximating bilinear terms within an MIQP [51–53]. We will discuss some of these approximations in Section 4.3. For now we simply introduce the approximating

variables, w:

$$\begin{aligned} w_{i,yz} &\approx s_{i,y} f_{i,z} \quad w_{i,zx} \approx s_{i,z} f_{i,x} \quad w_{i,xy} \approx s_{i,x} f_{i,y} \\ w_{i,zy} &\approx s_{i,z} f_{i,y} \quad w_{i,xz} \approx s_{i,x} f_{i,z} \quad w_{i,yx} \approx s_{i,y} f_{i,x}. \end{aligned} \tag{4.14}$$

Then the (4.10b) can be modified to yield the approximate rotational dynamics

$$\dot{\mathbf{k}} = \begin{bmatrix} w_{i,yz} - w_{i,zy} \\ w_{i,zx} - w_{i,xz} \\ w_{i,xy} - w_{i,yx} \end{bmatrix} + \boldsymbol{\tau}_i. \tag{4.15}$$

Thus, (4.11), (4.14), and (4.15) form a mixed-integer convex approximation of the centroidal dynamics of the robot.

4.2.2 Planar single-body model

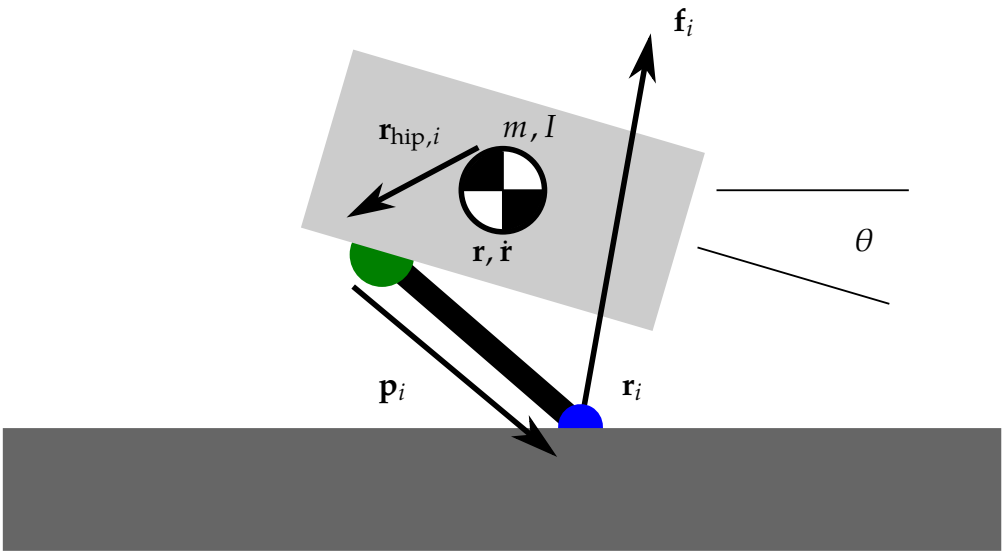


Figure 4-1: Schematic of the simplified planar model

Since we would like to reason about the position of the end-effectors, we require some notion of the orientation of the robot. Unfortunately, there is no rotational analog to the COM. To inform our search over rotations we treat the robot as a single rigid body, with

moment of intertia $\mathbf{I}$ and orientation $R \in \mathrm{SO}(3)$. For this system, (4.15) becomes

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1} \begin{bmatrix} w_{i,yz} - w_{i,zy} \\ w_{i,zx} - w_{i,xz} \\ w_{i,xy} - w_{i,yx} \end{bmatrix} + \boldsymbol{\tau}_i.$$

where $\boldsymbol{\omega}$ is the angular velocity of the rigid body expressed in body frame.

The relation between $\boldsymbol{\omega}$ and R is given by

$$\dot{R} = R\hat{\boldsymbol{\omega}} \tag{4.16}$$

where $\hat{\boldsymbol{\omega}}$ is the skew symmetric matrix

$$\hat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

From (4.16), we can see that including three-dimensional rotations in our program will add at least eighteen bilinear terms per knot-point. In fact, there will be even more, to constrain the end-effector positions relative to the body, we must compute the product $R^{-1}\mathbf{s}_i$, which yields an additional $9n_{\text{ee}}$ bilinear terms. As we will see in Section 4.3, the number of auxiliary integer variables increases with the number bilinear terms. This in turn greatly increases the computational effort required to solve the MIQP. Therefore, for our initial exploration of this approach we will consider the case where the robot is constrained to the xz-plane.

Figure 4-1 shows this model for $n_{\text{ee}} = 1$. In this case, (4.10b) becomes

$$\ddot{\theta} = \frac{1}{I} \sum_i s_{i,z} f_{i,x} - s_{i,x} f_{i,z}, \tag{4.17}$$

which is the rotational analog to (4.11).

To express approximate kinematic constraints on the end-effectors, which depend on θ, we introduce introduce auxiliary variables c_θ and s_θ, which are piecewise linear

approximations of $\cos\theta$ and $\sin\theta$ respectively. This is the same approach that Deits and Tedrake [12] take for approximating rotations. We also introduce $\mathbf{r}_{\text{hip},i}$ and $\mathbf{p}_i$ for $i = 1, \ldots, n_{\text{ee}}$. $\mathbf{r}_{\text{hip},i}$ is the approximate position of the attachment point of the i-th limb relative to the COM in world coordinates. It is given by

$$\mathbf{r}_{\text{hip},i} = \begin{bmatrix} {}^{B}r_{\text{hip},i,x} & {}^{B}r_{hip,i,z} \\ {}^{B}r_{\text{hip},i,z} & -{}^{B}r_{\text{hip},i,x} \end{bmatrix} \begin{bmatrix} c_\theta \\ s_\theta \end{bmatrix},$$

where ${}^{B}\mathbf{r}_{\text{hip},i}$ is the fixed position of the attachment point in the robot's body frame. $\mathbf{p}_i$ is the position of the i-th end-effector relative to the attachment point for the i-th limb in world coordinates. The approximate kinematic constraints on end-effector position are then given by bounds on the elements of $\mathbf{p}_i$ and the linear constraints

$$\mathbf{r}_i = \mathbf{r} + \mathbf{r}_{\text{hip},i} + \mathbf{p}_i.$$

Discretizing (4.11) and (4.17) with a backwards Euler scheme and a fixed time-step, h, gives the dynamic constraints for our simplified model.

$$\mathbf{r}[n+1] = \mathbf{r}[n] + h\dot{\mathbf{r}}[n+1] \tag{4.18a}$$

$$\theta[n+1] = \theta[n] + h\omega[n+1] \tag{4.18b}$$

$$\dot{\mathbf{r}}[n+1] = \dot{\mathbf{r}}[n] + h\mathbf{F}[n+1] \tag{4.18c}$$

$$\omega[n+1] = \omega[n] + \frac{h}{I}T[n+1] \tag{4.18d}$$

$$\mathbf{F}[n] = m\mathbf{g} + \sum_i \mathbf{f}_i[n] \tag{4.18e}$$

$$T[n] = \sum_i w_{i,zx}[n] - w_{i,xz}[n] \tag{4.18f}$$

Note that (4.18) is a set of linear equality constraints, which can be incorporated into an MIQP. We now address how to treat the approximations to the bilinear terms, w.

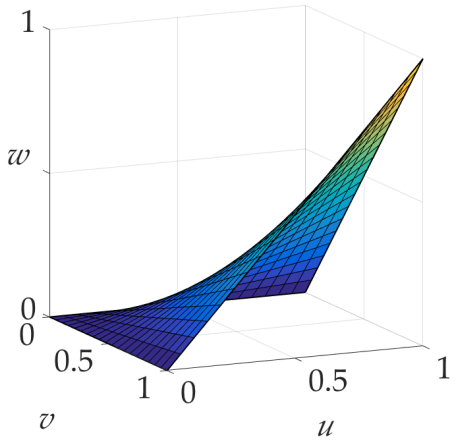


Figure 4-2: Surface defined by $w = uv$

4.3 Convex relaxation of bilinear terms

Optimization problems containing bilinear terms (bilinear programs) occur in a variety of real world applications, such as refinery optimization [54], process networks [55], bidimensional packing [56], and computer vision [57]. As a result, many schemes for generating convex (or MI-convex) relaxations can be found in the literature [51–53]. In this chapter we will compare two such techniques: piecewise McCormick envelopes and the multi-parametric disaggregation technique (MDT).

We describe these relaxations in the context of a single bilinear term

$$w = uv \tag{4.19}$$

over the domain [0, 1] x [0, 1]. Through an appropriate linear transformation results for that domain can be used for any bilinear term. Figure 4-2 shows the three dimensional surface defined by (4.19).

4.3.1 Piecewise McCormick Envelopes

The simplest convex relaxation for (4.19) is the constraint that (w, u, v) lie within the convex-hull of the surface shown in Fig. 4-2. This can be expressed as

$$
\begin{aligned}
w &\leq v \\
w &\leq u \\
w &\geq u + v - 1 \\
u, v, u &\in [0, 1]
\end{aligned}
\tag{4.20}
$$

This is the McCormick envelope of (4.20) as presented by McCormick [51]. It is shown in Fig. 4-3a

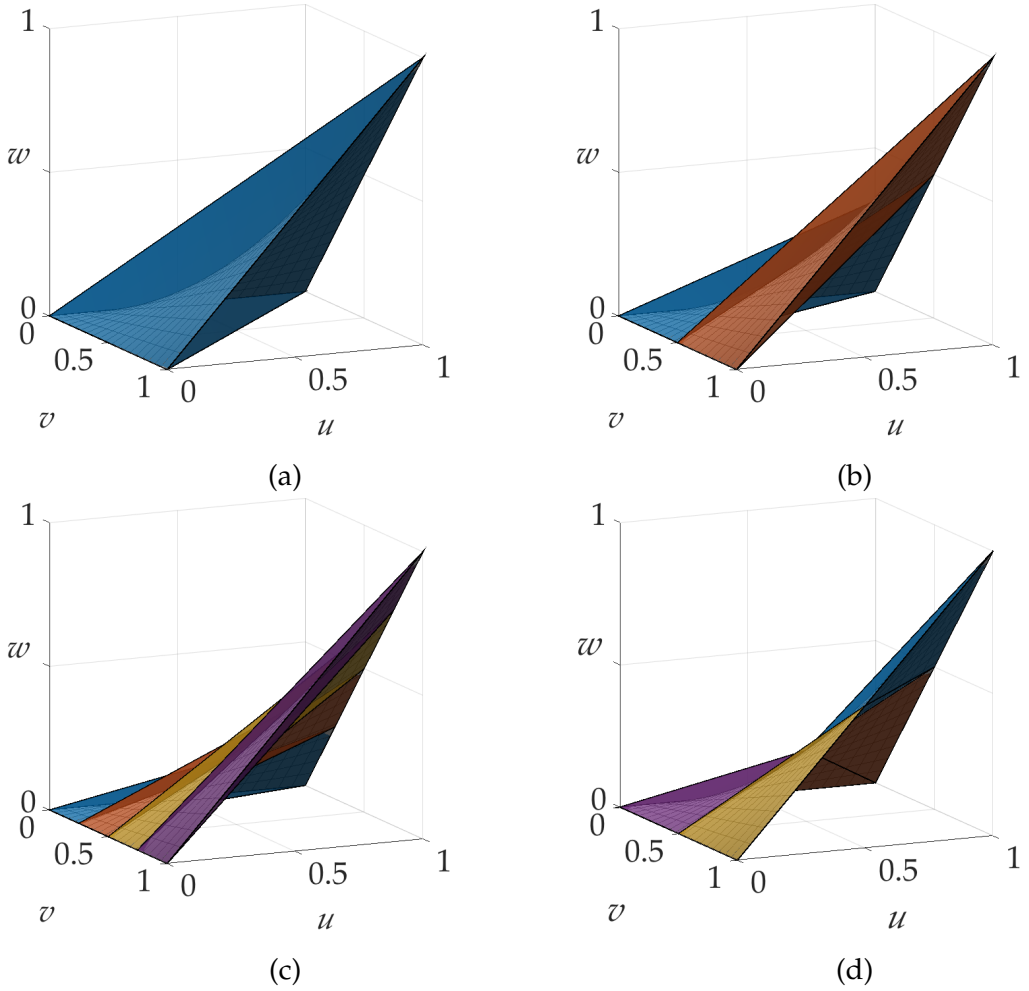


Figure 4-3: MI-convex approximations of $w = uv$: (a) McCormick envelope. (b) Piecewise McCormick envelope ($M = 2$). (c) Piecewise McCormick envelope ($M = 4$). (d) Bivariate piecewise McCormick envelope ($M = 4$)

If one is willing to use a mixed-integer convex relaxation, then a tighter relaxation may be obtained by partitioning the domain. Let M be the number of segments in a

partitioning of u and let $0 = u_0 \leq \cdots u_m \leq \cdots \leq u_M = 1$ define that partitioning. Then, for each partition we can define a set of linear constraints that are satisfied iff (u, v, w) lies within the convex-hull of w=uv over that partition. Our mixed integer convex relaxation over the entire interval is given by the disjunction:

$$\bigvee_{m=1}^{M} \begin{bmatrix} w \geq u_m^L \cdot v \\ w \geq u_m^U \cdot v + u - u_m^U \\ w \leq u_m^L \cdot v + u - u_m^L \\ w \leq u_m^U \cdot v \\ u_m^L \leq u \leq u_m^U \end{bmatrix} \tag{4.21}$$

This can be rewritten using a convex-hull reformulation [47] to yield

$$w \geq \sum_{m=1}^{M} u_m^L \cdot \hat{v}_m \tag{4.22a}$$

$$w \geq \sum_{m=1}^{M} u_m^U \cdot \hat{v}_m + \hat{u}_m - u_m^U \tag{4.22b}$$

$$w \leq \sum_{m=1}^{M} u_m^L \cdot \hat{v}_m + \hat{u}_m - u_m^L \tag{4.22c}$$

$$w \leq \sum_{m=1}^{M} u_m^U \cdot \hat{v}_m \tag{4.22d}$$

$$u = \sum_{m=1}^{M} \hat{u}_M \tag{4.22e}$$

$$v = \sum_{m=1}^{M} \hat{v}_M \tag{4.22f}$$

$$0 \leq \hat{v}_m \leq z_m \tag{4.22g}$$

$$z_m \cdot u_m^L \leq \hat{u}_m \leq z_m \cdot u_m^U \tag{4.22h}$$

where the binary variable z_i is true iff (u, v, w) lies in the i-th partition. Equation (4.21) defines a univariate partitioning, but we can also form a bivariate partitioning by splitting

v as well. An example of the regions resulting from such a partitioning, with M = 2 is shown in Fig. 4-3d. The number of binary variables required by this approximation grows exponentially with the power of the desired accuracy [52].

4.3.2 Multiparametric Disaggregation Technique

In order to improve the scaling between the maximum error and the number of binary variables required by a convex relaxation of a bilinear constraint, researchers have introduced other relaxations. The multiparametric disaggregation technique (MDT) [52, 58] discretizes one variable, but not by dividing it into segments. Rather, the variable is discretized by considering its representation in a given number system. Our presentation of this approach follows the treatment of Kolodziej, Castro, and Grossmann [52]. Let $b \in \mathbb{Z}, b \geq 0$ be the base. Then v can be written as the following double sum

$$v = \sum_{l \in \mathbb{Z}} \sum_{k=0}^{b-1} k \cdot b^l \cdot z_{kl} \tag{4.23}$$

where

$$z_{kl} \in \{0,1\} \; \forall \, k \in 0, \ldots, b-1, l \in \mathbb{Z}$$
$$\sum_{k=0}^{b-1} z_{kl} = 1 \; \forall \, l \in \mathbb{Z}$$

Equation (4.23) is exact because the outer sum runs over all the integers. Clearly we cannot implement such a representation in practice. However, we can represent any v up to an arbitrary level of precision simply by choosing bounds on l. This approximation is given by

$$v' = \sum_{l=p}^{P} \sum_{k=0}^{b-1} k \cdot b^l \cdot z_{kl} \tag{4.24}$$

where

$$z_{kl} \in \{0,1\} \; \forall \, k \in \{0,\dots,b-1\}, l \in \{p,\dots,P\}$$
$$\sum_{k=0}^{b-1} z_{kl} = 1 \; \forall \, l \in \{p,\dots,P\}$$

The representation in (4.24) is discrete - all potential values of v are separated by at least b^p. We can regain a continuous representation of v by adding a bounded continuous term, which yields

$$v^R = \sum_{l=p}^{P} \sum_{k=0}^{b-1} k \cdot b^l \cdot z_{kl} + \Delta v \tag{4.25}$$

$$0 \leq \Delta v \leq b^p \tag{4.26}$$

Now we can rewrite (4.19) as

$$w^R = \sum_{l=p}^{P} \sum_{k=0}^{b-1} u \cdot k \cdot b^l \cdot z_{kl} + u \cdot \Delta v. \tag{4.27}$$

Let $\hat{u}_{k,l} = u \cdot z_{k,l}$. Then

$$u = \sum_{k=1}^{b-1} \hat{u}_{k,l} \tag{4.28}$$

Substituting $\hat{u}_{k,l}$ into (4.27) yields

$$w^R = \sum_{l=p}^{P} \sum_{k=0}^{b-1} k \cdot b^l \cdot \hat{u}_{k,l} + u \cdot \Delta v. \tag{4.29}$$

We now bound the final term of (4.27) with its McCormick envelope:

$$\begin{aligned} 0 \leq \Delta w \leq \Delta v \\ b^p \cdot (u-1) + \Delta v \leq \Delta w \leq b^p \cdot u \end{aligned} \tag{4.30}$$

From its definition, the bounds on $\hat{u}_{k,l}$, are

$$0 \leq \hat{u}_{k,l} \leq z_{k,l} \tag{4.31}$$

Equations (4.28), (4.30), and (4.31) give the MDT approximation of the original bilinear term. The number of binary variables required for an MDT relaxation grows linearly with the power of the desired accuracy [52].

4.3.3 Discussion

Which approximation is best depends on a two primary factors: (1) the accuracy required and (2) the number of bilinear terms. In general, one wants to introduce as few binary variables as possible, while still maintaining sufficient accuracy. For the problem described in this chapter, an approximation is sufficiently accurate if the solution to the problem provides a useful seed to the whole body trajectory optimization that follows. I have evaluated all of the approximations discussed in Section 4.3 and found that even the simplest — the McCormick envelope — yields feasible end-effector placements. Therefore, the results presented in Section 4.6 use this approximation.

4.4 Cost function and full MIQP

I use the following cost function for this optimization:

$$J = \sum_{i=1}^{n_{\text{ee}}} \sum_{j=1}^{n_{\text{contact}}} \Gamma_{ij} + \sum_{n=1}^{N} \left(\|\mathbf{f}_i[n]\|^2 + \|\ddot{\mathbf{r}}_i[n]\|^2 \right)$$

where $\Gamma_{i,j}$ is an upper bound on the number of times the indicator variable for end-effector i and region j changes value. This is implemented with slack variables $\gamma_{ij}[n]$ as

$$\begin{aligned} \Gamma_{ij} &= \sum_{n=1}^{N-1} \gamma_{ij}[n] \\ -\gamma_{ij}[n] \leq z_{ij}[n+1] - z_{ij}[n] &\leq \gamma_{ij}[n] \\ \gamma_{ij}[n] &\in [0,1]. \end{aligned}$$

Minimizing $\sum \Gamma_{ij}$ penalizes frequent switching in and out of contact. The remaining terms penalize contact force and foot acceleration.

Thus, the mixed-integer convex formulation of the dynamic motion planning problem for legged robots is given by

minimize	J	(P1)
subject to	(4.6) and (4.18)	
	one of the approximations for bilinear terms described in Section 4.3	
	task-specific constraints	

where the task-specific constraints are any linear, or convex quadratic constraints on the decision variables, such as, for example, initial/final state constraints, average velocity constraints, or maximum force constraints. The primary choice left to the user is the time discretization. Equation (4.18) depends on the choice of time-step, or alternatively the total duration, as well as the number of knot-points, N.

4.5 Tools and implementation

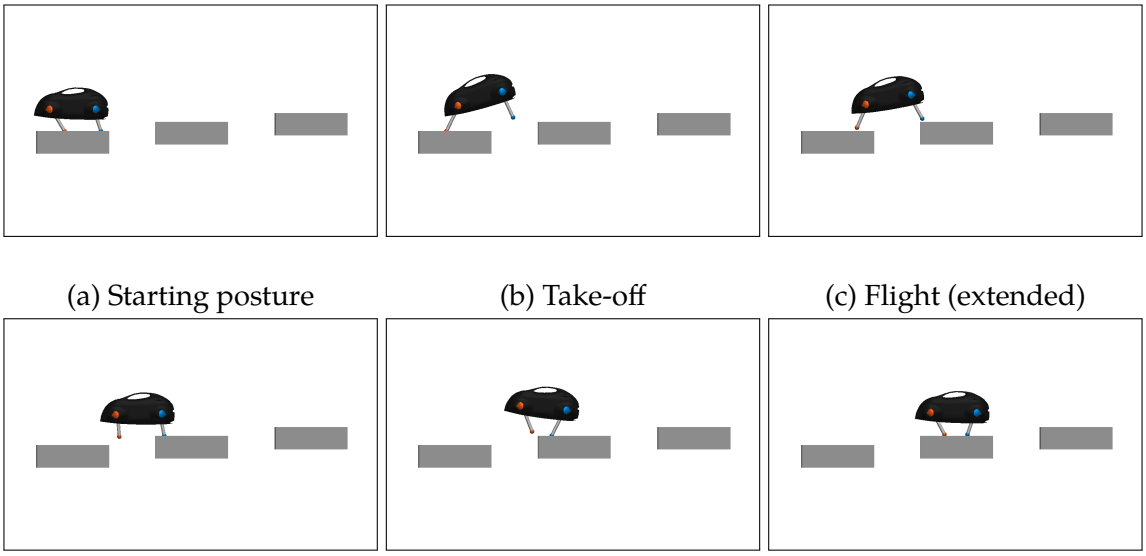
I have implemented (P1) in the Python optimization framework Pyomo [59, 60]. Pyomo provides a rich symbolic environment for formulating mathematical programs as well as interfaces to many commercial and open-source solvers. Pyomo also has some utilities for handling disjunctions built-in, as well as an implementation of the multiparametric dissaggregation technique. I used the mixed-integer quadratic programming solver

Gurobi [61] as a backend to solve the optimization problems.

4.6 Results

4.6.1 Separated Platforms

We apply our mixed-integer planning approach to the challenging problem of quadrupedal bounding over uneven terrain with gaps. The robot starts at the left end of a set of three platforms and must cross to the right end. This is encoded as constraints on the initial and final position of the robot's COM. The top of each platform is a contact region, while the space above each platform and extending to the edge of the adjacent platform is a free region. Thus $n_{\text{regions}} = 6$ for these examples. Fig. 4-4 shows snapshots from a motion plan for a set of ascending platforms. Figure 4-4 shows key frames from the first half of a plan



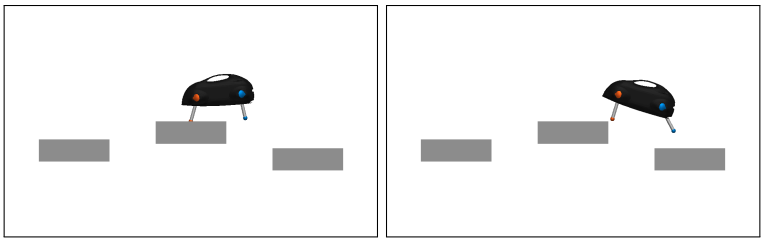
(a) Starting posture (b) Take-off (c) Flight (extended)

(d) Touch-down (e) Flight (gathered) (f) Touch-down

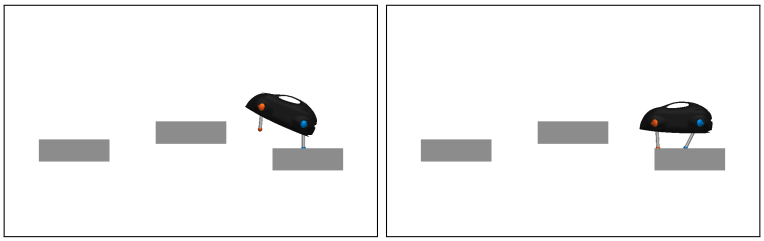
Figure 4-4: Snapshots from a motion plan for ascending platforms with gaps

to cross a set of ascending platforms. In Figs. 4-4c and 4-4e we see that the plan has the robot execute multiple aerial phases to complete this maneuver.

Figure 4-5 shows frames from the second half of a motion plan that traverses a set of staggered platforms. We will revisit both of these scenarios when we combine our



(a) Take-off (b) Flight (extended)



(c) Touch-down (d) Final posture

Figure 4-5: Snapshots from a motion plan for staggered platforms with gaps

MIQP-based planner with a whole body planner in Chapter 6.

4.6.2 Comparing MIQP and MINCP solutions

I attempted to compare the approach to a state of the art mixed-integer non-convex solver, COUENNE [62]. However, even for a small problem, comprising the first half of the terrain shown in Fig. 4-4, COUENNE was unable to find a feasible solution in an hour, while our MIQP-based approach solved to optimality in 12 s. This serves to underscore the performance advantages of using an MIQP-based approach.

4.6.3 Discussion

Qualitatively, working with an MIQP formulation makes solving problems such as these more straightforward than it would be if an NLP formulation were used. Simply knowing that the problem actually is infeasible if the solver declares it to be so eliminates much of the tweaking required in solving trajectory optimization problems as NLPs.

The solution times for problems similar to those discussed in Section 4.6.1 were generally a few minutes. In most cases, a feasible solution was found in the first two minutes.

As the number of contact and free-space regions increases, so do the solution times. For more complicated terrains with upwards of 20 regions, solution can take up to half an hour.

As mentioned in Section 4.4, the total duration of the motion and the number of knot-points are user selected quantities. For the examples in this section, I chose $N = 25$ and then increased the time interval between knot-points until the problem became feasible. This process was made much easier by the guarantees of the solver's feasibility reporting, which made it straightforward to tell whether a time interval value ought to be rejected. As a result, the search over time interval durations could be automated if a particular application required tuning that value without user input.

The segmentation of the space of admissible end-effector positions also needs to be specified prior to setting up the MIQP. This can have a significant effect on the solution time, or even the feasibility, of the problem. Larger numbers of regions can provide better coverage of the available space, but yield programs with more integer variables, which generally results in longer solve times. An automated method for selecting these regions would be ideal. While such a method is outside the scope of this thesis, I did perform initial evaluations of methods employing the IRIS algorithm [63], which efficiently computes large convex regions of obstacle-free space. It can be applied to the ground surface to yield safe contact regions and can also be applied to the free-space to yield free-space regions. These evaluations suggest that this is a viable path to automated segmentation.

Chapter 5

Centroidal-Dynamics Full-Kinematics Planning

5.1 A middle ground

As noted in Chapter 2, most approaches to dynamic motion planing for legged robots fall into one of two camps: (a) trajectory optimization on full robot models and (b) planning techniques that operate on greatly simplified models, such as the linear inverted-pendulum, or the spring-loaded inverted pendulum. The insight that motivates this chapter is that both the trajectory optimization based approaches and the simple model based approaches can be written as nonlinear programs. This means that we can view them not as two opposing methodologies, but rather as opposite ends of a whole spectrum of planning methods. In this chapter we develop an alternative (which we first presented in [64]) that occupies a powerful middle ground — planning with a model that combines the simple centroidal dynamics of the robot with its full kinematics.

5.2 Centroidal Dynamics of a Floating Base System

As in Section 4.2.1, we use the centroidal dynamics of the robot here, rather than the joint space dynamics. Recall that (4.10) does not explicitly include the joint positions $\mathbf{q}$. However, the centroidal angular momentum and linear momentum can both be written

in terms of the joint positions and velocities through the use of the centroidal momentum matrix (CMM), $\mathbf{A}_G$,

$$\mathbf{h}_G = \mathbf{A}_G(\mathbf{q})\mathbf{v} \tag{5.1}$$

where $\mathbf{A}_G$ may be computed as described by Orin and Goswami [65].

The validity of (4.10a) and (4.10b) at all times is a necessary condition for a given trajectory of a robot to be dynamically feasible. If a system has sufficient control authority it can also be a sufficient condition. In this case, "sufficient control authority" requires that all joints of the robot be actuated and that the torque limits on those actuators be large. In that case, given a joint acceleration there is always a set of joint torques that achieves those accelerations. As a result, the joint-level dynamics may be put aside during the planning phase. As long as the generated joint trajectory and centroidal momentum trajectory satisfy (4.10) and (5.1) there will be joint torques such that that joint trajectory is dynamically feasible.

5.3 Full-Kinematic Model and Constraints

Our kinematic model for the robot is the same as that presented in Section 3.1. Because of this, we can add any constraint that can be expressed as a differentiable function of link positions and orientations. This encompasses a rich variety of constraints, the principal ones of which are described below

5.3.1 Position constraints

A *relative-position constraint* restricts the position of a point on one body relative to a coordinate frame on another body to lie within some bounds. Given a point on body A, ${}^{A}\mathbf{r}$, and a frame B′ attached to body B, a relative-position constraint on ${}^{A}\mathbf{r}$ relative to B′ is given by

$$\mathbf{r}_{lb} \leq {}^{B'}T_B\, {}^{B}T_A(\mathbf{q})\, {}^{A}\mathbf{r} \leq \mathbf{r}_{ub}, \tag{5.2}$$

where $\mathbf{r}_{lb}$ and $\mathbf{r}_{ub}$ are lower and upper bounds on the relative-position respectively and the inequalities are applied element wise. Equation (5.2) can represent an absolute-position

constraint if B is taken as the world and ${}^{B'}T_B = \text{Id}$, where Id is the identity transform.

5.3.2 Orientation constraints

A *relative orientation constraint* requires that the rotation between the body-reference frames of bodies A and B be within some angular tolerance of a given rotation. This allows for constraints such as "align the robot's hand with the drill to within 5°." If $q_{AB}(\mathbf{q})$ is the quaternion representing the rotation of body A with respect to body B and q_d is the desired rotation, then the relative orientation constraint is given by

$$\cos(\theta) \leq 2\langle q_{AB}, q_d \rangle^2 - 1 \leq 1$$

5.3.3 Gaze constraints

A *relative-gaze direction constraint* specifies that a vector fixed in one body be aligned with a vector fixed in a second body to within a given tolerance angle. Given a unit vector expressed in the body-reference frame of body A, ${}^A\hat{\mathbf{u}}$, and a unit vector expressed in the body-reference frame of body B, ${}^B\hat{\mathbf{v}}$, the *relative-gaze direction constraint* between ${}^A\hat{\mathbf{u}}$ and ${}^B\hat{\mathbf{v}}$ is given by

$$\left\langle {}^B\hat{\mathbf{v}}, {}^BT_A\, {}^A\hat{\mathbf{u}} \right\rangle \leq \cos(\theta),$$

where θ is the tolerance angle.

A *relative-gaze target constraint* specifies that an axis on one link of the robot points towards a target on another link or in the environment. Specifically it requires that a cone with a given half-angle, fixed at a point on one body and pointing in a fixed direction relative to that body contain a point on a different body. This allows us to encode constraints such as "the robot's main camera must point at its left hand for the duration of the task. If ${}^A\mathbf{r}$ and ${}^A\hat{\mathbf{v}}$ are vectors fixed in the BRF of body A and ${}^B\mathbf{p}$ is a point fixed on body B, then the relative-gaze target constraint is given by

$$\left\langle {}^A\hat{\mathbf{v}}, \frac{{}^AT_B(\mathbf{q})\, {}^B\mathbf{p} - {}^A\mathbf{r}}{\left\| {}^AT_B(\mathbf{q})\, {}^B\mathbf{p} - {}^A\mathbf{r} \right\|} \right\rangle \leq \cos(\theta).$$

5.3.4 Collision avoidance constraints

One of the more complex kinematic constraints on the motion of a legged robot is that the motion be collision free. Our collision model consists of n_{elem} convex collision geometries, each of which is attached to the world or one of the robot's links at a known transform. Let $\mathrm{d}_{ij}(\mathbf{q})$ denote the minimum distance between the i-th and j-th collision geometries for the configuration vector $\mathbf{q}$. The distance between two collision geometries can be efficiently computed for many classes of convex geometries with the Gilbert-Johnson-Keerthi algorithm (GJK) [66]. In this work, we use the implementation of GJK in the Bullet Physics Software Development Kit [67]. Let d_{min} denote the minimum allowable distance between any pair of collision geometries, and let $\bar{\mathrm{d}}_{ij}(\mathbf{q})$ be given by

$$\bar{\mathrm{d}}_{ij}(\mathbf{q}) = \mathrm{d}_{ij}(\mathbf{q}) - d_{\text{min}}.$$

Thus, we wish to enforce that

$$\bar{\mathrm{d}}_{ij}(\mathbf{q}) \geq 0 \; \forall \, (i, j) \in P \tag{5.3}$$

where $P \subset \{1, \ldots, n_{\text{elem}}\} \times \{1, \ldots, n_{\text{elem}}\}$ is the set of index pairs that correspond to pairs of potentially colliding geometries. The number of potential collision pairs grows with the square of the number of collision geometries. In order to decrease the number of collision avoidance constraints that must be added to the trajectory optimization, we can combine all of the collision pairs using a hinge-loss-like function. Schulman et al. use a true hinge-loss function for a similar purpose [68]. Here we use a smooth function $\gamma(x)$ that is identically zero for all positive x, greater than zero for all negative x, and whose slope approaches a negative constant asymptotically as x goes to negative infinity:

$$\gamma(x) = \begin{cases} 0 & x \geq 0 \\ -x \exp\left(\frac{1}{x}\right) & x < 0 \end{cases}$$

This function has the advantage of being infinitely differentiable for all x. The overall collision constraint is given by

$$\Gamma(\mathbf{q}) = \sum_{(i,j)\in P_{ij}} \gamma\big(c\,\mathrm{d}_{ij}(\mathbf{q})\big) = 0, \tag{5.4}$$

where c is a positive scaling factor. In the examples shown in this chapter, c was taken to be $\frac{1}{d_{\min}}$. Since each term of the sum in (5.4) is non-negative, (5.4) holds if and only if all terms of that sum are zero, which in turn implies that (5.3) holds.

5.4 Trajectory Optimization

To compute a feasible motion plan we use direct transcription [27, 29, 69] to convert the differential equations for the centroidal dynamics (4.10) to a set of algebraic equations and solve the resulting NLP.

5.4.1 Decision variables and cost

The decision variables of the new problem consist of the values of all time-varying quantities in (4.10) at N knot-points as well as the duration of the intervals between the knot-points, h. Those time-varying quantities are the position vector, $\mathbf{q}$, and velocity vector, $\mathbf{v}$, the COM position, $\mathbf{r}$, velocity, $\dot{\mathbf{r}}$, and acceleration, $\ddot{\mathbf{r}}$, contact positions, $\mathbf{r}_i$, contact forces, $\mathbf{f}_i$, centroidal angular momentum, $\mathbf{k}$, and its rate, $\dot{\mathbf{k}}$. The cost function for all of the results in this chapter is:

$$J(\mathbf{x}) = \sum_{n=1}^{N} h[n]\Big[\|\mathbf{q}[n] - \mathbf{q}_{\text{nom}}[n]\|^2_{Q_\mathbf{q}} + \|\mathbf{v}[n]\|^2_{Q_\mathbf{v}} + \|\ddot{\mathbf{r}}[n]\|^2 + \sum_{i=1}^{n_{\text{ee}}} \Big(c_1 \Big\|\mathbf{f}_j[n]\Big\|^2\Big)\Big]$$

where $\mathbf{x}$ is the full decision variable vector

$$\begin{aligned}\mathbf{x} = (&h[1,\dots,N], \mathbf{q}[1,\dots,N]', \mathbf{v}[1,\dots,N]', \mathbf{k}[1,\dots,N]', \dot{\mathbf{k}}[1,\dots,N]', \\ &\mathbf{r}[1,\dots,N]', \dot{\mathbf{r}}[1,\dots,N]', \ddot{\mathbf{r}}[1,\dots,N]', \\ &\mathbf{r}_{1,\dots,N}[1]', \mathbf{f}_{1,\dots,N}[1]', \dots \mathbf{r}_{n_{\text{ee}}}[1,\dots,N]', \mathbf{f}_{n_{\text{ee}}}[1,\dots,N]',)'\end{aligned} \tag{5.5}$$

where $x[1,\dots,N] = (x[1]', \dots x[N]')'$ and where

$$\|\mathbf{y}\|_Q^2 = \mathbf{y}'Q\mathbf{y}$$

for some positive-semidefinite matrix Q. The first term penalizes deviation from a pre-specified joint trajectory. The second term penalizes non-zero joint velocities. The third term penalizes center of mass acceleration, while the fourth penalizes external forces and torques.

5.4.2 Basic constraints

Prior to the addition of any task specific constraints, the transcribed nonlinear program contains several basic constraints on the dynamics and kinematics of the robot. These ensure dynamic feasibility of the resulting trajectory (ignoring torque limits).

Dynamics

We enforce centroidal dynamics with the following constraints, which correspond to (4.10a) and (4.10b)

$$m\ddot{\mathbf{r}}[n] = \sum_{i=1}^{n_{\text{ee}}} \mathbf{f}[n] + m\mathbf{g}$$

$$\dot{\mathbf{k}}[n] = \sum_{i=1}^{n_{\text{ee}}} (\mathbf{r}_i[n] \times \mathbf{f}_i[n])$$

Time integration

Time integration for the generalized positions, angular momentum, and COM is implemented with a backwards-Euler scheme for simplicity and numerical stability in the following constraints:

$$\begin{aligned}
\mathbf{q}[n] &= \mathbf{q}[n-1] + \mathbf{v}[n]h[n] \\
\mathbf{k}[n] &= \mathbf{k}[n-1] + \dot{\mathbf{k}}[n]h[n]. \\
\mathbf{r}[n] &= \mathbf{r}[n-1] + \dot{\mathbf{r}}[n]h[n] \\
\dot{\mathbf{r}}[n] &= \dot{\mathbf{r}}[n-1] + \ddot{\mathbf{r}}[n]h[n]
\end{aligned}$$

Consistency

To ensure consistency between the various components of the decision variable we introduce the following kinematic constraints

$$\mathbf{k}[n] = \mathbf{A}_{\mathrm{G}}^{\mathbf{k}}(\mathbf{q}[n])\mathbf{v}[n]. \tag{5.6}$$

$$\mathbf{r}[n] = \mathrm{com}(\mathbf{q}[n]) \tag{5.7}$$

$$\mathbf{r}_i[n] = \mathrm{FK}_{\mathrm{ee},i}(\mathbf{q}[n]) \text{ for } i = 1, \ldots, n_{\mathrm{ee}} \tag{5.8}$$

where $\mathbf{A}_{\mathrm{G}}^{\mathbf{k}}$ is the top half of $\mathbf{A}_{\mathrm{G}}$ as defined in (5.1), com$(\cdot)$ is a function that computes the robot's COM at a given configuration, and $\mathrm{FK}_{\mathrm{ee},i}$ is the forward kinematics function that computes the location of the i-th end-effector at a given configuration, as given in (3.1). It is important to note that (5.6) ensures that the joint angle trajectories are consistent with the angular momentum trajectory. The result is that any feasible solution to the trajectory optimization satisfies the dynamics of the robot for some value of the joint torques.

Contact constraints

In the case where the sequence of end-effector contacts is specified a priori, then for each desired contact we have

$$\begin{bmatrix} \mathbf{r}_i \\ \mathbf{f}_i \end{bmatrix} \in \mathcal{S}_{j_i[n]}$$

where $j_i[n]$ is the index of the region to which the i-th end-effector's position and contact force belong at time n.

Joint limits

Finally, we place bounds on $\mathbf{q}$ and $\mathbf{v}$ to reflect the position and velocity of the robot's joints.

$$\mathbf{q}_{\text{lb}} \leq \mathbf{q}[n] \leq \mathbf{q}_{\text{ub}} \tag{5.9}$$

$$\mathbf{v}_{\text{lb}} \leq \mathbf{v}[n] \leq \mathbf{v}_{\text{ub}} \tag{5.10}$$

where (5.9) and (5.10) are enforced element-wise for all $n \in \{1, \ldots, N\}$.

Full problem definition

The full NLP definition is therefore given by

$$
\begin{aligned}
\underset{\mathbf{x}}{\text{minimize}} \quad & J(\mathbf{x}) && && \text{(P2)}\\
\text{subject to} \quad & \dot{\mathbf{r}}[n] = \dot{\mathbf{r}}[n-1] + \ddot{\mathbf{r}}[n]h[n] && \Big\} && \forall\, n \in \{2, \dots, N\}\\
& \mathbf{r}[n] = \mathbf{r}[n-1] + \dot{\mathbf{r}}[n]h[n] && && \\
& \mathbf{k}[n] = \mathbf{k}[n-1] + \dot{\mathbf{k}}[n]h[n] && && \\
& \mathbf{q}[n] = \mathbf{q}[n-1] + \mathbf{v}[n]h[n] && && \\
& \ddot{\mathbf{r}}[n] = \frac{1}{m}\sum_{i=1}^{n_{\text{ee}}} \mathbf{f}[n] + m\mathbf{g} && \Big\} && \forall\, n \in \{1, \dots, N\}\\
& \mathbf{r}[n] = \text{com}(\mathbf{q}[n]) && && \\
& \dot{\mathbf{k}}[n] = \frac{1}{I}\sum_{i=1}^{n_{\text{ee}}} (\mathbf{r}_i[n] \times \mathbf{f}_i[n]) && && \\
& \mathbf{k}[n] = \mathbf{A}^{\mathbf{k}}_{\text{G}}(\mathbf{q}[n])\mathbf{v}[n] && && \\
& (\mathbf{r}_i[n], \mathbf{f}_i[n])' \in \mathcal{S}_{j_i[n]} && && \\
& \mathbf{r}_i[n] = \text{FK}_{\text{ee},i}(\mathbf{q}[n]) \text{ for } i = 1, \dots, n_{\text{ee}} && && \\
& \text{task-specific constraints} && && .
\end{aligned}
$$

where the task-specific constraints may include, for example, initial/final state constraints, total duration constraints, or any of the constraints described in Section 5.3

5.5 Tools

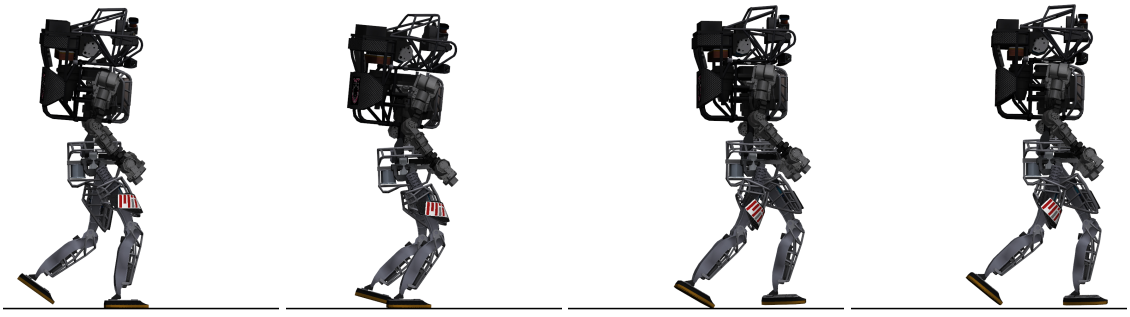
We solve (P2) using the Drake toolbox [70] and the sparse NLP solver SNOPT [71]. Drake is an open-source planning, control, and analysis toolbox for nonlinear dynamical system written in MATLAB [72] and C++, and maintained by Tedrake et al. It provides utilities for kinematics and dynamics of rigid-body manipulators, as well as classes for nonlinear programming and trajectory optimization. We implemented the approach presented here as part of Drake's trajectory optimization class hierarchy and used SNOPT as the solver for the underlying NLP.

5.6 Results

We first reported these results in Kuindersma et al. [3] and Dai, Valenzuela, and Tedrake [64] in the context of our planning and control architecture for Boston Dynamics' humanoid robot Atlas.

5.6.1 Humanoid Running

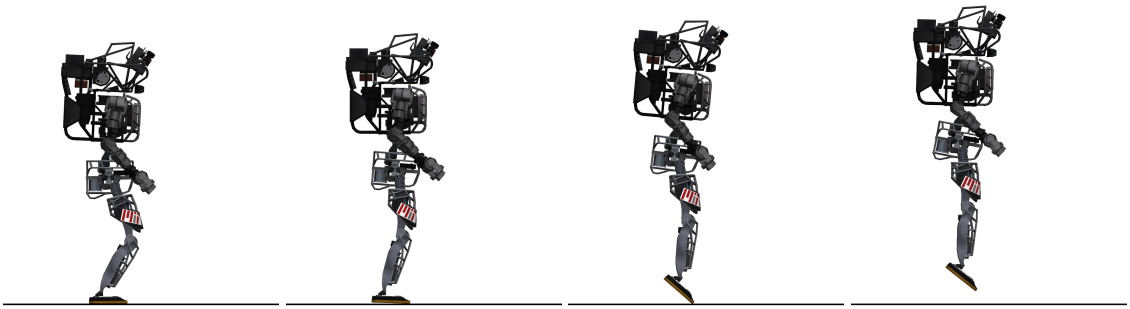
To plan a running motion for Atlas, we consider a single half stride starting at the apex of a flight phase. We fix the contact sequence to be flight, left-stance, left-toe-stance, flight. By constraining the initial and final conditions of the trajectory such that all quantities are mirrored about the robot's sagittal plane, we obtain a trajectory that can be mirrored to yield a full stride. In addition to enforcing the contact sequence and the periodicity of the trajectory, we also specify a stride-length and average speed (1.5 m and 2 m/s respectively for the gait shown in Figure 5-1), require at least 3 cm of clearance between links to avoid self-collisions, and constrain the gaze of the robot's head cameras to be no more than 15° from the direction of travel. Solving for the half-stride motion takes approximately 2.5 min on a computer with a 3.3 GHz Intel i7 CPU. The controller described by Kuindersma et al. [3, §4] is used to stabilize the resulting trajectory (mirrored and looped to produce a ten-stride sequence) on a simulated model of Atlas with accurate torque limits. One half-stride of the simulated motion is depicted in Figure 5-1. Figure 5-2 shows the first half



(a) (b) (c) (d)

Figure 5-1: Snapshots from a half-stride of simulated running at 2 m/s. (a) Touch-down, (b) mid-stance, (c), toe-off, (d) mid-flight

of a jumping motion generated by the whole-body planner and stabilized by the controller described by Kuindersma et al. [3, §4]. This motion was generated by constraining the corners of the feet to be at least 5 cm above the ground at the mid-point of the trajectory, and requiring that the robot's motion be symmetric about its sagittal plane. Solving for the jumping motion takes approximately 1.5 min on a computer with a 3.1 GHz Intel i7 CPU. As in the running case, this entire motion was stabilized in a simulation with torque limits.



(a) (b) (c) (d)

Figure 5-2: Snapshots from a simulated jump. (a) Starting posture, (b) heel-off, (c) toe-off, (d) mid-flight

Our final example is another jumping motion, this time from a cinder block to the ground. The robot's starting posture on top of the cinder block is specified, but its final posture is not. We require that the feet maintain at least 3 cm of clearance from the cinder block at all times after take-off and that they be on the ground at the end of the trajectory. Given these constraints, and the requirement that the robot's motion be symmetric about its sagittal plane, the planner finds the motion depicted in Figure 5-3. Solving this problem takes approximately 10 min on a computer with a 3.1 GHz Intel i7 CPU. The jump down motion was stabilized by our controller in simulation with torque limits.

5.6.2 Quadrupedal Gaits

With minimal alteration, the program that generated the results in Section 5.6.1 can be converted to synthesize walking and running gaits for a quadruped. We present here trotting and galloping gaits for LittleDog, a quadrupedal robot built by Boston Dynamics.

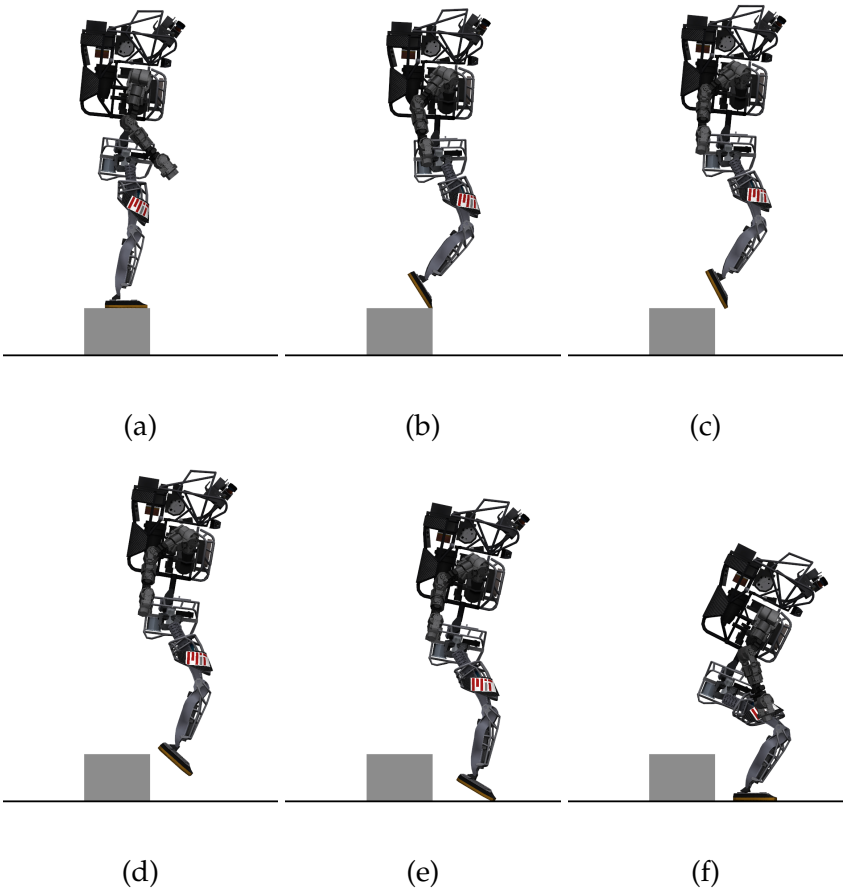
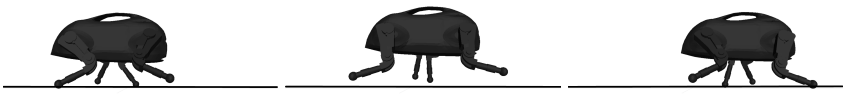


Figure 5-3: Snapshots from a motion plan for jumping off a cinder block. (a) Starting posture, (b) toe-off, (c) Apex, (d) avoiding collision, (e) touch-down, (f) final posture

To do this, we specify the contact assignments of the feet a priori and allow the spacing between knot-points to vary.

Trot

To generate a trotting gait, we formulate an optimization program over a single half-stride. Figure 5-5 shows the gait diagram for a full stride. Unlike a traditional gait diagram [73], which plots foot stance against percentage of the stride period, Fig. 5-5 plots foot stance against knot-point number. This is because we do not specify the timing of the knot-points a priori. As was the case in Section 5.6.1, the trajectory for a full stride can be recovered by concatenating a mirrored trajectory to the original half stride.



(a) Take-off (b) Mid-flight (c) Touch-down

Figure 5-4: Snapshots from a motion plan for a running trot.

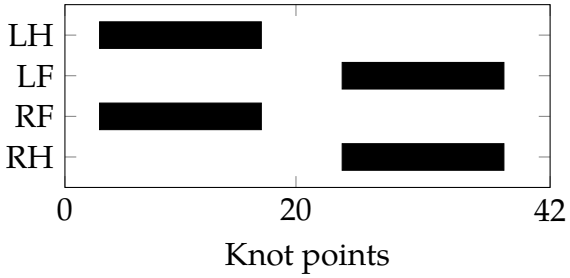


Figure 5-5: Gait graph for a running trot

Gallop

To generate a motion plan for galloping, we enforce the gait shown in Fig. 5-6 over a 41 knot trajectory. Note that, as before, the graph is in terms of the knot-points, not percentage of stride duration.

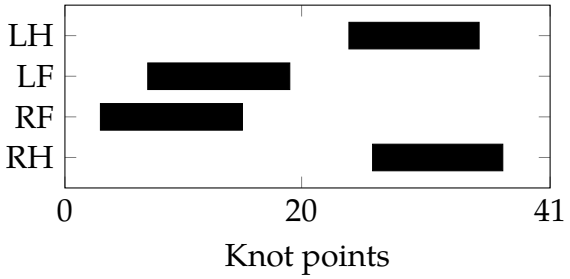


Figure 5-6: Gait graph for a rotary gallop

Chapter 6

Combining MI-convex Planning and Centroidal-Dynamics Full-Kinematics Planning

Once we have an approximate plan for the centroidal dynamics of the robot as obtained through the techniques described in Chapter 4, we are ready to convert it into a whole-body plan. The purpose of this conversion is two-fold.

Firstly, it ensures that our final plan respects the kinematics of the robot. Recall that the planning problem in Chapter 4 includes only approximate constraints on the end-effector positions. While ideally those constraints would be conservative, it may be that adjustments to the end-effector locations are required in order to make them kinematically feasible. This also gives us the opportunity to impose other kinematic constraints that don't fit within the formulation of Chapter 4, such as collision avoidance.

Secondly, it provides a full set of joint-angle reference trajectories. The controller used to implement the plan may require these.

6.1 Translating Footstep-Planning Results into Kinematic Constraints

Now that assignment of end-effector regions has been handled in the mixed-integer step, the constraints on end-effector position are much more straightforward. In the terminology introduced in Section 3.4, for the i-th end-effector there is a set of constants, $j_i[n]$, such that the desired constraints on end-effector positions and contact forces are given by

$$\mathbf{r}_i[n] \in \mathcal{R}_{j_i[n]} \tag{6.1}$$

$$\mathbf{f}_i[n] \in \mathcal{K}_{j_i[n]} \tag{6.2}$$

Equation (6.1) must then be composed with the forward kinematics of the end-effectors to yield

$$A_{j_i[n]}\mathbf{r}_i(\mathbf{q}[n]) \leq b_{j_i[n]} \tag{6.3}$$

6.2 Seeding Trajectory Optimization with Centroidal Quantities from MIQP

In addition to providing end-effector assignments, the results from Chapter 4 give us initial guesses for many of the elements of (5.5).

Centroidal dynamics The position of the robot's center of mass is a decision variable in both (P1) and (P2). Thus, we have

$$\mathbf{r}[n]^0_{\text{P2}} = \mathbf{r}[n]^*_{\text{P1}}.$$

We can seed the angular momentum terms of (P2) based on the solution to (P1) as

$$\mathbf{k}[n]^0_{\text{P2}} = \mathbf{I}\boldsymbol{\omega}[n]^*_{\text{P1}}.$$

Contact forces The end-effector forces are also shared between (P1) and (P2). Therefore,

$$\mathbf{f}_i[n]^0_{\text{P2}} = \mathbf{f}_i[n]^*_{\text{P1}} \text{ for } i = 1, \ldots, n_{\text{ee}}$$

Joint angles To obtain a seed value for the joint angles we solve an inverse kinematics problem at each knot point. We pose the inverse kinematics problem as an NLP with the robot's configuration as the decision variables.

$$\begin{aligned} \mathbf{q}[n]^0_{\text{P2}} = \arg\min_{\mathbf{q}} \quad & \|\mathbf{q} - \mathbf{q}_{\text{nom}}\| \\ \text{subject to} \quad & \text{com}(\mathbf{q}) = \mathbf{r}[n]^*_{\text{P1}} \\ & \text{FK}_i(\mathbf{q}) = \mathbf{r}_i[n]^*_{\text{P1}} \text{ for } i = 1, \ldots, n_{\text{ee}} \\ & \Gamma(\mathbf{q}) = 0 \end{aligned} \tag{6.4}$$

where Γ is the collion avoidance function defined by (5.4).

6.3 Constraints based on MIQP solution

In addition to seeding the optimization as described in Section 6.2, we can restrict the search space of (P2) to a region near the solution of (P1). This yields a new problem, (P2′), which includes the objective and constraints of (P2) as well as the additional constraints:

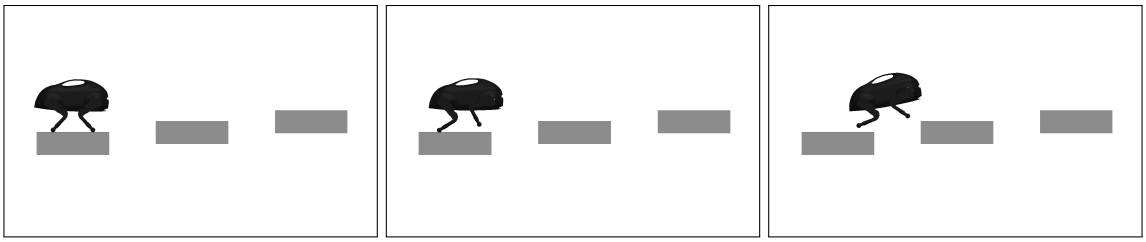
$$\text{FK}_i(\mathbf{q}) = \mathbf{r}_i[n]^*_{\text{P1}} \text{ for } i = 1, \ldots, n_{\text{ee}}$$

6.4 Results

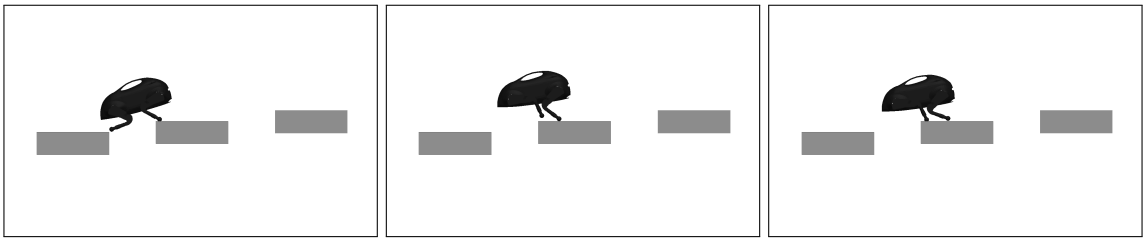
6.4.1 Negotiating irregular terrain

Figures 6-1 and 6-2 show frames from whole-body motion plans we generated by solving (P2′). Figures 6-3 and 6-4 show the center of mass trajectories for motion plans generated by (P1) and (P2).

Figures 6-1 and 6-2 show cases in which SNOPT succeeds in solving (P2′). As (P2′) is



(a) Starting posture (b) Take-off (c) Flight (extended)

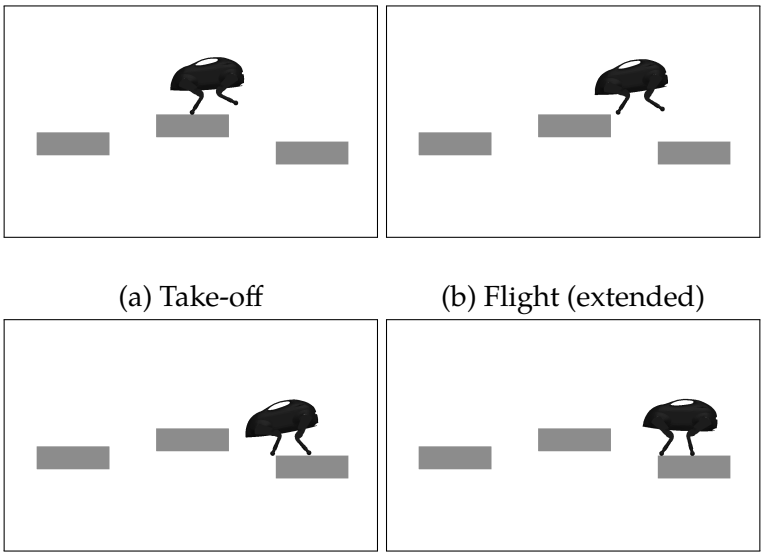


(d) Touch-down (e) Flight (gathered) (f) Touch-down

Figure 6-1: Snapshots from a motion plan for ascending platforms with gaps

a non-convex program, there are scenarios in which it fails. These generally fall into two categories: failure due to constraints on (P2′) that were not present in (P1) and failures due to the relaxation gap in (P1). The most common cause of the failures in the former category is the collision-avoidace constraint included in (P2′). In (P1), collision avoidance is enforced by requiring that if $\mathbf{r}_i[n] \in \mathcal{R}_j$, where $\mathcal{R}_j$ is a free-space region, then $\mathbf{r}_i[n \pm 1] \in \mathcal{R}_j$. This ensures that each segment of the end-effector trajectories is contained within a convex region of free-space. in (P2′), however, this constraint is insufficient, as other parts of the robot might be in collision. Instead, (P2′) incorporates the collision-avoidance constraint described in Section 5.3.4. In situations with complicated environments, this additional non-convex constraint can push the solver into regions of the search space in which no feasible solution can be found.

Failures in the second category are generally due to the solution to (P1) specifying moments about COM, contact forces, and foot positions that violate the bilinear constraints discussed in Section 4.3. This can be addressed by using a tighter relaxation, but at the cost of more expensive footstep planning problems.



(a) Take-off (b) Flight (extended)

(c) Touch-down (d) Final posture

Figure 6-2: Snapshots from a motion plan for staggered platforms with gaps

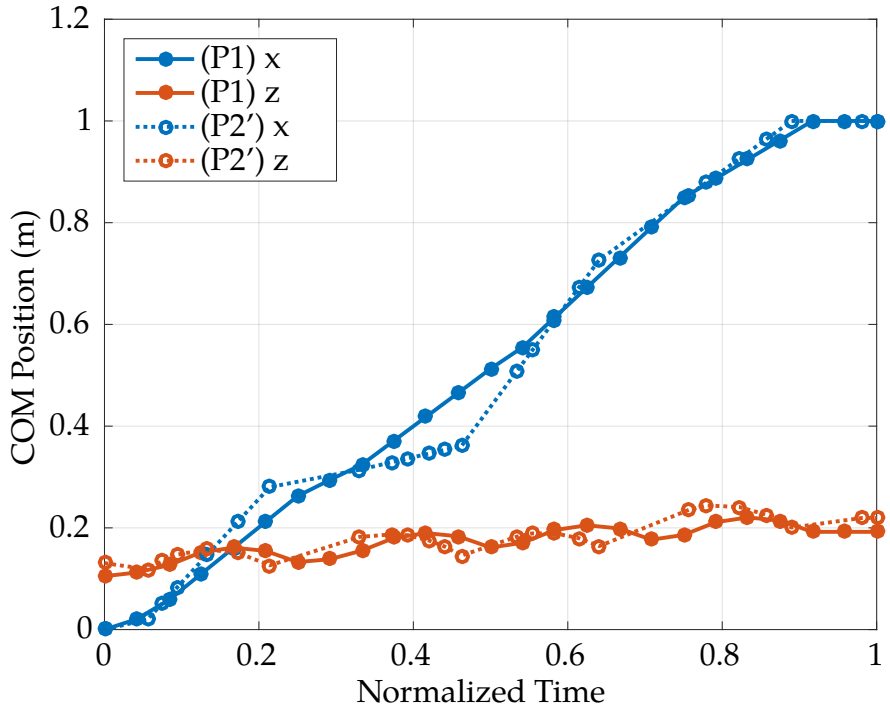


Figure 6-3: COM trajectories for traversing stairsteps with gaps

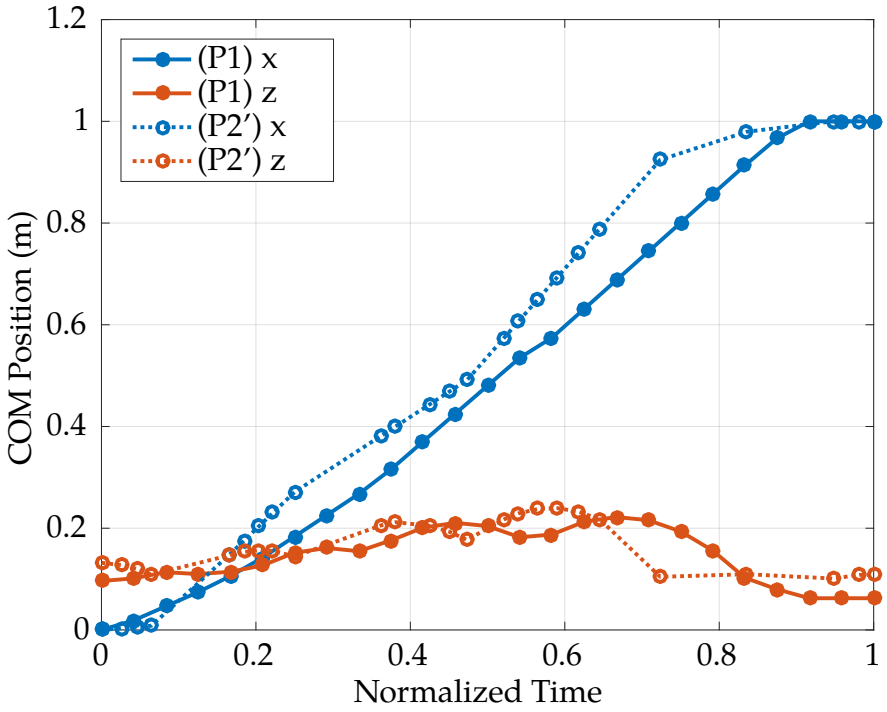


Figure 6-4: COM trajectories for traversing staggered platforms

Chapter 7

Final Discussion and Future Work

In this thesis I have demonstrated the need for planning approaches that can determine end-effector placement without prior specification and have developed a framework for planning aggressive motions of legged robots over irregular terrain. The two main components of that framework are: a mixed-integer convex program that selects end-effector placements as well as the centroidal motion of the robot, and a continuous nonlinear program that computes the centroidal motion of the robot as well as full joint-angle trajectories. Since both of these work with the centroidal dynamics of the robot, we are able to connect them by initializing the second optimization with the results of the first. Using this technique, I have generated plans for a running planar quadruped on stair step terrain with large gaps, a task which prior to this innovation would have been intractable. Additionally, I have applied the continuous portion of this framework to planning problems with pre-specified end-effector placement on quadrupedal and humanoid robots. I now present directions for further work that can expand the effectiveness of these techniques.

7.1 Trajectory stabilization

Ultimately, these planning techniques are only useful in so far as their results can be executed on robotic hardware. Executing the plans developed by the methods in Chapters 4 and 6 will require a stabilizing controller. Our simulated running and jumping results on Atlas (Section 5.6) demonstrate that the controller described in Kuindersma et al. [3,

§4] can be used for this purpose. One interesting aspect of that controller, as well as other similar QP-based controllers [2, 74, 75] is that it can work either with the full joint trajectory or with goals for specific links on the robot. This means that we can conceivably feed the COM and end-effector trajectories produced by our MIQP-based planner, (P1), directly into the controller, without needing to solve the NLP for joint angles. This will save time, making it more feasible to use on hardware.

7.2 Three-dimensional MIQP-based planning

Now that we have shown the planar model described in Section 4.2.2 to be effective in selecting end-effector placements, the logical next step is to use a three-dimensional version. As noted in Section 4.2.2, this approach will contain many more bilinear terms than the planar one. This will have a strong impact on the choice of mixed-integer convex approximation for the bilinear terms, as approximations that introduce more binary variables will drastically increase the amount of computation required by the method. While it yields the loosest relaxation of the strategies assessed in Section 4.3, a McCormick envelope approximation may be the best here, as it does not require the addition of any bilinear auxiliary variables.

The first consideration in the 3D model will be the representation of orientation. In Section 4.2.2 we show the beginning of an approach using rotation matrices to encode orientation, which would introduce nine bilinear terms for each rotated vector. The other option would be to use quaternions, which would introduce twenty-four third order terms. These can be converted to bilinear terms recursively [76], but will still result in a higher number of bilinear terms than a rotation matrix. Thus from this standpoint, rotation matrices are the superior choice.

The next challenge in a three-dimensional implementation is encoding the rotational dynamics of the system. Whereas the previous paragraph dealt with the tradeoffs between different representations of rotation at a single point in time, here we are concerned with the integration constraints applied to the rotations at adjacent knot points. Both rotation matrices and quaternions live in a space whose dimension is lower than their

number of parameters, which means that there are additional constraints that must be enforced during integration to ensure that they remain in the appropriate space: namely, orthonormality for rotation matrices and unit norm for quaternions. Lie-group-based integration methods provide a way to avoid these additional constraints but involve the exponential map, which is non-convex. However, the first course of action should be to use a second-order approximation of the exponential map in order to return to a bilinear model. If that is not successful, we will want to reevaluate the use of quaternions since they require fewer constraints in order to ensure that they remain in SO(3). The treatment of the translational dynamics and of the end-effector placement will remain unchanged.

7.3 Seeding trajectory optimization with full dynamics

Another variation of the method presented here would be to pass from the MIQP problem to a trajectory optimization that incorporates the full dynamics of the robot. This would allow us to put constraints on the joint torques exerted by the robot. Recall that by optimizing over the dynamics of the unactuated degrees of freedom in Chapter 5, we traded complexity of the dynamics for the ability to optimize over joint torques. While a trajectory optimization with torque constraints may not be necessary for high-powered robots with full actuation on their joints like Atlas, it would be very useful for less powerful robots where torque constraints are the limiting factor on what motions they can perform. Seeding this optimization would be more challenging than seeding the centroidal dynamics, full kinematics method because we would need to provide an initial guess for the joint torques. This could be accomplished by first computing a nominal joint angle trajectory as in Section 6.2 and then solving the inverse dynamics problem at each time step to generate a guess for the torques.

7.4 Comparisons to other approaches

It is possible that similar problems could be solved with an expanded version of the NLP described by Posa, Cantu, and Tedrake [44]. As it is now, this method only allows

each end-effector to have one potential support surface. An expansion would be needed to allow for the possibility of end-effectors contacting more than one surface over the course of a trajectory, for example: ground, cinder block and a stair. With such a change, a comparison between these two methods would be useful to determine their relative strengths and weaknesses. This comparison should include the success rates of the two methods over a common set of planning problems as well as the running times for each method. The strength of the method proposed by Posa is that it treats the contact selection and full multi-body dynamics simultaneously, which means that it may be able to find solutions in cases where the sequential approach presented here fails. That strength is also its weakness in that the very complex nonlinear program required may prove to be much more computationally expensive.

7.5 Conclusion

Prior to this thesis, the state of the art in motion planning for legged robots either selected footsteps first and dynamically-balanced center of mass trajectory second or solved for joint angles and contact placements as a single NLP. By way of simplification, the first approach disregards the fact that COM motion and footstep placement are inextricably linked; in so doing they restrict themselves to solving a smaller subset of motion planning problems. The second approach yields very large NLPs that are computationally expensive and sensitive to the initial guess provided. My primary contribution in this work is a formulation that simultaneously treats both the placement of end-effectors and the centroidal dynamics of the robot via MIQP. This novel approach to motion planning for legged robots allows us to solve, in a general way, challenging problems such as running and leaping over gaps as I have demonstrated on a model of the quadrupedal robot, LittleDog. Furthermore, I have presented a unique method developed in collaboration with H. Dai that generates full body trajectories that respect the centroidal dynamics of the robot as well as end-effector placements chosen either by the user directly or through the aforementioned MIQP-based planner. We have generated running, jumping and other aggressive maneuvers by applying this method to the humanoid robot Atlas. These con-

tributions put us one step closer to a world in which quadrupedal robots come to the aid of embattled soldiers and humanoids work in dangerous new places so that humans can stay out of harm's way.

Bibliography

[1] Hae-Won Park, Patrick Wensing, and Sangbae Kim. "Online Planning for Autonomous Running Jumps Over Obstacles in High-Speed Quadrupeds". In: *Proceedings of Robotics: Science and Systems*. Rome, Italy, July 2015.

[2] Siyuan Feng et al. "Optimization Based Controller Design and Implementation for the Atlas Robot in the DARPA Robotics Challenge Finals". In: *Humanoid Robots, 15th IEEE-RAS International Conference on*. 2015.

[3] Scott Kuindersma et al. "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot". In: *Autonomous Robots* (July 2015), pp. 1–27.

[4] J.R. Rebula et al. "A Controller for the LittleDog Quadruped Walking on Rough Terrain". In: *2007 IEEE International Conference on Robotics and Automation*. Apr. 2007, pp. 1467–1473.

[5] J.Z. Kolter, M.P. Rodgers, and A.Y. Ng. "A control architecture for quadruped locomotion over rough terrain". In: *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008*. May 2008, pp. 811–818.

[6] Katie Byl et al. "Reliable Dynamic Motions for a Stiff Quadruped". In: *Experimental Robotics*. Ed. by Oussama Khatib, Vijay Kumar, and George J. Pappas. Springer Tracts in Advanced Robotics 54. Springer Berlin Heidelberg, 2009, pp. 319–328.

[7] M. Kalakrishnan et al. "Learning, planning, and control for quadruped locomotion over challenging terrain". In: *The International Journal of Robotics Research* 30.2 (2011), pp. 236–258.

[8] Alexander Shkolnik and Russ Tedrake. "Sample-Based Planning with Volumes in Configuration Space". In: *arXiv:1109.3145 [cs]* (Sept. 2011). arXiv: 1109.3145.

[9] Matt Zucker et al. "Optimization and learning for rough terrain legged locomotion". In: *The International Journal of Robotics Research* 30.2 (Feb. 2011), pp. 175–191.

[10] Alexander W. Winkler et al. "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.

[11] A. Winkler et al. "Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. May 2014, pp. 6476–6482.

[12] R. Deits and R. Tedrake. “Footstep planning on uneven terrain with mixed-integer convex optimization”. In: *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Nov. 2014, pp. 279–286.

[13] J. Kuffner et al. “Online footstep planning for humanoid robots”. In: *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*. Vol. 1. Sept. 2003, 932–937 vol.1.

[14] J. Chestnutt et al. “An adaptive action model for legged navigation planning”. In: *2007 7th IEEE-RAS International Conference on Humanoid Robots*. Nov. 2007, pp. 196–202.

[15] Paul Vernaza et al. “Search-based planning for a legged robot over rough terrain”. In: *IEEE International Conference on Robotics and Automation, 2009. ICRA '09*. May 2009, pp. 2380–2387.

[16] L. Baudouin et al. “Real-time replanning using 3D environment for humanoid robot”. In: *2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Oct. 2011, pp. 584–589.

[17] A Herdt et al. “Online Walking Motion Generation with Automatic Footstep Placement”. In: *ADVANCED ROBOTICS* 24.5-6 (2010), pp. 719–737.

[18] M. Fallon et al. “An Architecture for Online Affordance-based Perception and Whole-body Planning”. In: *CSAIL Technical Reports* (Mar. 2014).

[19] A. Herdt, N. Perrin, and P.-B. Wieber. “Walking without thinking about it”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2010, pp. 190–195.

[20] S.M. LaValle and J.J. Kuffner Jr. “Randomized kinodynamic planning”. In: *1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings*. Vol. 1. 1999, 473–479 vol.1.

[21] J.J. Kuffner and S.M. LaValle. “RRT-connect: An efficient approach to single-query path planning”. In: *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00*. Vol. 2. 2000, 995–1001 vol.2.

[22] A Yershova et al. “Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005.* Apr. 2005, pp. 3856–3861.

[23] Alexander Shkolnik, M. Walter, and R. Tedrake. “Reachability-guided sampling for planning under differential constraints”. In: *IEEE International Conference on Robotics and Automation, 2009. ICRA '09*. May 2009, pp. 2859–2865.

[24] Anna Yershova and Steven M. LaValle. “Motion Planning for Highly Constrained Spaces”. In: *Robot Motion and Control 2009*. Ed. by Krzysztof R. Kozłowski. Lecture Notes in Control and Information Sciences 396. Springer London, Jan. 2009, pp. 297–306.

[25] S. Karaman et al. “Anytime Motion Planning using the RRT*”. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. May 2011, pp. 1478–1483.

[26] L. Jaillet and J.M. Porta. "Path Planning Under Kinematic Constraints by Rapidly Exploring Manifolds". In: *IEEE Transactions on Robotics* 29.1 (Feb. 2013), pp. 105–117.

[27] O. von Stryk and R. Bulirsch. "Direct and indirect methods for trajectory optimization". In: *Annals of Operations Research* 37.1 (Dec. 1992), pp. 357–373.

[28] Katja Mombaur. "Using optimization to create self-stable human-like running". In: *Robotica* 27.03 (May 2009), 321–330.

[29] J. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Advances in Design and Control. Society for Industrial and Applied Mathematics, Jan. 2010.

[30] A. El Khoury, F. Lamiraux, and M. Taïx. "Optimal motion planning for humanoid robots". In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. May 2013, pp. 3136–3141.

[31] Robert G. Gottlieb. "Rapid Convergence to Optimum Solutions Using a Min-H Strategy". In: *AIAA Journal* 5.2 (1966), pp. 322–329.

[32] Ernst D. Dickmanns and Klaus H. Well. "Approximate Solution of Optimal Control Problems Using Third Order Hermite Polynomial Functions". In: *Optimization Techniques IFIP Technical Conference*. Ed. by Prof Dr G. I. Marchuk. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 1975, pp. 158–166.

[33] Dr John T. Betts. "Trajectory Optimization Using Sparse Sequential Quadratic Programming". In: *Optimal Control*. Ed. by Prof Dr R. Bulirsch et al. ISNM International Series of Numerical Mathematics 111. Birkhäuser Basel, Jan. 1993, pp. 115–128.

[34] Daniel B. Leineweber et al. "An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: theoretical aspects". In: *Computers & Chemical Engineering* 27.2 (Feb. 2003), pp. 157–166.

[35] G. Schultz and K. Mombaur. "Modeling and Optimal Control of Human-Like Running". In: *IEEE/ASME Transactions on Mechatronics* 15.5 (Oct. 2010), pp. 783–792.

[36] C.D. Remy, K. Buffinton, and R. Siegwart. "A MATLAB framework for efficient gait creation". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2011, pp. 190–196.

[37] S. Kajita et al. "Biped walking pattern generation by using preview control of zero-moment point". In: *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*. Vol. 2. Sept. 2003, 1620–1626 vol.2.

[38] S. Kajita et al. "Resolved momentum control: humanoid motion planning based on the linear and angular momentum". In: *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings*. Vol. 2. Oct. 2003, 1644–1650 vol.2.

[39] H. Hirukawa et al. "A universal stability criterion of the foot contact of legged robots - adios ZMP". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. May 2006, pp. 1976–1983.

[40] K. Koyanagi et al. “A pattern generator of humanoid robots walking on a rough terrain using a handrail”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*. Sept. 2008, pp. 2617–2622.

[41] M. Vukobratović and J. Stepanenko. “On the stability of anthropomorphic systems”. In: *Mathematical Biosciences* 15.1–2 (Oct. 1972), pp. 1–37.

[42] Miomir Vukobratović, Branislav Borovac, and Veljko Potkonjak. “Zmp: a review of some basic misunderstandings”. In: *International Journal of Humanoid Robotics* 03.02 (June 2006), pp. 153–175.

[43] Igor Mordatch, Emanuel Todorov, and Zoran Popović. “Discovery of Complex Behaviors Through Contact-invariant Optimization”. In: *ACM Trans. Graph.* 31.4 (July 2012), 43:1–43:8.

[44] Michael Posa, Cecilia Cantu, and Russ Tedrake. “A direct method for trajectory optimization of rigid bodies through contact”. In: *The International Journal of Robotics Research* 33.1 (Jan. 2014), pp. 69–81.

[45] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson/Prentice Hall, 2005.

[46] Roy Featherstone. “Inverse Dynamics”. In: *Rigid Body Dynamics Algorithms*. Springer US, 2008, pp. 89–100.

[47] E. Balas. “Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems”. In: *SIAM Journal on Algebraic Discrete Methods* 6.3 (July 1985), pp. 466–486.

[48] Francisco Trespalacios and Ignacio E. Grossmann. “Review of Mixed-Integer Nonlinear and Generalized Disjunctive Programming Methods”. In: *Chemie Ingenieur Technik* 86.7 (July 2014), pp. 991–1012.

[49] Samuel Burer and Adam N. Letchford. “Non-convex mixed-integer nonlinear programming: A survey”. In: *Surveys in Operations Research and Management Science* 17.2 (July 2012), pp. 97–106.

[50] R. Deits and R. Tedrake. “Efficient mixed-integer planning for UAVs in cluttered environments”. In: IEEE, Jan. 2015.

[51] Garth P. McCormick. “Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems”. In: *Mathematical Programming* 10.1 (Dec. 1976), pp. 147–175.

[52] Scott Kolodziej, Pedro M. Castro, and Ignacio E. Grossmann. “Global optimization of bilinear programs with a multiparametric disaggregation technique”. In: *Journal of Global Optimization* 57.4 (Jan. 2013), pp. 1039–1063.

[53] Pedro M. Castro. “Normalized multiparametric disaggregation: an efficient relaxation for mixed-integer bilinear problems”. In: *Journal of Global Optimization* (July 2015), pp. 1–20.

[54] L. R. Foulds, D. Haugland, and K. JÖrnsten. “A bilinear approach to the pooling problem”. In: *Optimization* 24.1-2 (Jan. 1992), pp. 165–180.

[55] I. Quesada and I. E. Grossmann. "Global optimization of bilinear process networks with multicomponent flows". In: *Computers & Chemical Engineering*. An International Journal of Computer Application in Chemical Engineering 19.12 (Dec. 1995), pp. 1219–1242.

[56] Alberto Caprara and Michele Monaci. "Bidimensional packing by bilinear programming". In: *Mathematical Programming* 118.1 (Apr. 2009), pp. 75–108.

[57] M. Chandraker and D. Kriegman. "Globally optimal bilinear programming for computer vision applications". In: *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*. June 2008, pp. 1–8.

[58] João P. Teles, Pedro M. Castro, and Henrique A. Matos. "Multi-parametric disaggregation technique for global optimization of polynomial programming problems". In: *Journal of Global Optimization* 55.2 (Nov. 2011), pp. 227–251.

[59] William E Hart, Jean-Paul Watson, and David L Woodruff. "Pyomo: modeling and solving mathematical programs in Python". In: *Mathematical Programming Computation* 3.3 (2011), pp. 219–260.

[60] William E Hart et al. *Pyomo–optimization modeling in python*. Vol. 67. Springer Science & Business Media, 2012.

[61] Gurobi Optimization Inc. *Gurobi Optimizer Reference Manual*. 2015. URL: `http://www.gurobi.com`.

[62] Pietro Belotti et al. "Branching and bounds tighteningtechniques for non-convex MINLP". In: *Optimization Methods and Software* 24.4-5 (Oct. 2009), pp. 597–634.

[63] Robin L. H. Deits and Russ Tedrake. "Computing large convex regions of obstacle-free space through semidefinite programming". In: *Submitted to: Workshop on the Algorithmic Fundamentals of Robotics* (Aug. 2014).

[64] Hongkai Dai, A. Valenzuela, and R. Tedrake. "Whole-body motion planning with centroidal dynamics and full kinematics". In: *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Nov. 2014, pp. 295–302.

[65] David E. Orin, Ambarish Goswami, and Sung-Hee Lee. "Centroidal dynamics of a humanoid robot". In: *Autonomous Robots* 35.2-3 (Oct. 2013), pp. 161–176.

[66] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. "A fast procedure for computing the distance between complex objects in three-dimensional space". In: *IEEE Journal of Robotics and Automation* 4.2 (1988), pp. 193–203.

[67] Erwin Coumans. *Bullet physics library*. 2015. URL: `www.bulletphysics.org`.

[68] John Schulman et al. "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization". In: *Robotics: Science and Systems*. 2013.

[69] C. R. Hargraves and S. W. Paris. "Direct trajectory optimization using nonlinear programming and collocation". In: *Journal of Guidance, Control, and Dynamics* 10.4 (1987), pp. 338–342.

[70] Russ Tedrake. *Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems*. 2014. URL: `http://drake.mit.edu`.

[71] P. Gill, W. Murray, and M. Saunders. "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization". In: *SIAM Journal on Optimization* 12.4 (Jan. 2002), pp. 979–1006.

[72] The MathWorks Inc. *MATLAB*. Version R2015b. Natick, Massachusetts, United States of America.

[73] Milton Hildebrand. "The Quadrupedal Gaits of Vertebrates". In: *BioScience* 39.11 (1989).

[74] L. Righetti and S. Schaal. "Quadratic programming for inverse dynamics with optimal distribution of contact forces". In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Nov. 2012, pp. 538–543.

[75] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. "Hierarchical quadratic programming: Fast online humanoid-robot motion generation". In: *The International Journal of Robotics Research* 33.7 (June 2014), pp. 1006–1028.

[76] João P. Teles, Pedro M. Castro, and Henrique A. Matos. "Univariate parameterization for global optimization of mixed-integer polynomial problems". In: *European Journal of Operational Research* 229.3 (Sept. 2013), pp. 613–625.