Trajectory Optimization for Dynamic Aerial Motions of Legged Robots

by

Matthew Chignoli

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

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2021

Author .....
Department of Mechanical Engineering
January 15, 2021

Certified by .....
Sangbae Kim
Professor
Thesis Supervisor

Accepted by .....
Nicolas Hadjiconstantinou
Chairman, Committee on Graduate Students

Trajectory Optimization for Dynamic Aerial Motions of Legged Robots

by

Matthew Chignoli

Submitted to the Department of Mechanical Engineering
on January 15, 2021, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

A novel framework for planning and executing dynamic aerial motions for legged robots is developed. These dynamic capabilities allow legged robots to conquer challenging obstacles like gaps and hurdles that cannot be traversed via standard walking and running gaits. The framework consists of two main steps. First, a motion planning step uses trajectory optimization to generate a dynamically feasible motion of the robot that achieves a desired behavior. The desired behavior, which comes from a higher-level planner or a human operator, can specify an arbitrary 3D motion task such as jumping onto a platform or performing a front flip. The trajectory optimization simultaneously optimizes the centroidal dynamics and joint-level kinematics of the robot to plan general 3D motions. Novel actuator constraints are imposed on the optimization that ensure all planned motions are feasible for implementation on hardware, and a two-stage formulation of the optimization automatically generates dynamically-informed warm starts to the optimization that dramatically reduce solve times. The second step of the framework is a unified whole-body controller that tracks these planned motions. The whole-body controller uses a prioritized task hierarchy that is optimized for robust tracking and safe landing of dynamic aerial motions. The ability of the proposed framework to reliably produce 3D aerial motions such as running jumps, barrel rolls, and flips is demonstrated on the MIT Humanoid robot in simulation and on the MIT Mini Cheetah robot both in simulation as well on hardware.

Thesis Supervisor: Sangbae Kim
Title: Professor

Acknowledgments

I would first like to express my immense gratitude for my family and the support they have provided me throughout my life. None of my accomplishments would have been possible without their love and encouragement. To my parents especially, I want to thank you for all you have done to allow me to pursue my education and for being role models who inspire me to work tirelessly in pursuit of my goals.

I also want to thank my advisor Sangbae Kim for providing me the opportunity to pursue my research interests and for the constant guidance he has given me throughout his mentorship. Professor Kim's wealth of robotics knowledge, his ability to inspire and educate his students, and the culture of academic curiosity he has fostered in the MIT Biomimetic Robotics Lab has offered me an opportunity for personal and professional growth that I will always be thankful for.

To all members of the Biomimetic Robotics Lab, I thank you for not only your generous practical assistance (developing ideas, debugging code, maintaining the Cheetahs) but also for providing me a constant source of motivation and a reminder of why I am so passionate about robotics. I would especially like to thank Donghyun Kim for his exceptional generosity, expertise, and dedication to the lab and all of its members. I simply cannot imagine the lab running without him.

I would also like to thank Patrick Wensing. Professor Wensing introduced me to robotics when I was an undergraduate, and I have been thankful for that ever since. Pat, thank you for helping me find my passion and for your continued guidance.

There are simply too many people who have helped make this thesis a success to list them all. To all those people I have not listed, please know that your generosity and helpfulness are not lost on me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Considerable research efforts over the past two decades aimed at improving the mobility of robotic systems have lead to the development of impressive designs for legged robots as well as extraordinarily capable locomotion controllers and high-level navigation planners. The overwhelming majority of this research into legged robots has focused on walking and running as the means to achieve high speeds and conquer challenging terrain like stairs or unstructured outdoor environments. While walking and running are essential abilities for legged systems, these capabilities alone fail to unlock the full potential of legs as mechanisms for achieving mobility. Consider the dynamic feats displayed by humans when they play sports or the graceful precision with which a house cat jumps onto and off of surfaces many times taller than its own height. Planning and executing these motions are independently difficult tasks, and coordinating planning with execution is more difficult still. Endowing robots with these skills requires a systemic approach, or framework, that can reliably produce a diverse set of such aerial motions that unlock the full dynamic capabilities of legged systems.

1.1 Motivation

Mobile robots present a unique opportunity to improve the health and safety of human workers across a variety of industries. Achieving anywhere close to human level

perception, locomotion, and manipulation capabilities with robots would mean that humans working hazardous or physically strenuous jobs could have those burdens significantly alleviated or even altogether eliminated. Search and rescue is the most commonly offered application for mobile robots, but similarly, any job that takes place in a potentially hazardous environment such as construction, cleaning, inspection, or maintenance could also be accomplished by suitably capable robot. Health risks from laborious jobs that involve heavy lifting or other repetitive tasks that place undue stress on the body could likewise be greatly reduced.

This general goal of human health and safety motivates why, at a high-level, legged robots are a desirable research avenue, but not why these robots require the ability to plan and execute dynamic aerial motions. To this end, there are two primary motivations for endowing legged robots with such dynamic capabilities. First, the set of obstacles and terrain that the robot is able to traverse becomes vastly larger. To illustrate this point, consider how limited animals like dogs, cheetahs, squirrels, and ibexes would be if they were unable to leap and bound (Fig. 1-1). These animals live in environments dense with challenging obstacles that they must vault over, across, or onto using their strong legs. The second motivation is that the ability to perform dynamic aerial maneuvers broadens the set of tasks the robot can accomplish. For example, a dynamically-capable robot could perform parkour rather than walk exclusively on structured paths when it needs to move quickly, it could leap into the air to look over walls or to grabs things in high places, or it could use its mobility to escape dangerous situations it may find itself in when deployed in place of a human.

1.2 Recent Advancements

Legged robots offer distinct advantages over their wheeled and tracked counterparts. Legged systems do not require constant contact with the ground, they can handle discontinuous surfaces, they are more agile, and they can use their legs for non-locomotive functions such as jumping, kicking, digging, etc. These advantages, however, are only possible due to the increased complexity involved in controlling legged

Figure 1-1: Legged animals that use dynamic aerial motions to conquer their terrain.

systems. The discrete nature of contact results in underactuation that the robot must reason about. Regulating the posture of the robot requires manipulation of the robot's nonlinear dynamics, and the typically high number of degrees of freedoms contained by most robots means that any proposed control strategy will likely be subject to the curse of dimensionality.

Optimization-based approaches are well suited to handle these challenges, and, as such, they have been the most common approach to controlling legged systems [39, 62, 67]. There are, however, two central dilemmas with which optimization-based approaches must contend. The first dilemma has to do with planning contact sequences, i.e. determining when and where to make and break contact with the ground. Including the contact sequence as a variable in the optimization, while possible [20, 46, 59], introduces complex hybrid dynamics and creates a dramatically more complicated optimization problem that is difficult to solve in real-time. Pre-defining the contact sequence, while computationally simpler, tends to result in controllers that are limited in their ability to react to disturbances and are therefore less robust. The second dilemma involves the complexity of the dynamic model used in the optimization.

Using a full-body dynamic model is the most accurate method and can be used to control arbitrary motions of the robot [17, 45, 58], but tends to be prohibitively expensive in terms of computation time. Simplified models such as the spring-loaded inverted pendulum [24] are efficient but involve making assumptions that limit their generality.

Since motion planning using full-body dynamics of the robot typically cannot be performed in real-time, these motion planners are generally paired with online feedback controllers that track the previously planned motions in real-time [18, 28, 40]. In cases where motion planning does happen fast enough to continuously replan trajectories in real-time, model predictive control (MPC)-based approaches are often employed [8]. Sufficiently fast motion replanning can be made possible by model simplification [16], regularization [7], extensive numerical and software engineering [38, 47], or a combination of all three. While MPC-based approaches have produced impressive locomotion results both in simulation and on hardware, they have to date not been extended to controllers capable of producing highly dynamics aerial motions such as jumping. Jumping behavior has been produced on bio-inspired robot designs using relatively simple controllers [44, 75] but these designs do not generalize very well beyond their intended purpose of jumping and are therefore limited in their practicality. Alternatively, graph search-based methods that sample over lower-dimensional motion primitives have proven successful at producing dynamic aerial motions in simulation [15, 61], but restrictive assumptions about controllability, robustness, and kinematics prevent their implementation on hardware. The most common approach to producing jumping motions on quadrupedal and bipedal robots has been trajectory optimization using reduced-order [71] or planar models of the robot's dynamics [48, 74]. Such approaches have been used to achieve acrobatic motions like the back flipping MIT Mini Cheetah [34] and running jumps on the MIT Cheetah 2 [53], but these optimizations were tuned for specific jumps only in the sagittal plane.

Centroidal momentum-based motion planners, on the other hand, have gained popularity recently because they are general enough to produce rich sets of 3D motions while maintaining computational tractability. These motion planners consider

only the centroidal dynamics of the robot [50] rather than the full-body dynamics. Some planners, referred to as “kino-dynamic" planners, simultaneously consider the robot's centroidal dynamics and full joint kinematics [13,30]. By enforcing that these dynamic and kinematic trajectories agree with one another, rich sets of dynamically feasible motions that respect constraints such as end effector placement and collision avoidance can be efficiently generated so long as the actuators are sufficiently powerful to produce the commanded torque profiles.

Alternatively, given a kinematically feasible trajectory for the robot's center of mass and end effectors, a centroidal momentum optimization can compute the set of admissible contact forces necessary to achieve this motion [29]. This centroidal momentum optimization can be posed as a convex problem in both the dynamics [12,57] as well as the contact timings [56]. The implicit assumption that actuators are suitably powerful to carry out contact forces commanded by these centroidal dynamics planners can be restrictively limiting, especially in cases where highly dynamic motions such as running jumps and flips are desired. For this reason, motions demonstrated on hardware using these approaches [9] tend to be slow, quasi-static motions.

These centroidal dynamics-based motion planners treat contact forces as the control inputs for the system dynamics. The mapping of these contact forces to actuator torques (inverse dynamics) along with the mapping of end effector positions to joint positions (inverse kinematics) is typically carried out via a whole-body controller (WBC) [63]. Whole-body control approaches have demonstrated an impressive ability to assist in balancing [37,65] as well as locomotion [19,32] tasks for both humanoid and quadruped robots. Far fewer whole-body control implementations exist, however, for tracking highly dynamic motions, especially motions involving significant rotations of the robot's body. Minimizing centroidal angular momentum is observed to be a useful goal in humanoid balance and locomotion [27], and has thus become a common capability of state-of-the-art WBCs [4, 25]. Tracking motions that involve non-zero centroidal angular momentum, however, is a more difficult task [60]. The approaches that have been proposed for planning and tracking such highly dynamics motions are limited to simulation [14, 70, 76] and typically intended for computer graphics

applications rather than implementation on an actual robot.

An alternative to optimization-based approaches that has emerged in recent years is learning-based motion planning. Learning-based approaches work by collecting large amounts of data through trial and error and using that data to tune, or "train", a controller that optimizes a cost function. The cost function encodes a task such as locomotion or path planning. Learning can be used to produce end-to-end locomotion controllers [66] or can be used selectively to replace a specific component of a controller that is especially difficult for model-based approaches [41]. Although translating learned policies from simulation to real hardware challenged researchers initially, robust solutions to this problem have been proposed [11, 33].

While these learning-based approaches have proven highly effective at conventionally difficult tasks like footstep planning and navigating obstacles stairs [42, 68], hardware demonstrations of these controllers have not shown any especially dynamic behaviors such as leaps or spins. Recent work in the machine learning community has shown that such highly dynamic behaviors can be learned, either from scratch [26, 55] or using motion capture data [54], but the contributions of these works are primarily advancements in training procedures. The approaches are not aimed at implementation on real robots, and as such simplifying assumptions are made and relevant constraints like actuator limits are ignored.

1.3 Contributions

The main contribution of this thesis is a framework for planning and executing highly dynamic aerial motions for legged robots. The framework consists of two key steps. First, a centroidal dynamics-based motion planner produces a dynamically feasible trajectory for the robot along with the set of contact forces necessary to achieve that trajectory. This kino-dynamic planner can efficiently generate 3D aerial motions of robots with arbitrary numbers of legs. The contributions of this thesis to centroidal dynamics-based motion planning are first, the inclusion of constraints that ensure optimized trajectories respect the robot's actuator limits while only marginally in-

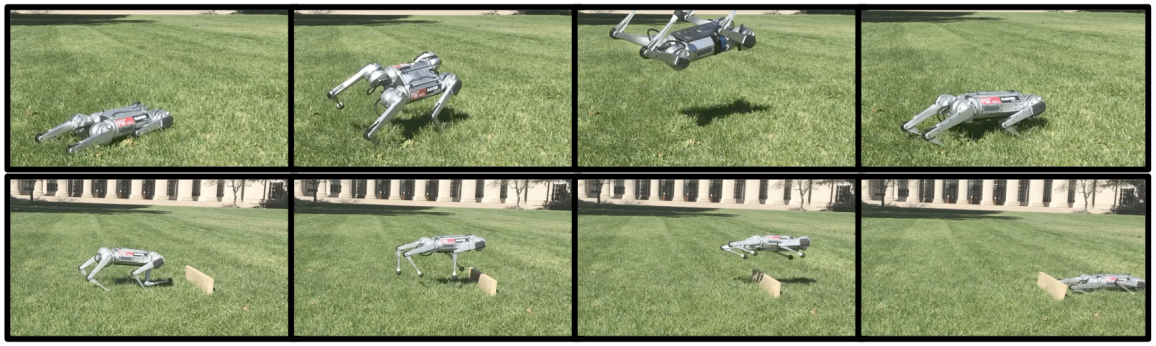

Figure 1-2: Hardware demonstrations of the MIT Mini Cheetah performing dynamic aerial motions.

creasing the solve times, and second, the formulation of a two-stage optimization that uses the output of a simplified first-pass optimization as a warm start for the full kino-dynamic optimization. The inclusion of actuator constraints helps ensure that any planned motions, such as the jumping and barrel rolling motions in Fig. 1-2, will always be feasible on the hardware. The two-stage optimization greatly reduces the time required to generate complex motion plans, nearly enough to run the planner in real time.

The second step in the framework is a prioritized WBC designed specifically to track highly dynamic, or "acrobatic," motions of legged robots. The WBC uses a prioritized task hierarchy that is general to all phases of the planned motions including takeoff, flight, and landing. The task hierarchy is enforced via null-space projection methods that ensure lower priority tasks don't violate higher priority commands. This thesis demonstrates the benefits of a null-space projection-based task prioritization and proposes an optimal task prioritization for tracking dynamic aerial motions. To the author's knowledge, the herein proposed WBC is the first unified WBC capable of tracking a diverse set of dynamic aerial motions, including motions with large rotational components.

The rest of this thesis is organized as follows. Chapter 2 overviews the robotic platforms on which the proposed framework is deployed and introduces the rigid-body dynamics used to model the system. Chapter 3 explains the framework and provides specific technical details about how the actuation-aware kino-dynamic planner and

the WBC are formulated. Chapter 4 presents simulation and hardware results that validate the computational efficiency of the framework and demonstrate its ability to reliably produce dynamic motions. Chapter 5 concludes the work and discusses avenues for potential research.

Chapter 2

System Modeling

The control framework outlined in this work is implemented on two robots, the MIT Mini Cheetah and the MIT Humanoid[1]. For both control and simulation of these robots, rigid body models are used to describe their kinematics and dynamics. Accurate modeling increases the robustness of a controller and minimizes the gap between simulation and real world performance of the robot. This chapter will overview the rigid-body dynamics algorithms used for system modeling as well as describe the robots on which the herein described control framework is implemented.

2.1 Rigid-Body Dynamics

The first step in developing the rigid-body model of the robot is to describe the robot's state. This is done using a kinematic tree representation of the robot that describes (1) the number of bodies/links contained by the robot, (2) how the bodies/links of the robot are connected (in what order and with what type of joint), (3) the positions and orientations of the bodies relative to each other, and (4) the inertial properties of each of the bodies [22]. The configuration of the robot can therefore be described by a complete and independent set of generalized coordinates $\boldsymbol{q} \in \mathbb{R}^n$ corresponding to the positions of each of the joints, where n is the total number of degrees of freedom of the robot. An example of a robot and its kinematic tree representation are shown

[1]Working title for the robot, still in final stages of design.

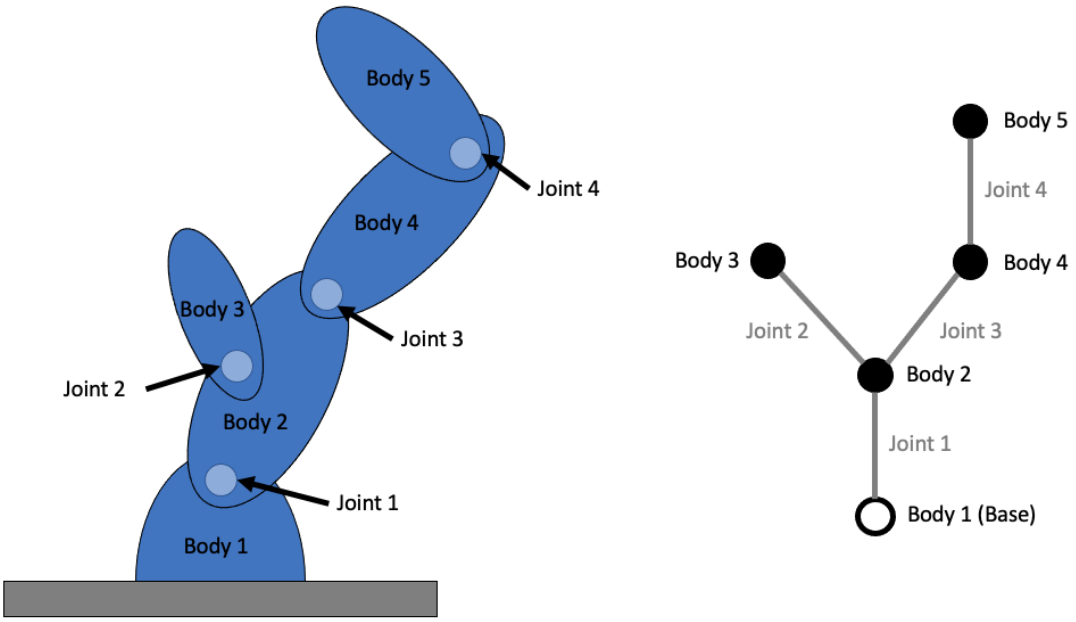

Figure 2-1: Kinematic tree representation of a fixed-base robot.

in Fig. 2-1.

Any robot described in such a manner has dynamics that follow equations of motion given by

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{2.1}$$

where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is the joint-space inertia matrix, $\boldsymbol{C}\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the velocity bias term, $\boldsymbol{G} \in \mathbb{R}^n$ is the gravity term, and $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of generalized forces/torques at each joint. Efficient algorithms such as the recursive Newton-Euler algorithm [51], composite-rigid-body algorithm [23], and the articulated-body algorithm [21] are used throughout this work for computing the forward and inverse dynamics of the robots based solely on their kinematic tree representations.

The example robot shown in Fig. 2-1 is a fixed-base robot, meaning that the robot's base, i.e. the body from which all other links branch from, is the ground. Floating-base systems, such as legged robots, are slightly more complicated than fixed-base systems in that the robot's base is free to move around in space. To account for this nuance, the robot's floating base is treated a single body connected virtually to the ground through a six degree of freedom "virtual joint." The virtual joint describes

the position and orientation of the robot's base relative to some inertial reference frame. Consequently, floating-base systems can be described by an augmented set of generalized coordinates $\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_{fb}^T & \boldsymbol{q}_j^T \end{bmatrix}^T$ where $\boldsymbol{q}_{fb} \in \mathbb{R}^6$ describes the position and orientation of the floating base while $\boldsymbol{q}_j \in \mathbb{R}^n$ describes the positions of each non-virtual joint. The augmented state results in equations of motions that resemble (2.1), but with a few necessary modifications

$$\underbrace{\begin{bmatrix} \boldsymbol{H}_{fb} & \boldsymbol{H}_{fb,j} \\ \boldsymbol{H}_{j,fb} & \boldsymbol{H}_{j,j} \end{bmatrix}}_{\boldsymbol{H}} \underbrace{\begin{bmatrix} \ddot{\boldsymbol{q}}_{fb} \\ \ddot{\boldsymbol{q}}_j \end{bmatrix}}_{\ddot{\boldsymbol{q}}} + \underbrace{\begin{bmatrix} \boldsymbol{C}_{fb} \\ \boldsymbol{C}_j \end{bmatrix}}_{\boldsymbol{C}} \underbrace{\begin{bmatrix} \dot{\boldsymbol{q}}_{fb} \\ \dot{\boldsymbol{q}}_j \end{bmatrix}}_{\dot{\boldsymbol{q}}} + \underbrace{\begin{bmatrix} \boldsymbol{G}_{fb} \\ \boldsymbol{G}_j \end{bmatrix}}_{\boldsymbol{G}} = \underbrace{\begin{bmatrix} \boldsymbol{0}_6 \\ \boldsymbol{\tau}_j \end{bmatrix}}_{\boldsymbol{\tau}} + \underbrace{\sum_{i=1}^{n_c} \boldsymbol{J}_{c,i}^T \mathbf{f}_{c,i}}_{\text{Ground Contact}} . \tag{2.2}$$

The general form of (2.1) is maintained, with the main differences being that the first six entries in $\boldsymbol{\tau}$ are zero since the floating-base virtual joint is not actuated and that the effect of the n_c ground contacts is included through the external spatial force $\mathbf{f}_c \in \mathbb{R}^6$ and the corresponding contact Jacobian $\boldsymbol{J}_c \in \mathbb{R}^{6 \times n+6}$. For clarity, the mass matrix, bias force vector, and gravity vector have been segmented into floating-base components, denoted with subscript fb, and joint-space components, denoted with subscript j. In practice, the same set of rigid-body algorithms mentioned for forward and inverse dynamics of fixed-base systems also applies to floating-base systems.

Note that throughout this work, vectors describing positions and velocities of the robot will be expressed either in the inertial reference frame, denoted $\{\mathcal{I}\}$, or a local frame fixed to a link of the robot, such as the floating-base frame, denoted $\{\mathcal{B}\}$. A preceding superscript such as i or $\mathcal{B}$ means that the vectors they precede are expressed in local $\{i\}$ and $\{\mathcal{B}\}$ frames, respectively. All position and velocity vectors without a preceding superscript are assumed to be expressed in the inertial frame. For example, ${}^{\mathcal{B}}\boldsymbol{x}$ is expressed in the body-fixed frame, ${}^{2}\boldsymbol{x}$ is expressed in the frame local to link 2, and $\boldsymbol{x}$ is expressed in the inertial frame. Recall that the position and orientation of the floating-base, or body-fixed, frame relative to the inertial frame is given by $\boldsymbol{q}_{fb}$. An illustration of these two frames is depicted in Fig. 2-2.

Lastly, an important concept in rigid-body dynamics that is especially relevant to this work is the concept of centroidal momentum. The centroidal momentum of

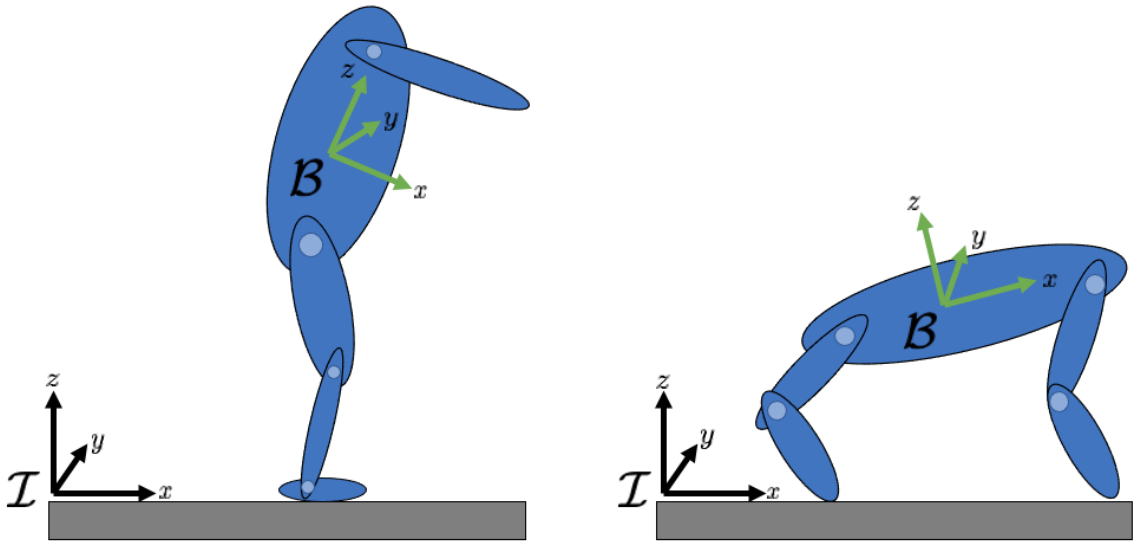

Figure 2-2: Inertial- and body-fixed coordinate frames for two simplified floating-base systems.

a robot is defined as the sum of each individual link's momentum projected on the robot's center of mass (CoM) [49]. Using spatial notation [22], the momentum of an individual link of the robot $\boldsymbol{h}_i \in \mathbb{R}^6$ can be expressed as

$$\boldsymbol{h}_i = \begin{bmatrix} \boldsymbol{\alpha}_i \\ \boldsymbol{l}_i \end{bmatrix} = \boldsymbol{I}_i \mathbf{v}_i, \tag{2.3}$$

where $\boldsymbol{\alpha}_i \in \mathbb{R}^3$ is the angular momentum of the link about the origin of its local frame $\{i\}$, $\boldsymbol{l}_i \in \mathbb{R}^3$ is the linear momentum of the link, $\boldsymbol{I}_i \in \mathbb{R}^{6\times6}$ is the spatial inertia of the link, and $\mathbf{v}_i \in \mathbb{R}^6$ is the spatial velocity of the link. The centroidal momentum of the robot $\boldsymbol{h}_G \in \mathbb{R}^6$, therefore, is given by

$$\boldsymbol{h}_G = \sum_{i=1}^{n} {}^i\boldsymbol{X}_G^T \boldsymbol{h}_i, \tag{2.4}$$

where ${}^i\boldsymbol{X}_G^T$ is a spatial momentum transform [22] from frame $\{i\}$ to the centroidal frame $\{G\}$. The centroidal frame has its origin at the robot's CoM and is parallel to the inertial frame. The generalized velocities of the individual links, given by $\dot{\boldsymbol{q}} \in \mathbb{R}^{n+6}$ are linearly related to the centroidal momentum $\boldsymbol{h}_G$ via the centroidal

momentum matrix (CMM) $\boldsymbol{A}_G \in \mathbb{R}^{6\times n+6}$ by

$$\boldsymbol{h}_G = \boldsymbol{A}_G(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{2.5}$$

By employing the computationally efficient algorithm proposed in [72], the CMM can be computed via

$$\boldsymbol{A}_G = \begin{bmatrix} \boldsymbol{A}_{CAM} \\ \boldsymbol{A}_{CLM} \end{bmatrix} = {}^i\boldsymbol{X}_G^T\boldsymbol{U}_1\boldsymbol{H}, \tag{2.6}$$

where $\boldsymbol{A}_{CAM} \in \mathbb{R}^{3\times n+6}$ is the centroidal angular momentum matrix, $\boldsymbol{A}_{CLM} \in \mathbb{R}^{3\times n+6}$ is the centroidal linear momentum matrix, $\boldsymbol{U}_1 = \begin{bmatrix} \mathbf{1}_6 & \mathbf{0}_{6\times n} \end{bmatrix}$ is a selection matrix, and $\boldsymbol{H}$ is the mass matrix from (2.2).

The rate of change of the centroidal momentum $\dot{\boldsymbol{h}}_G \in \mathbb{R}^6$ is another quantity of interest when doing centroidal momentum-based control. This quantity is related to the generalized velocity and acceleration of the system via

$$\dot{\boldsymbol{h}}_G = \boldsymbol{A}_G\ddot{\boldsymbol{q}} + \dot{\boldsymbol{A}}_G\dot{\boldsymbol{q}} \tag{2.7}$$

where $\dot{\boldsymbol{A}}_G\dot{\boldsymbol{q}}$ is referred to as the centroidal momentum bias force. Again using the algorithms in [72], this vector can be computed via

$$\dot{\boldsymbol{A}}_G\dot{\boldsymbol{q}} = {}^i\boldsymbol{X}_G^T\boldsymbol{U}_1\boldsymbol{C}\dot{\boldsymbol{q}}, \tag{2.8}$$

where $\boldsymbol{C}$ is the velocity bias matrix from (2.2).

2.2 Robot Platforms

The MIT Mini Cheetah and MIT Humanoid are legged robots designed by the MIT Biomimetic Robotics Lab for the purpose of research on legged locomotion and control of legged systems. The designs of both robots are based on the design paradigm of previous MIT Cheetah robots [6, 52, 64], which involves a unique combination of torque dense electric motors, high-bandwidth force control, and the ability to mitigate

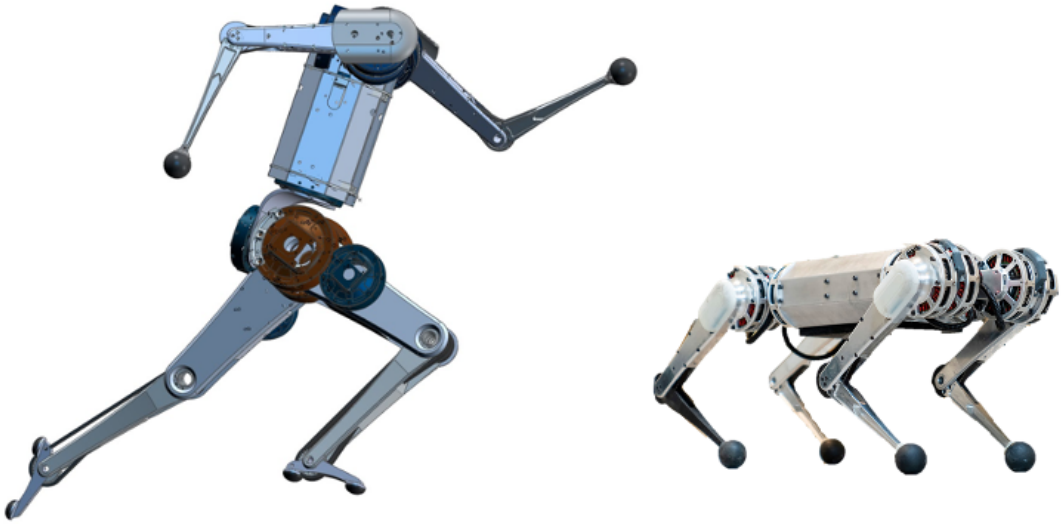

Figure 2-3: Virtual model of the MIT Humanoid (left) and physical model of the MIT Mini Cheetah (right) robots. Images are not to scale with one another.

impacts through backdrivability [73]. The MIT Mini Cheetah [34] is a proven robot that has demonstrated a variety of impressive dynamic behaviors [5, 10, 35, 43]. The MIT Humanoid is currently in the final steps of the design process and at the time of writing has not been physically assembled.

The MIT Mini Cheetah, shown in Fig. 2-3, is a small-scale quadruped robot developed to promote legged locomotion research by providing a durable and relatively inexpensive hardware platform for prototyping and validating proposed control algorithms. The robot stands approximately .3 m tall and weighs only 9 kg. It consists of twelve degrees of freedom that come from ab/adduction, hip, and knee joints on each of its four legs. An important feature of the Mini Cheetah design is that the low inertia of its legs relative to its body. The combined mass of all legs equals roughly 10% of the body mass. The robot can operate completely untethered, with all control, estimation, and planning performed onboard on the Mini Cheetah's low-power UP board computer.

The MIT Humanoid is the first attempt at applying the highly successful design principles of the MIT Cheetah robots to a humanoid robot. A virtual model of the

near-final design is shown in Fig. 2-3. The MIT Humanoid will stand approximately 0.7 m tall and will weigh approximately 46 kg. The legs of the robot each have five degrees of freedom, comprised of a three degree of freedom hip, a knee joint, and a one degree of freedom ankle joint that controls its pitch. The arms of the robot each have three degrees of freedom, comprised of a two degree of freedom shoulder and an elbow joint.

The results presented in this thesis consist of both hardware demonstrations as well as simulation results. All simulation results in this work come from the dynamic simulator developed by the MIT Biomimetic Robotics Lab. The simulator has been designed to replicate hardware conditions as closely as possible. One important feature that contributes to this sim-to-real fidelity is the inclusion of rotor dynamics in the dynamic simulation. Additionally, realistic actuator models based on empirically verified parameters such as torque constant, motor resistance, damping, and dry friction are used to ensure that any behaviors demonstrated in simulation are also possible on the hardware. State estimation is likewise simulated using empirically verified noise parameters for the various sensors on the robot such as IMUs, joint encoders, depth cameras, and localization sensors. Contact with the ground is modeled using an impulse-based contact model that assumes perfectly rigid ground.

2.3 Single Rigid-Body Model

Both robots are easily described using the kinematic tree representation explained in Section 2.1. However, forward and inverse dynamics computations using this full model of the robot can be computationally costly, too costly for controllers that require continuous replanning in real-time. To this end, we often employ a simpler model, the single rigid body model (SRBM), which is more amenable to real-time predictive control. In the SRBM, all bodies/links of the robot are treated a single body such that the state of the robot simply becomes $\boldsymbol{q}_{fb}$. This simplification is illustrated in Fig. 2-4. The design of the MIT Mini Cheetah, and to a lesser extent the MIT Humanoid, are especially amenable to this SRBM simplification because of

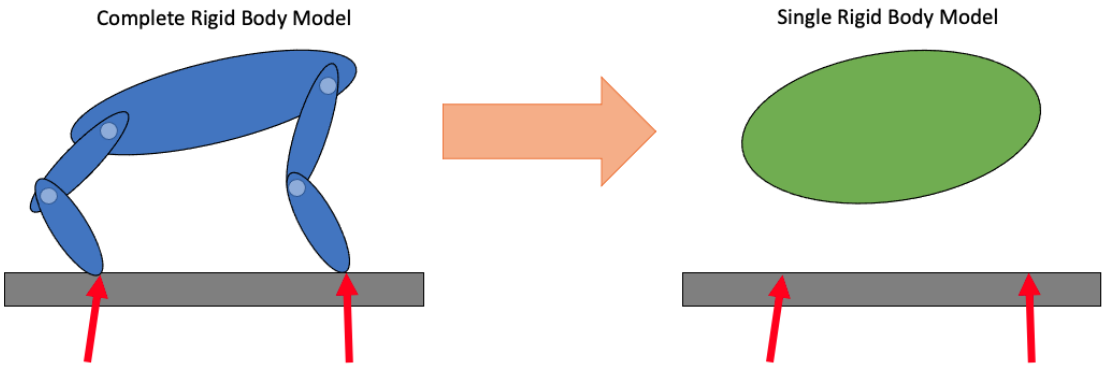

Figure 2-4: Simplification of the robot into a single rigid body model. Red arrows indicate the ground reaction forces at the feet.

their low inertia limbs relative to their torso inertias.

If the orientation of the robot is defined using Euler angles, then the full state of the SRBM can be described by

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{q}_{fb} \\ \dot{\boldsymbol{q}}_{fb} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_c \\ \boldsymbol{\Theta} \\ \mathbf{v}_c \\ {}^{B}\boldsymbol{\omega} \end{bmatrix}, \tag{2.9}$$

where $\mathbf{p}_c \in \mathbb{R}^3$ is the position of CoM, $\boldsymbol{\Theta} \in \mathbb{R}^3$ is the orientation of the body represented in Z-Y-X Euler angles, $\mathbf{v}_c \in \mathbb{R}^3$ is the velocity of the CoM, and ${}^{B}\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity of the body. The resulting dynamics, which are much simpler than (2.2), are given by

$$\dot{\mathbf{x}} = \frac{d}{dt}\begin{bmatrix} \boldsymbol{p}_c \\ \boldsymbol{\Theta} \\ \boldsymbol{v}_c \\ {}^{B}\boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_c \\ \boldsymbol{B}(\boldsymbol{\Theta}){}^{B}\boldsymbol{\omega} \\ \frac{1}{m}\boldsymbol{f} - \boldsymbol{g} \\ {}^{B}\boldsymbol{I}^{-1}(\boldsymbol{R}_b^T\boldsymbol{\tau} - {}^{B}\widehat{\boldsymbol{\omega}}\,{}^{B}\boldsymbol{I}\,{}^{B}\boldsymbol{\omega}) \end{bmatrix}, \tag{2.10}$$

where $\boldsymbol{B}(\boldsymbol{\Theta}) \in \mathbb{R}^{3\times 3}$ is an orientation-dependent matrix that converts angular velocity to Euler angle rates, $m \in \mathbb{R}$ is the mass of the body, $\boldsymbol{f} \in \mathbb{R}^3$ is the net external force

on the CoM, $\boldsymbol{g} \in \mathbb{R}^3$ is the acceleration of the body due to gravity, ${}^B\boldsymbol{I} \in \mathbb{R}^{3\times 3}$ is the inertia tensor about the CoM, $\boldsymbol{R}_b \in SO(3)$ is the rotation matrix from the body frame to the inertial frame, and $\boldsymbol{\tau} \in \mathbb{R}^3$ is the net moment about the CoM. Since the masses of all links are lumped into a single rigid body, the rotational inertia of the body ${}^B\boldsymbol{I}$ is assumed constant.

The reaction forces at each ground contact point dictate the net external wrench delivered to the body. This net wrench, which consists of a net force $\boldsymbol{f}$ and net torque about the CoM $\boldsymbol{\tau}$, is given by

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} = \sum_{i=1}^{n_c} \begin{bmatrix} \mathbf{1}_3 \\ \widehat{\boldsymbol{p}}_i \end{bmatrix} \boldsymbol{f}_i, \tag{2.11}$$

where $\mathbf{1}_3$ is the 3×3 identity matrix, $\boldsymbol{p}_i$ is the position of the ith foot relative to the CoM, $\widehat{\boldsymbol{p}}_i$ maps $\mathbb{R}^3 \to \mathfrak{so}(3)$ such that $\widehat{\boldsymbol{p}}_i \mathbf{y} = \boldsymbol{p}_i \times \mathbf{y}$ for all $\boldsymbol{p}_i, \mathbf{y} \in \mathbb{R}^3$, and $\boldsymbol{f}_i \in \mathbb{R}^3$ is the reaction force at the ith contact point.

Chapter 3

Framework for Dynamic Aerial Motions

The aim of this thesis is to expand the locomotive capabilities of legged robots by introducing a framework for planning and executing highly dynamic aerial motions. While many proven frameworks exist for walking and running, from WBC and MPC-based frameworks to reinforcement learning-based frameworks, no such established framework has emerged for producing dynamic aerial motions such as jumping, flipping, and general acrobatics. Dynamic aerial motions dramatically improve the ability of legged robots to conquer challenging terrain, escape potentially dangerous situations, and perform aerial-based tasks.

The proposed framework, shown in Fig. 3-1, consists of three steps. The first is a motion selector, which, based on an evaluation of the terrain and the task at hand, selects the necessary motion for the robot (e.g. jump onto the table, do a backflip, etc.). In this work, the step is performed by a human operator since the focal point of the thesis is the two subsequent steps: motion planning and motion tracking. The motion planning step consists of a centroidal dynamics-based trajectory optimization that produces a dynamically feasible trajectory for the robot and the required reaction forces at the ground contact points. These planned motion are tracked in the third step by a prioritized WBC that regulates body position, body orientation, centroidal angular momentum, and end effector position.

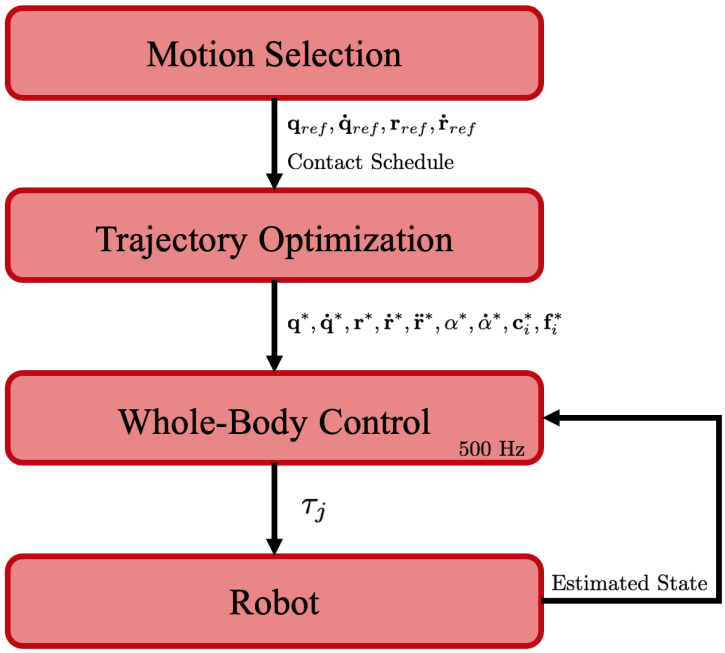

Figure 3-1: Three step framework for achieving dynamic aerial motions of legged robots.

The framework offers the following advantages. First, the framework is general to any legged robot, regardless of size or number of legs. Furthermore, the same formulation of the trajectory optimization as well as the same task hierarchy for the WBC are used for all dynamic motions. Another advantage is that the framework is designed for immediate implementation on hardware. The trajectory optimization respects the actuator limits of the robot, something most centroidal dynamics-based planners neglect, and the WBC has empirically proven itself to be robust to the uncertainties of hardware implementation such as modeling errors, state estimation, and imperfect force control.

3.1 Trajectory Optimization

The role of the motion planner is to generate a dynamically feasible trajectory for the robot and the corresponding set of end effector positions and ground reaction forces that accomplish that motion. Trajectory optimization offers an efficient solution to such a task, as the combination of cost function and constraints can produce a

diverse set of dynamically feasible motions while respecting the constraints of the system. An inherent trade-off exists when selecting the dynamic model upon which to base a trajectory optimization. Increasingly simple models require increasingly restrictive assumptions and limitations while increasingly complex models require increasingly long solve times. A “middle ground" approach that requires minimal limiting assumptions while maintaining tractable solve times involves considering the full joint kinematics of the robot, but only its centroidal dynamics. This approach, referred to as “kino-dynamic" planning [13], greatly reduces the complexity of the dynamics compared to the full-body dynamics, while guaranteeing that the generated motions are kinematically feasible. This section will both overview the formulation of the kino-dynamic optimization used in this work and also detail the two contributions of this thesis to conventional kino-dynamic planning: SRBM-based warm starting and consideration of actuator limits.

3.1.1 Kino-Dynamic Optimization

The optimization variables of the kino-dynamic optimization include discrete trajectories for the generalized position vector of the robot $\boldsymbol{q} \in \mathbb{R}^{n+6}$, the generalized velocity vector for the robot $\dot{\boldsymbol{q}} \in \mathbb{R}^{n+6}$, the position $\boldsymbol{r} \in \mathbb{R}^3$, velocity $\dot{\boldsymbol{r}} \in \mathbb{R}^3$, and acceleration $\ddot{\boldsymbol{r}} \in \mathbb{R}^3$ of the robot's CoM, the robot's centroidal angular momentum $\boldsymbol{\alpha} \in \mathbb{R}^3$ and it's time derivative $\dot{\boldsymbol{\alpha}} \in \mathbb{R}^3$, the positions of the robot's end effector $\boldsymbol{c}_i \in \mathbb{R}^3$, and the external ground reaction forces acting on the end effectors $\boldsymbol{f}_i \in \mathbb{R}^3$. The total number of end effectors, which corresponds to the total number of potential ground contact points considered by the optimization, is given by n_c. Each of these trajectories are discretized into n_t timesteps where n_t is the total number of timesteps and Δt is the time between each timestep. These variables are condensed into a single optimization variable $\boldsymbol{X} \in \mathbb{R}^{(2n+6n_c+27)\times n_t}$ where $\boldsymbol{X} = [\boldsymbol{X}_1, ..., \boldsymbol{X}_{n_t}]$ and

$$\boldsymbol{X}_k = \begin{bmatrix} \boldsymbol{x}_k^T & \boldsymbol{r}_k^T & \dot{\boldsymbol{r}}_k^T & \ddot{\boldsymbol{r}}_k^T & \boldsymbol{\alpha}_k^T & \dot{\boldsymbol{\alpha}}_k^T & \boldsymbol{c}_k^T & \boldsymbol{f}_k^T \end{bmatrix}^T, \tag{3.1}$$

with

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} \boldsymbol{c}_1 \\ \vdots \\ \boldsymbol{c}_{N_c} \end{bmatrix}, \quad \boldsymbol{f} = \begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_{N_c} \end{bmatrix}. \tag{3.2}$$

Note that the contact schedule, i.e. the timing at which end effectors make and break contact with the ground, is not included as an optimization variable. Rather, contact timing is pre-specified and only the locations at which contacts occur are left to the optimization. This is a limiting assumption, and while such an optimization variable could easily be included in the formulation, its inclusion creates an optimization that is dramatically more difficult to solve in a tractable amount of time.

The kino-dynamic optimization can be posed in the following manner,

$$\begin{aligned} \min_{\boldsymbol{X}} \quad & \mathcal{L}(\boldsymbol{X}) \\ \text{s.t.} \quad & \boldsymbol{x}_0 \in \mathcal{X}_0 \\ & \boldsymbol{x}_{n_t} \in \mathcal{X}_{n_t} \\ & \boldsymbol{x} \in \mathcal{K}_x \\ & \text{Centroidal Dynamics (3.5-3.6)} \\ & \text{Joint Kinematics (3.7)} \\ & \text{Momentum Agreement (3.8)} \\ & \text{Contact (3.9-3.10} \\ & \text{Actuator Limits (3.11),} \end{aligned} \tag{3.3}$$

where the goal of the optimization is to minimize a user-specified cost function $\mathcal{L}(\boldsymbol{X})$ subject to constraints including the initial and final poses of the robot $\boldsymbol{x}_0$ and $\boldsymbol{x}_{n_t}$, the robot's joint limits $\mathcal{K}_x$, the centroidal dynamics, the joint kinematics, agreement between the robot's momentum as seen by the centroidal dynamics and the joint kinematics, contact constraints, and actuator limits.

While the formulation is amenable to any form of cost function such as linear, quadratic, nonlinear, etc., a quadratic cost function is chosen due to its simplicity

and the well-defined gradients it produces. The same quadratic form is used for all optimizations, regardless of the task selected by the motion selector. This invariance to task increases the generality of the proposed framework. The result is that given simply a kinematic reference motion (even an infeasible one), a desired final state, and a contact schedule from the motion selector, the trajectory optimization can efficiently generate a rich set of dynamically feasible aerial motions. The cost function, which is aimed at tracking the reference trajectory, is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{X}) = ||\boldsymbol{q}_{\text{ref}} - \boldsymbol{q}||^2_{\boldsymbol{Q}_q} + ||\dot{\boldsymbol{q}}_{\text{ref}} - \dot{\boldsymbol{q}}||^2_{\boldsymbol{Q}_{\dot{q}}} + \\ ||\boldsymbol{r}_{\text{ref}} - \boldsymbol{r}||^2_{\boldsymbol{Q}_r} + ||\dot{\boldsymbol{r}}_{\text{ref}} - \dot{\boldsymbol{r}}||^2_{\boldsymbol{Q}_{\dot{r}}} + \boldsymbol{f}^T \boldsymbol{Q}_f \boldsymbol{f},\end{aligned} \tag{3.4}$$

where $\boldsymbol{Q}_q \in \mathbb{R}^{n+6\times n+6}$, $\boldsymbol{Q}_{\dot{q}} \in \mathbb{R}^{n+6\times n+6}$, $\boldsymbol{Q}_r \in \mathbb{R}^{3\times 3}$, $\boldsymbol{Q}_{\dot{r}} \in \mathbb{R}^{3\times 3}$, and $\boldsymbol{Q}_f \in \mathbb{R}^{3n_c\times 3n_c}$ are weight matrices. The reference trajectory is generated automatically by the motion selector based on the obstacle at hand. Typically, the reference trajectory encodes a simple ballistic motion for the CoM and, if applicable, a rotation of the robot's body.

The first two constraints for the optimization deal with the initial and final positions of the robot. The allowable initial state $\mathcal{X}_0$ is simply the state of the robot at the time of planning, and the allowable terminal state $\mathcal{X}_{n_t}$ depends on the desired motion. For example, when planning a back flip, the terminal state would require that the robot body's be rotated 360° about its pitch axis and that the CoM is at the desired landing height.

The rest of the constraints listed in (3.3) are enforced at each timestep k and are posed as follows. The kinematic limits $\mathcal{K}_x$ in (3.3) simply encode range of motion constraints on each of the joints. More complex kinematic tasks like collision avoidance would require collision checking constraints that ensure the distance between collision points is greater than zero. Such collision constraints tend not to scale well to systems with a large number of potential collision points. Furthermore, such collision checking is not necessary in the aerial motion trajectory optimizations presented in this work, and thus it is not included in the optimization. Instead, collision checking is carried out after the optimization to see if the motion is safe to execute.

The dynamics constraints enforce Euler's laws of motions for rigid bodies according to

$$m\ddot{\boldsymbol{r}}_k = \sum_{i=1}^{n_c} \boldsymbol{f}_{i,k} - m\boldsymbol{g}, \tag{3.5a}$$

$$\dot{\boldsymbol{\alpha}}_k = \sum_{i=1}^{n_c} (\boldsymbol{c}_{i,k} - \boldsymbol{r}_k) \times \boldsymbol{f}_{i,k}. \tag{3.5b}$$

While still nonlinear due to the external torque term, notice how much simpler these equations of motion for the dynamics are compared the full-body equations of motion given by (2.2). Despite their simplicity, the centroidal dynamics still capture the core dynamics of the system without imposing restrictive constraints. Dynamic feasibility is ensured through Euler integration between timesteps. This constraint is given by

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \dot{\boldsymbol{q}}_k \Delta t, \tag{3.6a}$$

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + \dot{\boldsymbol{\alpha}}_k \Delta t, \tag{3.6b}$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k + \frac{\dot{\boldsymbol{r}}_k + \dot{\boldsymbol{r}}_{k+1}}{2} \Delta t, \tag{3.6c}$$

$$\dot{\boldsymbol{r}}_{k+1} = \dot{\boldsymbol{r}}_k + \ddot{\boldsymbol{r}}_k \Delta t. \tag{3.6d}$$

The kinematic constraints that are enforced in this kino-dynamic optimization are

$$\boldsymbol{r}_k = \gamma_{com}(\boldsymbol{q}_k), \tag{3.7a}$$

$$\boldsymbol{c}_{i,k} = \gamma_{c_i}(\boldsymbol{q}_k) \quad \text{for } i = 1, 2, ..., n_c, \tag{3.7b}$$

where $\gamma_{com}(\boldsymbol{q}_k)$ is a function that computes center of mass position based on the configuration of the robot $\boldsymbol{q}_k$ and $\gamma_{c_i}(\boldsymbol{q}_k)$ is a function that computes the location of the ith contact point based on the state of the robot $\boldsymbol{q}_k$. These constraints ensure agreement between centroidal dynamics and full-body kinematics.

An additional constraint that ensures agreement between centroidal dynamics and full-body kinematics is the centroidal angular momentum constraint. The evolution of the robot's centroidal angular momentum is determined by the external wrench

acting on the robot's center of mass (3.5b); however, at each timestep the centroidal angular momentum of the robot must also agree with the centroidal angular momentum computed according to the full-body kinematics of the robot. This constraint is given by

$$\boldsymbol{\alpha}_k = \boldsymbol{A}_{CAM}(\boldsymbol{q}_k)\dot{\boldsymbol{q}}_k, \tag{3.8}$$

where $\boldsymbol{A}_{CAM} \in \mathbb{R}^{3\times(N+6)}$ is the centroidal angular momentum matrix computed according to the steps outlined in Chapter 2.1.

The contact constraints at each timestep depend on whether a given end effector is in or out of contact with the ground. When the ith end effector is in contact with the ground it is subject to the following constraints

$$\boldsymbol{c}_{i,k} = \boldsymbol{c}_{i,k-1}, \tag{3.9a}$$

$$-\mu f_{i,z} \leq f_{i,x} \leq \mu f_{i,z}, \tag{3.9b}$$

$$-\mu f_{i,z} \leq f_{i,y} \leq \mu f_{i,z}, \tag{3.9c}$$

where μ is the coefficient of static friction between the end effector and the ground. Conversely, when it is out of contact, it is subject to the following constraint

$$c_{i,z} \geq \gamma_{gnd}(c_{i,x}, c_{i,y}), \tag{3.10}$$

where γ_{gnd} maps the xy position of the end effector to the height of the ground at that point. Since the contact schedule is known beforehand, these constraints are easily enforced because the timesteps at which they are applied is fixed.

The last set of constraints are novel constraints for kino-dynamic planning. Most kino-dynamic optimizations assume that the actuators are suitably strong to produce any commanded reaction force at a given end effector. To date, kino-dynamic motion planners have primarily been used for slow moving motions of robots with very powerful actuators [30,56]. In these cases, this assumption is generally valid. However, in the case of highly dynamic aerial motions for the MIT Mini Cheetah and Humanoid, the desired motions typically push the robot's hardware capabilities to their limits.

As a result, actuators constraints have been added to the kino-dynamic optimization, but done so in a way that minimally impacts the complexity of the problem. The constraints are based on the torque-speed limitations of the robot's electric motors and are given by

$$\dot{\boldsymbol{q}}_{j,min} \leq \dot{\boldsymbol{q}}_{j,k} \leq \dot{\boldsymbol{q}}_{j,max}, \tag{3.11a}$$

$$\boldsymbol{\tau}_{min} \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_{max}, \tag{3.11b}$$

$$\boldsymbol{V}_{min} \leq \underbrace{\frac{\boldsymbol{\tau}_k}{\boldsymbol{k_t g_r}}}_{\substack{\text{Desired} \\ \text{Current}}} \boldsymbol{R} + \underbrace{\dot{\boldsymbol{q}}_k \boldsymbol{g_r} \boldsymbol{k_t}}_{\text{Back EMF}} \leq \boldsymbol{V}_{max}, \tag{3.11c}$$

where $\dot{\boldsymbol{q}}_j \in \mathbb{R}^n$ is the vector of joint velocities, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of joint torques, $\boldsymbol{k_t} \in \mathbb{R}^n$ is the vector of torque constants for each motor, $\boldsymbol{g_r} \in \mathbb{R}^n$ is the vector of gear reductions for each motor, $\boldsymbol{R} \in \mathbb{R}^n$ is the vector of motor resistances for each motor, $\boldsymbol{V} \in \mathbb{R}^n$ is the vector of voltage limits for each motor. Note that these constraints are specific to robots actuated via electric motors.

Recall that $\boldsymbol{\tau}$ is not an optimization variable. So while this constraint is written as a constraint on joint torques and joint velocities, in the optimization the constraint is actually a function of joint position, joint velocity, and ground reaction forces. To understand this relationship, consider a reformulation of (2.2) where the unactuated portions of the equations of motions, the floating base portion, is excluded,

$$\begin{bmatrix} \boldsymbol{H}_{j,fb} & \boldsymbol{H}_{j,j} \end{bmatrix} \ddot{\boldsymbol{q}} + \boldsymbol{C}_j \dot{\boldsymbol{q}} + \boldsymbol{G}_j = \boldsymbol{\tau}_j + \sum_{i=1}^{n_c} \boldsymbol{J}_{c,i}^T \mathbf{f}_{c,i}. \tag{3.12}$$

If the generalized positions, velocities, and accelerations of the system are known along with the ground reaction forces, then the corresponding joint torques needed to achieve that motion are given by

$$\boldsymbol{\tau}_j = \begin{bmatrix} \boldsymbol{H}_{j,fb} & \boldsymbol{H}_{j,j} \end{bmatrix} \ddot{\boldsymbol{q}} + \boldsymbol{C}_j \dot{\boldsymbol{q}} + \boldsymbol{G}_j - \sum_{i=1}^{n_c} \boldsymbol{J}_{c,i}^T \mathbf{f}_{c,i}. \tag{3.13}$$

However, the kino-dynamic optimization does not consider joint accelerations, and

as a result the exact required joint torques cannot be computed without introducing considerable complexity to the problem. Empirically, it was found that the reaction force terms $\boldsymbol{J}_{c,i}^T \mathbf{f}_{c,i}$ dominate (3.13) in the sense that these terms are of significantly larger magnitude than the acceleration and bias terms. Based on this observation, the joint torques for the robot can be approximated via

$$\boldsymbol{\tau}_j \approx -\sum_{i=1}^{n_c} \boldsymbol{J}_{c,i}^T \mathbf{f}_{c,i}. \tag{3.14}$$

Note that for limbs that are not in contact with the ground, since there are no reaction forces to create torques, only the constraint in (3.11a) is enforced in the optimization.

Even with the approximation in (3.14), however, the relationship between forces and joint torques is still very nonlinear and thus the inclusion of this constraint results in a non-trivial increase in solves times. To reduce this increase in solve time, an observation about the kinematic trajectories of the leg joints was leveraged. It was observed that for all motion tasks that were attempted on either robot (jumps, flips, spins, etc.), the hip joints responsible for abduction/adduction and medial/lateral rotation experienced minimal displacements throughout the course of the motions, rarely greater than 8°. Furthermore, the contact Jacobians for the legs were minimally sensitive to changes in these joint angles. Based on these observations, the contact Jacobians can be well approximated by assuming that joints responsible for abduction/adduction and medial/lateral rotation are fixed in a reference position while only the joints responsible for hip, knee, and ankle flexion change throughout motion. This approximate contact Jacobian depends on fewer optimization variables and therefore results in simpler constraints. The results presented in Chapter 4 validate these approximations for the actuator constraints as well as demonstrate their advantages.

The final step in the proposed kino-dynamic trajectory optimization stage of the framework is a novel feature that significantly reduces the time required to converge to an optimal solution. This step involves performing a centroidal dynamics-based trajectory optimization on a SRBM of the robot as a first pass motion plan. This

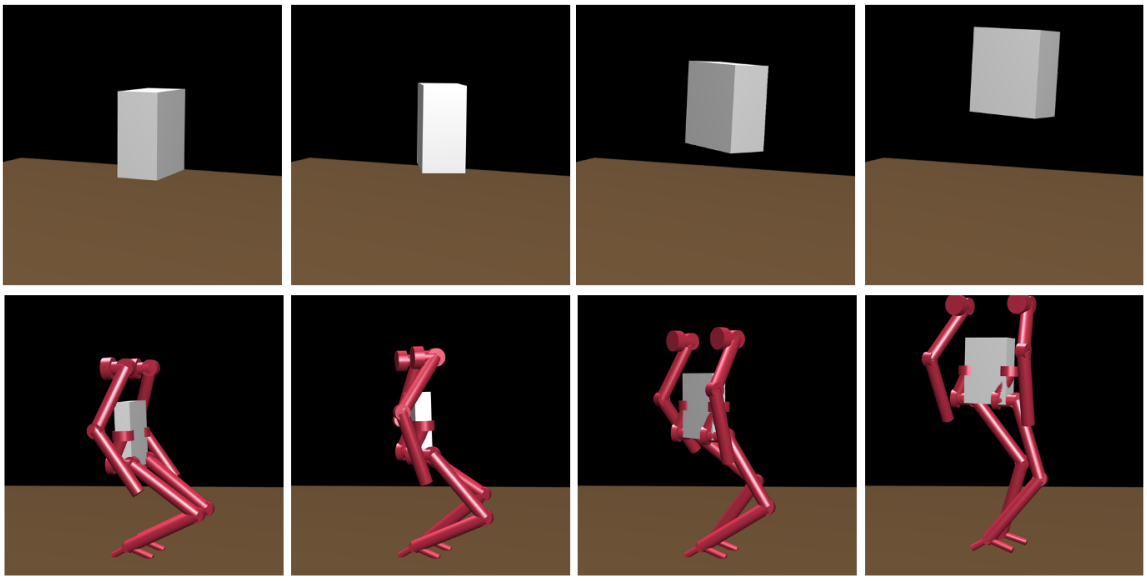

Figure 3-2: A single rigid-body model optimization is performed first (top) and its result is used to warm start the kino-dynamic optimization (bottom).

optimization, which typically takes less than a second to solve, is used as a warm start for the actuator-aware kino-dynamic optimization described thus far in this section. Although the SRBM optimization can only approximately respect the kinematic and actuator limits of the robot, it captures the core dynamics of the desired motion (Fig. 3-2) and is able to reduce net solve times by an average of 78.7%.

3.1.2 Single Rigid Body Optimization

Solving large nonlinear optimizations is a challenging process. Solvers tend to be sensitive to the starting point that they are provided. Consequently, “cold starting" an optimization, i.e. providing a starting point of zero for all optimization variables, typically results in much longer solve times than an educated starting point, i.e. a “warm start." Furthermore, cold starts are more prone to low quality solutions as a result of getting stuck in local minima. Rather than warm starting by manually authoring starting points based on intuition, the herein proposed motion planner uses a two-stage optimization where the warm start for the main kino-dynamic optimization comes from an initial, reduced order optimization.

The reduced-order optimization uses the SRBM and therefore does not consider

the joint level kinematics of the robot. Because of this simplification, certain restrictions apply, but those restrictions are acceptable because the aim of this first pass optimization is not a precise motion plan, but rather, a robust starting point for the main optimization. The first restriction is that, since joint positions are not considered, only approximate kinematic constraints based on the positions of the limb attachment points (hips and shoulders) relative to the end effectors (feet and hands) can be enforced. Second, since joint velocities are not considered, actuator constraints on the torque-speed limits of the motors cannot be enforced. Conservative estimates for maximum ground reaction force must instead be used in the optimization.

The optimization variables for the SRBM optimization include discrete trajectories for the SRBM state $\mathbf{x} \in \mathbb{R}^{12}$ given by (2.9), the positions of the robot's end effector $\boldsymbol{c}_i \in \mathbb{R}^3$, and the external ground reaction forces acting on the end effectors $\boldsymbol{f}_i \in \mathbb{R}^3$. The trajectories are again discretized into n_t timesteps separate by increments of Δt. The variables are condensed into a single optimization variable $\boldsymbol{X}_{SRBM} \in \mathbb{R}^{(12+6n_c)\times n_t}$ where $\boldsymbol{X}_{SRBM} = [\boldsymbol{X}_{SRBM,1}, ..., \boldsymbol{X}_{SRBM,n_t}]$ and

$$\boldsymbol{X}_{SRBM,k} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{c}_k \\ \boldsymbol{f}_k \end{bmatrix}, \tag{3.15}$$

with $\boldsymbol{c}_k \in \mathbb{R}^{3n_c}$ and $\boldsymbol{f}_k \in \mathbb{R}^{3n_c}$ given by (3.2).

The SRBM optimization is formulated identically to the kino-dynamic optimization (3.3), with the only difference being the specifics of how the constraints are imposed on the new definition of the robot's state. The same quadratic cost function form is used, but it is modified to account for the new optimization variables

$$\mathcal{L}_{SRBM}(\boldsymbol{X}_{SRBM}) = ||\mathbf{x}_{\text{ref}} - \mathbf{x}||^2_{\boldsymbol{Q}_x} + \boldsymbol{f}^T \boldsymbol{Q}_f \boldsymbol{f}, \tag{3.16}$$

where $\boldsymbol{Q}_x \in \mathbb{R}^{12\times 12}$ is another weight matrix. Importantly, the same reference trajectory, terminal constraints, and contact schedule, which come from the motion selector, are supplied to both the kino-dynamic optimization as well as the first-pass SRBM

optimization. This contributes to the generality of the framework since it allows warm starts to be generated automatically rather than manually.

The new dynamics constraint is given by (2.10) and is again used in an Euler integration constraint

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \dot{\mathbf{x}}_k \Delta t. \tag{3.17}$$

Since joint kinematics are not explicitly considered, there are no joint level constraints, but approximate constraints are placed on the maximum distance of an end effector relative to its attachment point on the torso

$$||\boldsymbol{c}_i - \boldsymbol{c}_{i,attach}|| \leq \beta \ell_i, \tag{3.18}$$

where $\boldsymbol{c}_{i,attach} \in \mathbb{R}^3$ is the position of the attachment point on the torso for the ith limb, $\ell_i \in \mathbb{R}$ is the maximum kinematically feasible distance from attachment point to end effector for the ith limb, and $\beta \in \mathbb{R}$ is a fixed scaling factor that determines how conservative the constraint is.

For limbs in contact with the ground, the contact constraints remain identical to the kino-dynamic case (3.9a)-(3.9c). No constraints, however, are placed on the out of contact limbs since they do not show up in the dynamics. The final set of constraints, the actuator constraints, are simplified to simple boundary constraints

$$\boldsymbol{f}_{i,min} \leq \boldsymbol{f}_{i,k} \leq \boldsymbol{f}_{i,max}. \tag{3.19}$$

This SRBM optimization is still nonlinear in both the dynamics constraint as well as the approximate kinematic constraint, yet the dimension of the problem is greatly reduced and the most troublesome nonlinear constraint, the joint kinematics constraint, is removed. The output of this first-pass optimization provides a rough approximation of the desired floating-base motion and the reaction forces required to generate that motion. Using this as a starting point, the kino-dynamic optimization can converge to a solution more rapidly because the optimization primarily involves refining the kinematics to agree with the warm start rather than finding all compo-

nents of the trajectory from scratch.

3.2 Whole-Body Control

Whole-body control is a popular approach for controlling both humanoid and quadruped robots. However, it is typically used to control slow moving tasks or tasks that do not involve significant rotations of the robot's body. Consequently, whole-body control tasks and objective functions involving the centroidal angular momentum of the robot typically involve minimizing this quantity. Regulating non-zero angular momentum to produce motions such as back flips, barrel rolls, and aerial spins, however, is a considerably more difficult problem. A general WBC capable of handling such rotations through centroidal momentum tracking is one of the primary contributions of this thesis. The whole-body controller is general in the sense that the same task hierarchy is used to track all motions on all robots in all stages of motion (i.e. takeoff, flight, and landing). This section details the WBC implementation and provides an analysis of the effect of different task hierarchies on optimal WBC performance.

3.2.1 Whole-Body Impulse Control

The WBC used in this work follows the design of the whole-body impulse controller (WBIC) demonstrated previously on the MIT Mini Cheetah [35]. The WBIC is a quadratic programming-based WBC that differs from conventional whole-body control in that it is designed to track commanded reaction forces and body trajectories simultaneously rather than strictly body trajectories. The commanded reaction forces, $\boldsymbol{f}_r^{\text{cmd}} \in \mathbb{R}^{3n_c}$, come from a higher-level planner such as a model predictive controller or, in this case of this work, a trajectory optimization-based motion planner. The body trajectories are described in terms of commanded generalized accelerations $\ddot{\boldsymbol{q}}^{\text{cmd}} \in \mathbb{R}^{n+6}$ that come from a prioritized task hierarchy. Tasks in the hierarchy are

given in the form

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_x \dot{\boldsymbol{q}}, \tag{3.20a}$$

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}_x \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_x \dot{\boldsymbol{q}}, \tag{3.20b}$$

where $\boldsymbol{x} \in \mathbb{R}^m$ represents the task's operational coordinates and $\boldsymbol{J}_x \in \mathbb{R}^{m \times n}$ is the task Jacobian that maps generalized velocities to velocities in these operational coordinates. Commands for generalized accelerations $\ddot{\boldsymbol{q}}^{\text{cmd}}$ come from commanded task space accelerations $\ddot{\boldsymbol{x}}^{\text{cmd}}$. For example, when an end effector is scheduled to be in contact with the ground, it is given a commanded task space acceleration of zero. The resulting commanded generalized acceleration is given by

$$\ddot{\boldsymbol{q}}^{\text{cmd}} = \overline{\boldsymbol{J}_c^{\text{dyn}}}(-\boldsymbol{J}_c \dot{\boldsymbol{q}}), \tag{3.21}$$

where $\overline{\boldsymbol{J}_c^{\text{dyn}}}$ is the dynamically consistent inverse [36] of the contact Jacobian. Contact tasks such as this, if they exist, are always given highest task priority to ensure that desired contacts are maintained. Subsequent tasks such as body posture, centroidal momentum, and joint posture tasks generally get their commanded task space acceleration from feedback laws of the form

$$\ddot{\boldsymbol{x}}_j^{\text{cmd}} = \ddot{\boldsymbol{x}}_j^{\text{des}} + \boldsymbol{K}_{p,j}(\boldsymbol{x}_j^{\text{des}} - \boldsymbol{x}_j) + \boldsymbol{K}_{d,j}(\dot{\boldsymbol{x}}_j^{\text{des}} - \dot{\boldsymbol{x}}_j), \quad j = 1, \ldots, n_{tk}, \tag{3.22}$$

where $\boldsymbol{K}_{p,j} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{K}_{d,j} \in \mathbb{R}^{m \times m}$ are gain matrices for the jth task and n_{tk} denotes the total number of tasks. The commanded generalized acceleration $\ddot{\boldsymbol{q}}^{\text{cmd}}$ is then updated by projecting these subsequent task accelerations through the null space of higher priority tasks and constraints [36],

$$\ddot{\boldsymbol{q}}_j^{\text{cmd}} = \ddot{\boldsymbol{q}}_{j-1}^{\text{cmd}} + \overline{\boldsymbol{J}_{j|pre}^{\text{dyn}}}(\ddot{\boldsymbol{x}}_j^{\text{cmd}} - \dot{\boldsymbol{J}}_j \dot{\boldsymbol{q}} - \boldsymbol{J}_j \ddot{\boldsymbol{q}}_{j-i}^{\text{cmd}}), \tag{3.23}$$

where $\overline{\boldsymbol{J}_{j|pre}^{\text{dyn}}}$ is the dynamically consistent inverse of the task Jacobian projected through the null space of all previous task Jacobians. The first commanded acceler-

ation, $\ddot{\boldsymbol{q}}_0^{\text{cmd}}$, generally comes from the contact constraint task (3.21). This null space projection method ensures that lower priority task never command accelerations that will lead to violation of higher priority tasks.

In conventional whole-body control, once commands for the generalized accelerations are set, a quadratic program (QP) is solved to determine the ground reaction forces necessary to achieve the floating base portion of that commanded acceleration. However, that is not the case in WBIC. Instead, since ground reaction forces are also commanded, a QP instead solves for the minimum deviation from these commanded reaction forces and generalized accelerations. This QP is subject to the constraints that the resulting motion of the robot must respect the floating-base dynamics of the system and the final reaction forces must respect contact constraints such as friction. The formulation of the QP is given by

$$\begin{aligned} \min_{\boldsymbol{\delta}_q, \boldsymbol{\delta}_f} \quad & \boldsymbol{\delta}_q^T \boldsymbol{Q}_1 \boldsymbol{\delta}_q + \boldsymbol{\delta}_f^T \boldsymbol{Q}_2 \boldsymbol{\delta}_f \\ \text{s.t.} \quad & \boldsymbol{S}_{fb}(\boldsymbol{H}\ddot{\boldsymbol{q}} + \boldsymbol{C}) = \boldsymbol{S}_{fb}\boldsymbol{J}_c^T \boldsymbol{f}_r \\ & \ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}^{\text{cmd}} + \begin{bmatrix} \boldsymbol{\delta}_q \\ \boldsymbol{0}_n \end{bmatrix} \\ & \boldsymbol{f}_r = \boldsymbol{f}_r^{\text{cmd}} + \boldsymbol{\delta}_f \\ & \boldsymbol{W}\boldsymbol{f}_r \geq \boldsymbol{0}, \end{aligned} \tag{3.24}$$

where $\boldsymbol{\delta}_q \in \mathbb{R}^6$ is a variation to the floating base portion of the commanded generalized acceleration, $\boldsymbol{\delta}_f \in \mathbb{R}^{3n_c}$ is a variation to the commanded reaction forces, $\boldsymbol{Q}_1 \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{Q}_2 \in \mathbb{R}^{3n_c \times 3n_c}$ are weight matrices, $\boldsymbol{S}_{fb} \in \mathbb{R}^{6\times n+6}$ is a selection matrix that selects only the floating base portion of the dynamics, $\boldsymbol{J}_c \in \mathbb{R}^{3n_c \times n+6}$ is the augmented contact Jacobian, and $\boldsymbol{W} \in \mathbb{R}^{5n_c \times 3n_c}$ is the matrix version of the contact constraints in (3.9b) and (3.9c). The allowable variation from the commanded acceleration in (3.23) enables the WBIC to be used during periods of underactuated or even no actuation (e.g. flight) without violating the constraints of the QP. Furthermore, the QP formulation is compact, making it amenable to real-time control applications.

3.2.2 Task Prioritization

Strict task prioritization via null space projection has an advantage over cost function-based prioritization for whole-body control because tuning cost functions to find a proper set of task weights general to all possible motions can be rather tedious, and in some cases impossible. Null space projection methods, on the other hand, ensure that lower priority tasks never violate higher priority tasks, which is beneficial when a specific component of the motion like end effector position or body orientation is the most important. However, finding the optimal task hierarchy is also a non-trivial process, especially for the case of general dynamic motions like the ones examined in this thesis. The remainder of this section will present the tasks and task hierarchy used by the herein proposed WBC for dynamic aerial motions. This formulation of the WBIC, which constitutes one of the primary contributions of this thesis, enables the same WBC to be used for all stages of motion including takeoff, flight, and landing.

Based on the outputs of the trajectory optimizations (3.1), the possible tasks for the WBC to track are the joint trajectories, the end effector trajectories, the floating-base trajectory, and the centroidal momentum trajectory. These tasks are defined according to the following relationships. The joint task is given by

$$\underbrace{\dot{\boldsymbol{q}}_j}_{\dot{\boldsymbol{x}}_{\text{task}}} = \underbrace{\begin{bmatrix}\mathbf{0}_{n\times 6} & \mathbf{1}_n\end{bmatrix}}_{\boldsymbol{J}_{\text{task}}} \dot{\boldsymbol{q}}. \tag{3.25}$$

The end effector task is given by

$$\underbrace{\dot{\boldsymbol{c}}_i}_{\dot{\boldsymbol{x}}_{\text{task}}} = \underbrace{\boldsymbol{J}_{c,i}}_{\boldsymbol{J}_{\text{task}}} \dot{\boldsymbol{q}}. \tag{3.26}$$

The floating-base task is given by

$$\underbrace{\begin{bmatrix}\boldsymbol{\omega}_{fb}\\ \boldsymbol{v}_{fb}\end{bmatrix}}_{\dot{\boldsymbol{x}}_{\text{task}}} = \underbrace{\begin{bmatrix}{}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times n}\\ \mathbf{0}_{3\times 3} & {}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}} & \mathbf{0}_{3\times n}\end{bmatrix}}_{\boldsymbol{J}_{\text{task}}} \dot{\boldsymbol{q}}, \tag{3.27}$$

where $\boldsymbol{\omega}_{fb} \in \mathbb{R}^3$ and $\boldsymbol{v}_{fb} \in \mathbb{R}^3$ are the angular and linear velocities of the robot's floating base, respectively, and ${}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}}$ is the rotation matrix from the body frame to the inertial frame. Lastly, the centroidal momentum task is given by

$$\underbrace{\boldsymbol{h}_G}_{\dot{\boldsymbol{x}}_{\text{task}}} = \underbrace{\boldsymbol{A}_G}_{\boldsymbol{J}_{\text{task}}} \dot{\boldsymbol{q}}. \tag{3.28}$$

Recall that the first task in the whole-body control hierarchy is always the contact task for any limbs that are in contact with the ground. This task ensures that limbs intended to be in contact do not break contact. The first non-contact task in the hierarchy must span the centroidal dynamics of the robot, which is equivalent to spanning the floating base dynamics [36]. This is required in order for the QP (3.24) to properly solve for ground reaction forces. Thus, from (3.25)-(3.28) we see that the options for the first task in the hierarchy are floating-base control or centroidal momentum control. The challenge in tracking centroidal momentum, specifically centroidal angular momentum, as the first task in the hierarchy is that there is no position analog $\boldsymbol{x}_{\text{task}}$ that corresponds to the task velocity $\dot{\boldsymbol{x}}_{\text{task}}$. In other words, while the integral of the robot's centroidal linear momentum corresponds to the COM position of the robot, the integral of the robot's centroidal angular momentum may not correspond to any meaningful physical quantity [60]. Without such a position analog, it is difficult to formulate a feedback law for commanding task accelerations. The standard formulation for such a feedback law (3.22) is not applicable.

Table 3.1: WBIC Task Prioritization

Task 0	Contact
Task 1	Floating Base
Task 2	Centroidal Angular Momentum

For these reasons, the following task prioritization, shown in Table 3.1, is used for all robots. The first task in the hierarchy is the contact task, which is essentially an end effector task intended to either maintain contact between the feet and the ground or track the positions of the feet during flight phases. The floating-base task is chosen as the first non-contact task in the hierarchy because not only does this

task have a position analog that is physically meaningful, but, furthermore, the task directly regulates the posture of the robot. This is beneficial because maintaining the desired posture of the floating-base is crucial in preventing the robot from falling down. Unlike the centroidal momentum task, however, the floating-base task does not span the full configuration space, only the floating-base portion. This means that the joint states of any non-contact limbs (e.g. the arms of the humanoid) are in the null space of the contact tasks and the first task in the hierarchy. These joint states can therefore be fully utilized in subsequent tasks. As a result, the centroidal angular momentum task is placed as the second non-contact task in the hierarchy. If the robot has non-contact limbs such as arms, this angular momentum task results in the emergence of limb motions that assist in either generating body rotations for desired jumps or damping out centroidal angular momentum to help stabilize landing. Consequently, no subsequent joint position or end effector position tasks need to be specified for the non-contact limbs since their motion is purely emergent. This set of task spans the full configuration space of the robot, which means that the WBIC will always be able to solve for the set of joint torques that best achieve the commanded motion.

The floating-base and end effector task acceleration commands take the form of the general feedback law in (3.22), while the centroidal angular momentum task acceleration command is given by

$$\dot{\boldsymbol{\alpha}}_G^{\text{cmd}} = \dot{\boldsymbol{\alpha}}_G^{\text{des}} + \boldsymbol{K}_{d,CAM}(\boldsymbol{\alpha}_G^{\text{des}} - \boldsymbol{\alpha}_G). \tag{3.29}$$

This prioritization of floating-base task ahead of centroidal angular momentum task strikes the proper balance between these two tasks. The WBIC QP solves for the ground reaction forces that produce the desired motion of the floating-base as closely as possible. These reaction forces may produce a centroidal angular momentum of the robot that violates the commanded centroidal angular momentum, but this is acceptable because this command serves to assist in stabilizing body rotation rather than explicitly regulate it.

To further justify this prioritization, consider the alternative options. In the case where centroidal momentum is prioritized first, the ground reaction forces would satisfy this command without any regard for the floating-base orientation. Since the centroidal momentum task spans the full configuration space of the robot, no subsequent tasks could be employed to correct the inevitable errors in body orientation produced by the centroidal angular momentum tracking. The other option would be to place these two tasks on the same priority level by combining them into a single task. This approach is not desirable because performance then becomes dependent on relative gain tuning, and it is typically intractable to find a set of gains that performs well for a diverse set of desired motions.

Chapter 4

Results

The proposed framework for planning and executing dynamic aerial motions was tested extensively in simulation on the MIT Humanoid and both in simulation and on hardware for the MIT Mini Cheetah. The custom simulator used to produce the results presented in this section is described in Chapter 2.2. Dynamic simulations as well as all motion planning optimizations are run on a laptop computer with an Intel i7 processor. When the framework is deployed on hardware, motion planning still takes place off-board on this same laptop computer because the Mini Cheetah's low power onboard computer is unable to handle even modestly-sized nonlinear optimizations. Planned trajectories are sent to the robot via Lightweight Communications and Marshalling [31]. The WBC used the track these planned trajectories, however, is run on the Mini Cheetah's onboard computer since the QP it solves is much less computationally costly.

The kino-dyanmic optimization is solved in `MATLAB` using `CasADI` [2], an open-source optimization toolbox built for optimal control applications. `CasADI` constructs the optimization problem and provides a symbolic framework for computing gradients, Jacobians, and Hessians via algorithmic differentiation. The optimization itself is sovled using `IPOPT` [69], a software library for large scale nonlinear optimization that implements a primal-dual interior point method to find locally optimal solutions. The kino-dynamic optimization is only used to generate a trajectory for the takeoff portion of a planned motion. For a given robot, the flight and landing commands

supplied to the WBC are common to all motions. In flight, limbs used for landing are given position commands that prepare them for landing. No contact detection has currently been implemented on the hardware, so the transition from flight commands to landing commands is currently time-based. The landing commands specify that the robot's CoM stays at a set height in the center of its support polygon and that its body has zero roll or pitch.

4.1 MIT Mini Cheetah

The framework has enabled the MIT Mini Cheetah to perform a variety of novel behaviors including jumps in all direction, 180° spins, barrel rolls, front flips, and running jumps from trot and bounding gaits. All cost function weights and WBC gains are identical for all of these motions. An Euler integration time step of $\Delta t = 0.01$ is used for the dynamics constraint (3.6).The kino-dynamic planner is able to generate all of these motions plans in less than 5.0s. Furthermore, the herein proposed actuation-aware kino-dynamic planner is able to generate motions at roughly the same rate as a conventional kino-dynamic planner with ensuring dynamic feasibility. To prove this, a benchmark comparison is performed on various kino-dynamic planning options.

The first option is standard kino-dynamic (KD) planning as described in [13]. The second option is actuation-aware kino-dynamic (AAKD) planning, which is identical to KD with the addition of actuator constraints (3.11a)-(3.11c). The third option is approximate actuation-aware planning (AAKD*), which is identical to AAKD except approximate contact Jacobians are used as described in Chapter 3.1.1. Furthermore, these planning optimization are compared when they are solved with and without the SRBM warm start (WS). The comparison of these planning options for various motions of the Mini Cheetah is shown in Table 4.1. The benchmarking results show not only that the proposed AAKD*+WS optimization is efficient, only about an order of magnitude away from being viable for real-time predictive control, but also that the actuator constraints increase nominal solve times by only 30% on average. Addi-

tionally, as subsequent results will show, using the approximate actuator constraints results in rare and essentially negligible violations of actuator limits.

Table 4.1: Solve Times for Various Kino-Dynamic Approaches to Mini Cheetah Motion Planning

	180° Spin	Barrel Roll	Front Flip	Forward Jump	Left Jump
KD	11.03s	11.95s	12.16s	5.20s	5.95s
KD+WS	1.67s	3.77s	4.16s	1.82s	1.85s
AAKD	50.42s	30.72s	20.05s	13.33s	14.22s
AAKD+WS	2.50s	4.92s	4.61s	2.87s	3.11s
AAKD*	19.19s	12.18s	30.18s	11.59s	12.74s
AAKD*+WS	2.32s	4.91s	4.55s	2.70s	2.81s

Jumping

The robot, starting from a static pose, is able to jump onto and off of surfaces up to 16cm in any direction. To produce jumping motions, kinematic reference motions are used in the cost function that guide the robot's CoM to the desired landing position. Furthermore, terminal constraints are placed on the optimization such that the robot's CoM velocity at takeoff will result in a ballistic CoM trajectory that ends in the desired landing position. Unlike previous approaches that allow for only planar jumping, the kino-dynamic motion planner can produce 3D jumps in any desired direction. Hardware demonstrations of the Mini Cheetah jumping forward onto an 18 cm platform and laterally onto a 17 cm platform are shown in Fig. 4-1.

Acrobatics

The motion planner can also plan acrobatic motions involving large rotations of the robot's body about any of its axes. Currently, the back and front flips have only been demonstrated in simulation (Fig. 4-2), but the barrel roll and 180° spin have been achieved on hardware (Fig. 4-3). Similar to the jumps, kinematic reference motions are given that guide the robot's rotation about a given axis and terminal constraints enforce that the robot's orientation is suitable for landing at the time the feet are expected to come into contact with the ground.

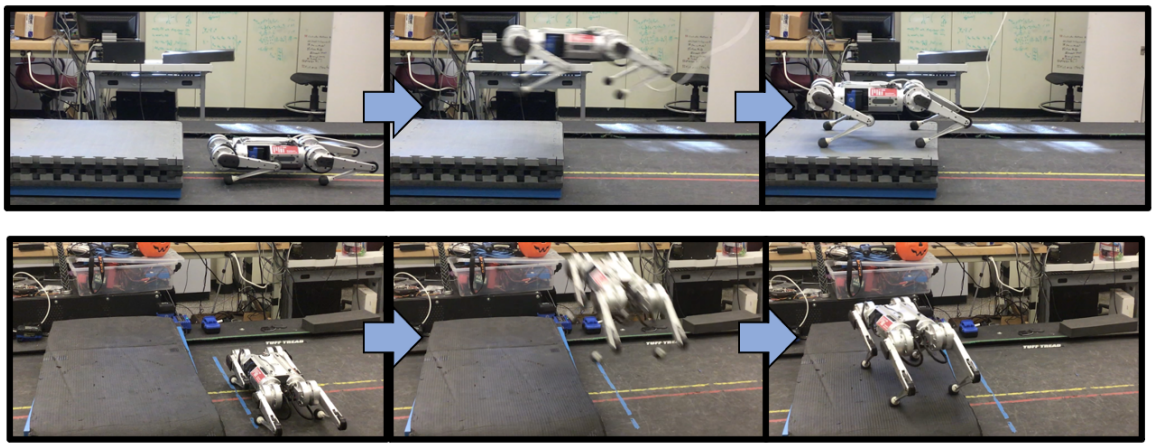

Figure 4-1: Hardware demonstration of the Mini Cheetah jumping forward onto a 20 cm platform and laterally onto a 18 cm platform.

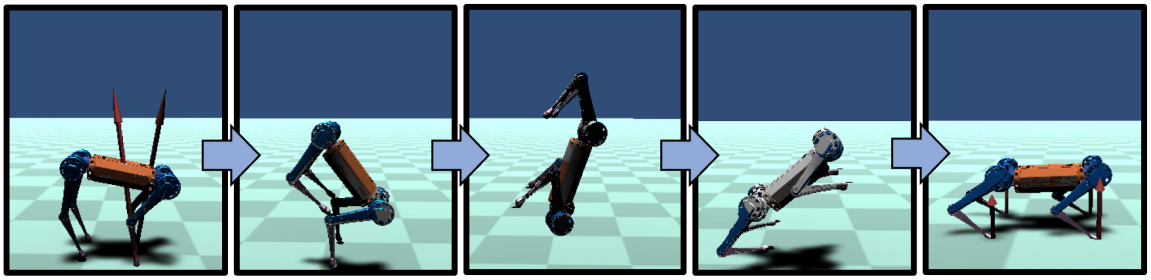

Figure 4-2: Simulation of Mini Cheetah performing a front flip.

Of all the aerial motions performed by the Mini Cheetah, the barrel roll involves the most significant displacement of the hip ab/adduction joints. In other words, the barrel roll produces the worst case scenario for the AAKD* actuator constraints. The torque and motor voltage profiles in Fig. 4-4 show that even in this worst case scenario, actuator limits are virtually always respected when the motion is planned with the KDAA* planner. Compare these torque and voltage profiles to those commanded

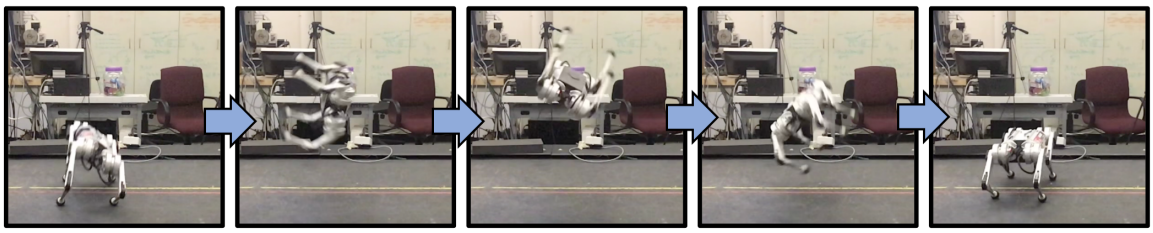

Figure 4-3: Hardware demonstration of Mini Cheetah performing an aerial barrel roll.

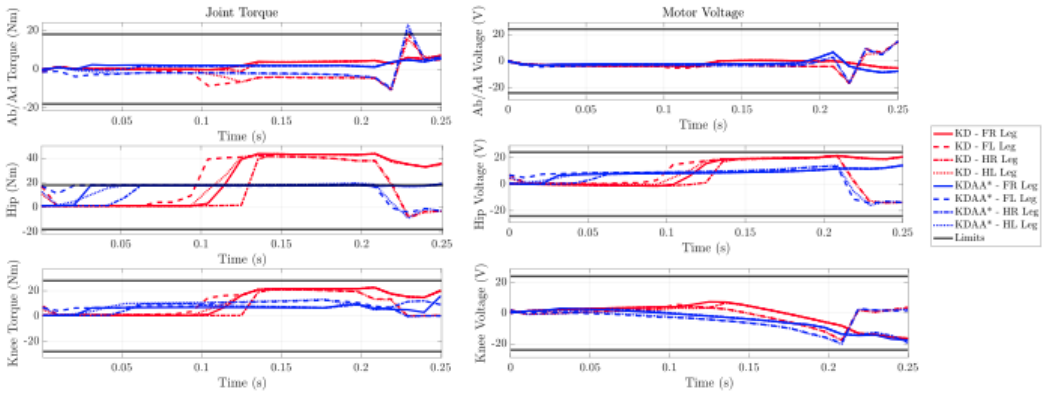

Figure 4-4: Joint torque and motor voltage profiles for the previously shown barrel roll motion.

using the standard KD planner that is blind to actuator limits. These profiles, also shown in Fig 4-4, call for sustained and severe violation of the limits of the hip motor, rendering the planned motion infeasible for implementation on the hardware. The KDAA* planner, on the other hand, commands only a mild 20% violation of the hip ab/adduction joint torque limit at a single timestep. The hardware results prove that such minimal violations commanded by KDAA* are small enough that the WBC can recover from the error that is introduced.

The front flip trajectory especially demonstrates the impressive tracking ability of the WBC, since this flip involves rotation about the robot's intermediate axis of rotation [3]. As a result of the rotation about this unstable axis, small errors in angular velocity at takeoff are greatly amplified. Despite this, the WBC is able to closely track the flipping motion (Fig. 4-5) and land the jump safely.

Running Jumps

The framework enables the Mini Cheetah to perform not only jumps starting from a static pose, but also jumps that begin in the middle of a running gait. Specifically, the MIT Mini Cheetah is able to perform 3D running jumps starting from either the trot and bounding gait. The reference motion and terminal conditions that produce these jumping motions are the same as the static jumping case described early. The contact schedule provided to the kino-dynamic optimization is simply a continuation of the

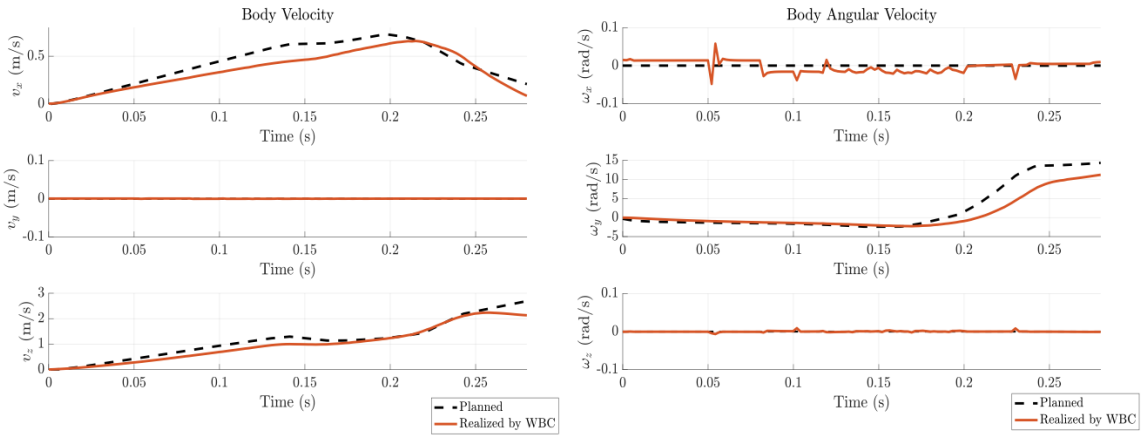

Figure 4-5: WBC tracking performance for the takeoff portion of the Mini Cheetah front flip.

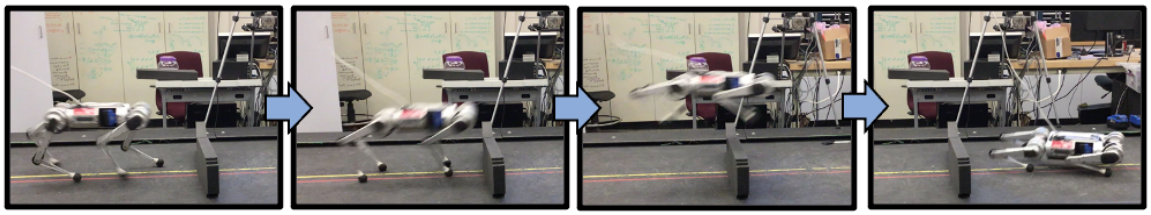

Figure 4-6: Jump from a trotting gait over a 17cm hurdle.

current gait schedule (e.g. trotting or bounding) for half of a gait cycle. Running jumps can clear hurdles of up to 17cm, as shown in Fig 4-6.

Both the trotting jump and the bounding jump experience significant periods of underactuation that make controlling the orientation of the robot considerably difficult. As illustrated by the WBC tracking performance shown in Fig. 4-7, the WBC is able to handle these periods of underactuation and maintain stability while still generating the force necessary to propel the robot over hurdles and onto elevated surfaces.

4.2 MIT Humanoid

The framework is general to robots with any number of legs, so in addition to the previously described aerial motions of the Mini Cheetah, the same framework has also produced 3D jumping, 180° spinning, and back flipping on the MIT Humanoid.

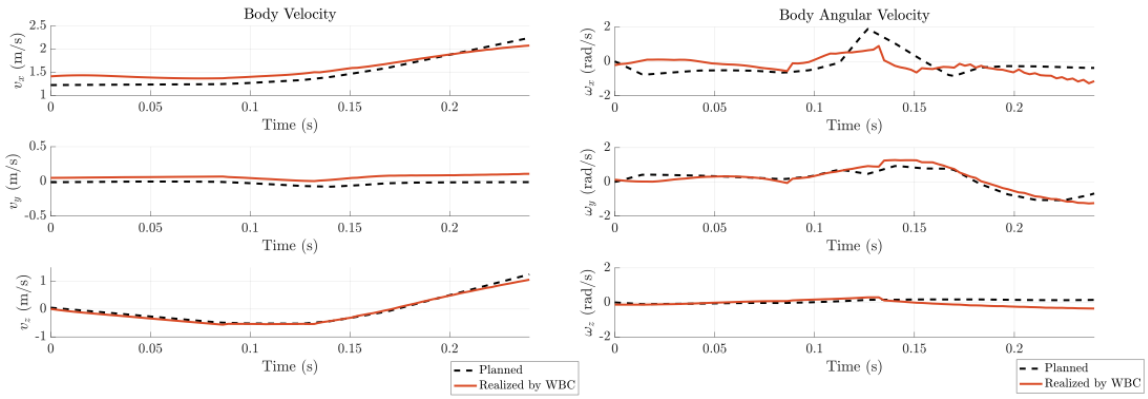

Figure 4-7: WBC tracking performance for the takeoff portion of the Mini Cheetah jump from a trotting gait.

Performing such motions on a humanoid robot is considerably more challenging than performing analogous motions on a quadruped robot because humanoids are inherently less stable. The ability of the WBC to track and safely land these inherently less stable motions of the humanoid demonstrates its robustness and validates the task prioritization discussed in Chapter 3.2.2. Similar to the Mini Cheetah results, the three kino-dynamic planning approaches with various forms of actuator constraints are again compared. The solve times required by each of these optimizations for the planned motions of the Humanoid are compared in Table 4.2. Again, solve times are considerably low, never greater than 10s, and the actuator constraints result in an increase in solve times of only 25% on average when compared to the standard KD case.

Table 4.2: Solve Times for Various Kino-Dynamic Approaches to Humanoid Motion Planning

	180° Spin	Forward Jump	Lateral Jump	Back Flip
KD	16.44s	53.34s	12.55s	56.00s
KD+WS	4.61s	4.72s	4.37s	3.62s
AAKD	62.15s	64.69s	29.08s	116.91s
AAKD+WS	4.95s	7.28s	6.57s	16.00s
AAKD*	38.61s	16.37s	56.77s	15.31s
AAKD* +WS	4.66s	5.70s	5.85s	9.98s

Jumping, spinning, and flipping motions, shown in and Fig. 4-8 to 4-10, are produced on the Humanoid via the same process used to produce analogous motions

on the Mini Cheetah. Terminal constraints on the robot's state at takeoff ensure that the robot's intended motion is accomplished, while simple kinematic reference trajectories guide the solution towards the desired behavior.

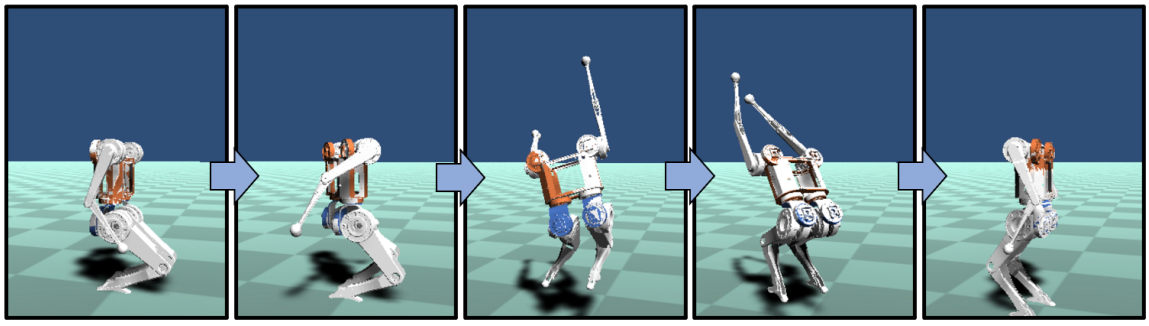

Figure 4-8: Humanoid performing a 180° spinning jump in simulation.

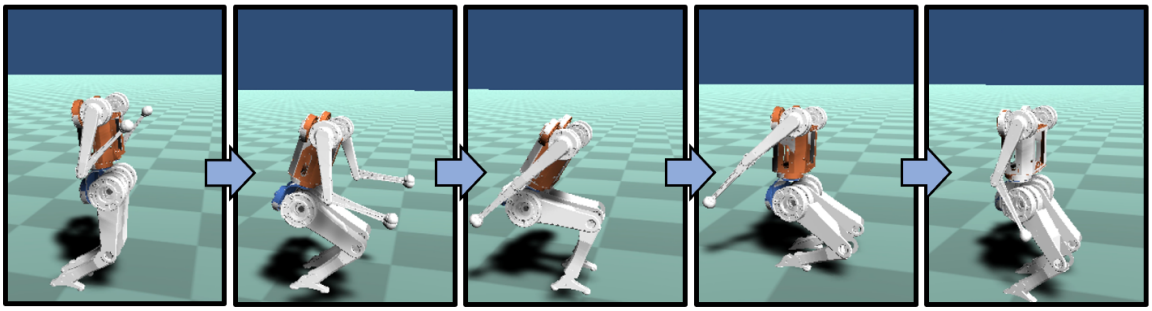

Figure 4-9: Humanoid performing a forward jump of 0.52m in simulation.

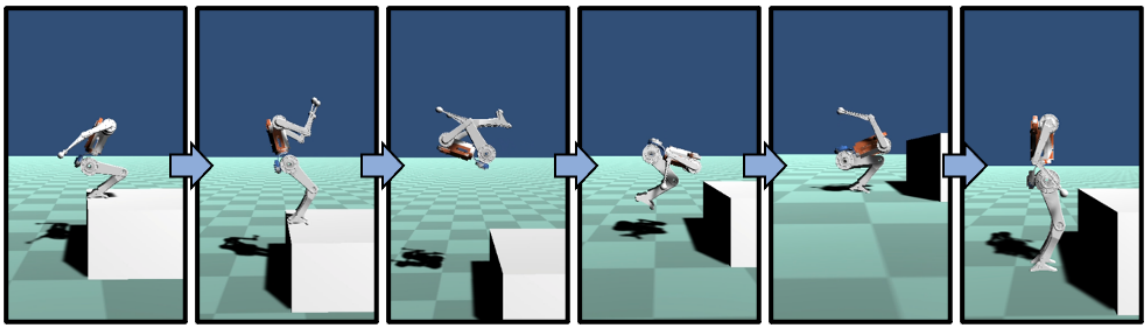

Figure 4-10: Humanoid performing a back flip off of a of 0.4m platform in simulation.

Again, the approximate actuator constraints enforced by AAKD* ensure that planned trajectories virtually never violate actuator constraints. The worst-case motion on the Humanoid for actuator constraint violation was the 180° spin. The joint

torque and motor voltage profiles for this motion, shown in Fig. 4-11, verify that even in the worst case there is essentially no constraint violation. The only violation that occurs happens for the ab/adduction joint of the right hip, but this violation occurs only briefly at the end of the motion. Other than that, all actuator limits are respected at all times. Compare this to the standard, actuation-blind KD planning case for the same desired motion, also shown in Fig. 4-11. The severe torque limit violations of the knee motors, which are also seen across most of the other aerial motions, make this trajectory virtually impossible for the WBC to track because it cannot create the large vertical impulses commanded near takeoff. The AAKD* approach gets around this issue by commanding smaller impulses, but sustained over longer periods of time.

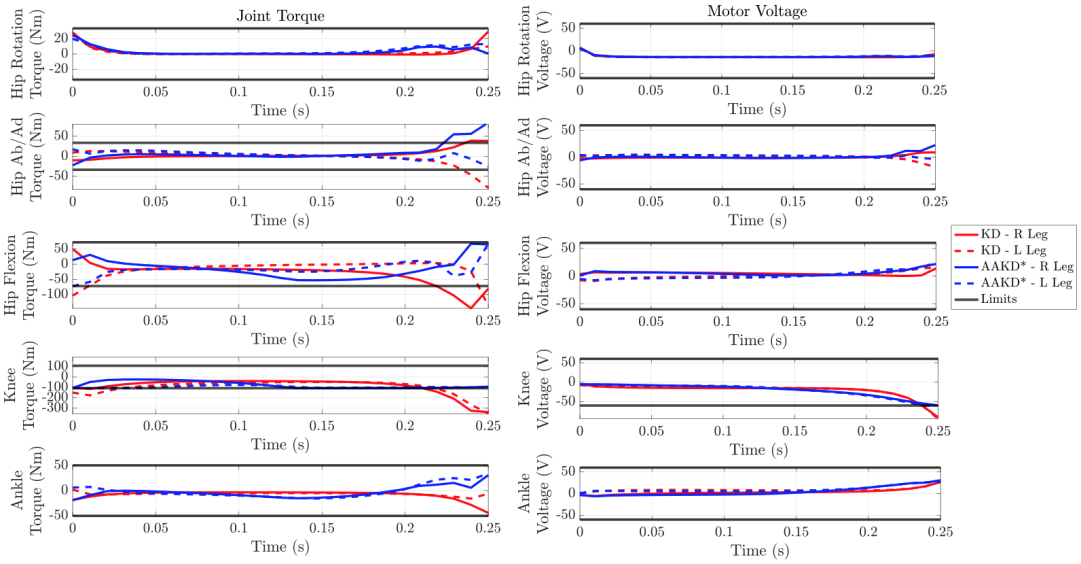

Figure 4-11: Joint torque and motor voltage profiles for the spinning jump motion of the Humanoid.

Notice that in Fig. 4-8 to 4-10, the Humanoid displays significant arm movements during takeoff, flight, and landing. None of these arm motions are explicitly commanded by the WBC. The WBC tracks positions of the feet in flight so that they can be in position for landing, but all arm motions emerge from the centroidal angular momentum tracking task of the WBC. This phenomena is not observed in the Mini Cheetah motions because the Mini Cheetah does not have any arms employ for the

angular momentum tracking task.

These emergent arm motions are essential in stabilizing the robot, especially during landing. In fact, none of the dynamic aerial motions of the Humanoid presented in this work could be achieved using a task prioritization for the WBC other than the one provided in Chapter 3.2.2. To validate this claim, a comparison of the tracking performances of WBCs with alternative task prioritizations was done. Specifically, three WBCs are compared, with their task hierarchies shown in Table 4.3. WBC-A uses the optimal prioritization laid out in Chapter 3.2.2, while WBC-B and WBC-C use sub optimal prioritizations that come close to achieving the jumps, but fail for various reasons.

Table 4.3: WBC Task Prioritizations Used in Performance Comparison

	WBC - Option A	WBC - Option B	WBC - Option C
Task 1	Floating-Base Posture	Centroidal Momentum	Floating-Base Posture
Task 2	Centroidal Angular Momentum		Arm Joint Position

The results in Fig. 4-12 compare the tracking performances of these WBC for the forward jump, and the results in Fig. 4-13 compare the tracking performances for the 180° spinning jump. In each of the cases, the sub-optimal task prioritizations fail for the same reason. In the case of WBC-B, which uses a centroidal momentum task rather than a floating-base task, the WBC is able to track the takeoff portion of the jumps fairly well. However, when the robot lands back on the ground with non-zero roll and pitch, the inability of the centroidal angular momentum feedback law to correct for the body's orientation error prevents the robot from stabilizing its posture and sticking the landing. In the case of WBC-C, which uses a joint position task after the floating base task, the WBC displays two major problems. First, the robot does a poor job tracking angular momentum, which results in under rotation for jumps like the spinning jump and the back flip. Furthermore, without being able to swing the arms upon landing to damp out angular momentum, this WBC is likewise unable to stick any of the landings.

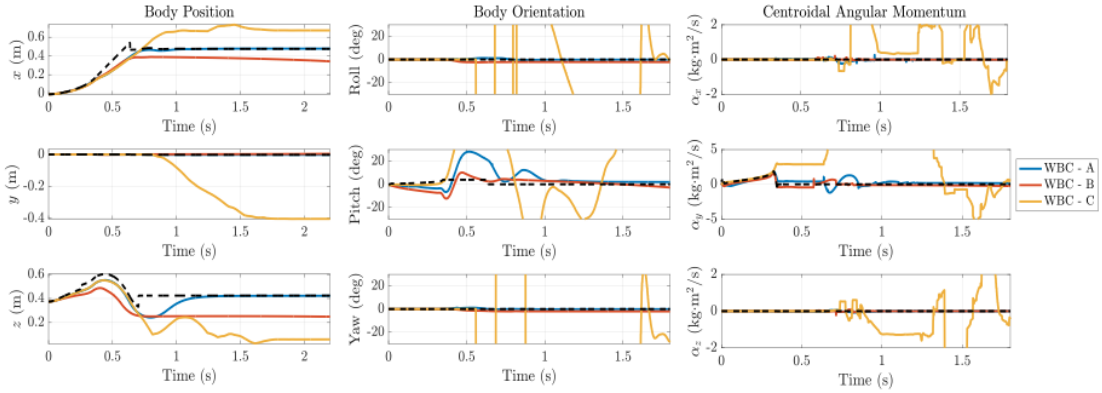

Figure 4-12: Tracking performance of different WBC task prioritizations for a forward jump of the Humanoid.

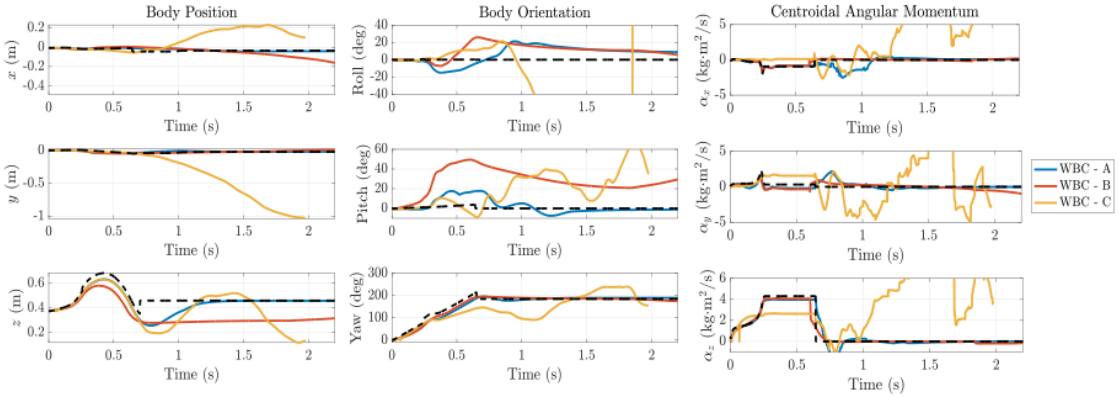

Figure 4-13: Tracking performance of different WBC task prioritizations for a 180° spinning jump of the Humanoid.

Chapter 5

Conclusion

The work set forth in this thesis represents the first step in the development of a control framework aimed at unlocking the full dynamic potential of legged robots. Through an efficient combination of kino-dynamic motion planning and robust whole-body control, legged robots can perform dynamic aerial motions that allow them to swiftly navigate environments not otherwise traversable by standard walking or running gaits. The generality of the proposed framework is demonstrated by the framework's ability to produce a rich set acrobatic motions for both a humanoid robot as well as a quadruped robot. Furthermore, the implementation of the framework on the MIT Mini Cheetah hardware verifies that the framework respects the constraints of the system and reliably generates motion plans that are dynamically feasible for the robot.

The novel two-stage kino-dynamic planning optimization dramatically increases the efficiency of the motion planning step by leveraging the SRBM approximation of the robot's dynamics to automatically generate a dynamically-informed starting point for the optimization. Additionally, the inclusion of novel actuator constraints for the kino-dynamic optimization allow the framework to be used for dynamic motions rather than the slower, quasi-static motions they are conventionally used for. The proposed hierarchical WBIC provides a reliable tracking controller that, without any task-specific modifications, can both track a diverse range of planned dynamic motions as well as stabilize the robot as it lands these maneuvers.

5.1 Future Work

While the results presented in this thesis demonstrate the viability of this framework as a means to vastly improve the mobility of legged robots, considerable work remains before the ultimate goal of fully autonomous navigation of challenging environments can be achieved. Such a goal can be accomplished through not only improvements to the planning and WBC stages of the framework, but also through integration of the framework with higher-level planning algorithms that replace a human operator in deciding where and when the robot needs to perform a dynamic maneuver.

Kino-Dynamic Planning

Presently, the most limiting aspect of the framework is that the kino-dynamic planner takes on the order of seconds rather than milliseconds to generate a motion plan. At the current solve rate, the framework cannot be run in real-time for motions like running jumps that start from any non-static initial position. Numerous steps can be taken to reduce the solve time of the planning step to enable real-time control. First, the optimization, which is currently solved in `MATLAB`, can be formulated and solved in `C++`. While `MATLAB` is a better development tool that allows for simpler debugging and design iteration of the planner, solving the optimization in `C++` is expected to provide a 4 to 10 times improvement in solve time. Furthermore, the linear solver `MUMPS` [1] is currently being used by `IPOPT` for solving the optimization. A benchmark study of different linear solvers that can exploit the specific sparsity pattern of this kino-dynamic optimization will likely provide a significant boost in efficiency.

In terms of theoretical rather than purely software improvements, there are additional improvements that can be made. First, the mostly costly constraints in the optimization involve the joint kinematics and the computation of the centroidal momentum matrix. A clever approximation of the centroidal momentum matrix such as a learned relationship between joint configuration and momentum matrix could offer a means for reducing the complexity of the problem while minimally impacting its accuracy or generality. Another limiting aspect of the planner is its assumption of

the robot's contact sequence. Restricting the time at which limbs come in and out of contact with the ground limits the ability of the robot to exploit its full dynamic capabilities. While simultaneous optimization of contact sequence, floating-base motion, joint kinematics, and reaction forces is almost certainly intractable for anything close to real-time control, exploring potential contact sequences can perhaps be done at a higher level of planning. Consider the manner in which the SRBM optimization largely determines reaction forces and floating-base motion while the main kino-dynamic optimization simply refines these in a way that respect joint kinematics. In a similar fashion, an initial contact scheduler could select contacts based on the robot's environment, and then the subsequent SRBM and kino-dyanmic optimizations could slightly modify that schedule as needed to ensure kinematics and actuator constraints are met. This potential contact planning step would be in line with the existing structure of the framework, which is based on the idea of distributing complexity at various levels rather than solving the entire problem all at once.

Whole-Body Control

While the proposed WBC demonstrated the ability to track planned motions with considerable fidelity, the WBC is limited in its ability to recover from perturbations from the reference trajectory. This limitation is a result of the WBC considering only the instantaneous desired motion of the robot, rather than the motion of the robot over some time horizon into the future. Reformulating the problem to consider this stabilization of the trajectory over a time horizon, similar to the approach in [10], could improve the ability of the WBC to reason about underactuated configurations as well as recover from perturbations from the nominal trajectory.

Perception Integration

For all of the results presented in this work, the dynamic motions demonstrated by the robots were selected and triggered by a human operator. Ideally, an autonomous robot would be able to use onboard perception and planning algorithms to both detect

obstacles as well as determine the motion best suited for navigating over, through, or around that obstacle. While preliminary work on perception and planning with the MIT Mini Cheetah [35] has shown impressive results (Fig. 5-1), the planning has been limited to finding safe footholds or walkable paths that avoid all obstacles rather than trying to dynamic maneuver over or across them.

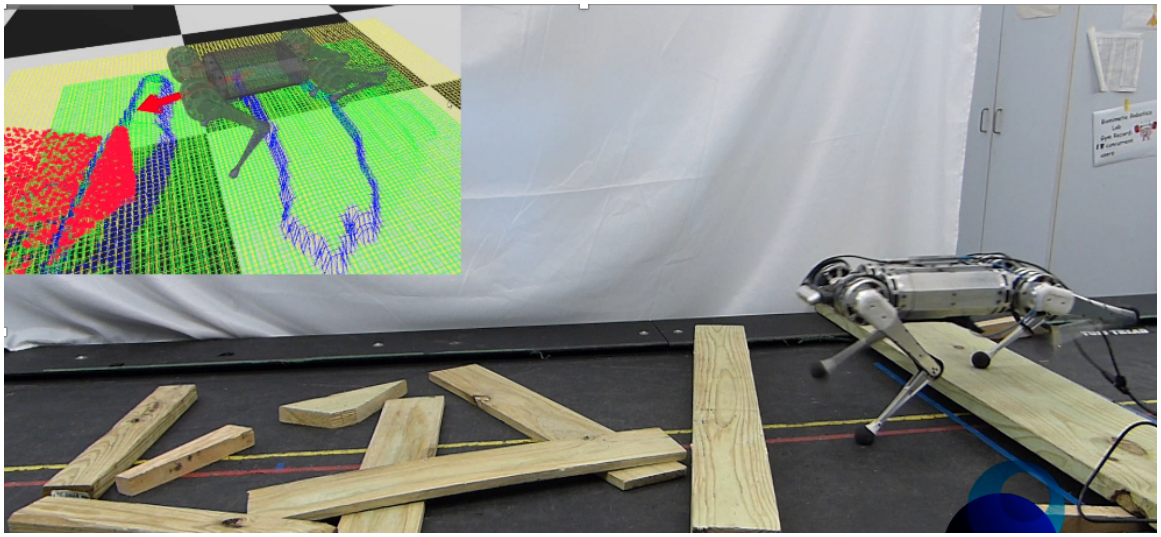

Figure 5-1: Mini Cheetah using onboard perception to select safe footholds for locomotion over rough terrain.

Such planners can, for the most part, consider only the robot's kinematics while ignoring its dynamics. This is not the case for a planner aimed at using dynamic motions to conquer challenging obstacles. Such a planner would either need to sample from a library of motions the robot has proven capable of or perform an approximate check for dynamic feasibility when planning something like a jump over an obstacle.

Bibliography

[1] Patrick R Amestoy, Iain S Duff, Jean-Yves L'Excellent, and Jacko Koster. Mumps: a general purpose distributed memory sparse solver. In *International Workshop on Applied Parallel Computing*, pages 121–130. Springer, 2000.

[2] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[3] Mark S Ashbaugh, Carmen C Chicone, and Richard H Cushman. The twisting tennis racket. *Journal of Dynamics and differential Equations*, 3(1):67–85, 1991.

[4] Sylvain Bertrand, Jerry Pratt, et al. Momentum-based control framework: Application to the humanoid robots atlas and valkyrie. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Workshop Slides*, 2014.

[5] Gerardo Bledt and Sangbae Kim. Extracting legged locomotion heuristics with regularized predictive control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 406–412. IEEE, 2020.

[6] Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

[7] Gerardo Bledt, Patrick M Wensing, and Sangbae Kim. Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the mit cheetah. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4102–4109. IEEE, 2017.

[8] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[9] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard. A versatile and efficient pattern generator for generalized legged locomotion. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3555–3561. IEEE, 2016.

[10] Matthew Chignoli and Patrick M Wensing. Variational-based optimal control of underactuated balancing for dynamic quadrupeds. *IEEE Access*, 8:49785–49797, 2020.

[11] Xingye Da and Jessy Grizzle. Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots. *The International Journal of Robotics Research*, 38(9):1063–1097, 2019.

[12] Hongkai Dai and Russ Tedrake. Planning robust walking motion on uneven terrain via convex optimization. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 579–586. IEEE, 2016.

[13] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302. IEEE, 2014.

[14] Martin De Lasa, Igor Mordatch, and Aaron Hertzmann. Feature-based locomotion controllers. *ACM Transactions on Graphics (TOG)*, 29(4):1–10, 2010.

[15] Christopher M Dellin and Siddhartha S Srinivasa. A framework for extreme locomotion planning. In *2012 IEEE International Conference on Robotics and Automation*, pages 989–996. IEEE, 2012.

[16] Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[17] Moritz Diehl, Hans Georg Bock, Holger Diedam, and P-B Wieber. Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*, pages 65–93. Springer, 2006.

[18] Yanran Ding, Abhishek Pandala, and Hae-Won Park. Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8484–8490. IEEE, 2019.

[19] Shamel Fahmi, Carlos Mastalli, Michele Focchi, and Claudio Semini. Passive whole-body control for quadruped robots: Experimental validation over challenging terrain. *IEEE Robotics and Automation Letters*, 4(3):2553–2560, 2019.

[20] Farbod Farshidian, Michael Neunert, Alexander W Winkler, Gonzalo Rey, and Jonas Buchli. An efficient optimal planning and control framework for quadrupedal locomotion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 93–100. IEEE, 2017.

[21] Roy Featherstone. The calculation of robot dynamics using articulated-body inertias. *The international journal of robotics research*, 2(1):13–30, 1983.

[22] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[23] Roy Featherstone and David Orin. Robot dynamics: equations and algorithms. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 826–834. IEEE, 2000.

[24] Robert J Full and Daniel E Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of experimental biology*, 202(23):3325–3332, 1999.

[25] Gabriel Garcia, Robert Griffin, and Jerry Pratt. Convex optimization of the full centroidal dynamics for planning in multi-contact scenarios. *available online at ResearchGate Preprint*, 2019.

[26] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[27] Hugh Herr and Marko Popovic. Angular momentum in human walking. *Journal of experimental biology*, 211(4):467–481, 2008.

[28] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491, 2016.

[29] Alexander Herzog, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. Trajectory generation for multi-contact momentum control. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 874–880. IEEE, 2015.

[30] Alexander Herzog, Stefan Schaal, and Ludovic Righetti. Structured contact force optimization for kino-dynamic motion generation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2703–2710. IEEE, 2016.

[31] Albert S Huang, Edwin Olson, and David C Moore. Lcm: Lightweight communications and marshalling. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4057–4062. IEEE, 2010.

[32] Marco Hutter, Hannes Sommer, Christian Gehring, Mark Hoepflinger, Michael Bloesch, and Roland Siegwart. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research*, 33(8):1047–1062, 2014.

[33] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[34] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[35] D Kim, D Carballo, J Di Carlo, B Katz, G Bledt, B Lim, and S Kim. Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2464–2470. IEEE, 2020.

[36] Donghyun Kim, Jaemin Lee, J Ahn, Orion Campbell, Hochul Hwang, and Luis Sentis. Computationally-robust and efficient prioritized whole-body controller with contact constraints. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[37] Donghyun Kim, Ye Zhao, Gray Thomas, Benito R Fernandez, and Luis Sentis. Stabilizing series-elastic point-foot bipeds using whole-body operational space control. *IEEE Transactions on Robotics*, 32(6):1362–1379, 2016.

[38] Jonas Koenemann, Andrea Del Prete, Yuval Tassa, Emanuel Todorov, Olivier Stasse, Maren Bennewitz, and Nicolas Mansard. Whole-body model-predictive control applied to the hrp-2 humanoid. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3346–3351. IEEE, 2015.

[39] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40(3):429–455, 2016.

[40] Scott Kuindersma, Frank Permenter, and Russ Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2589–2594. IEEE, 2014.

[41] Taesoo Kwon, Yoonsang Lee, and Michiel Van De Panne. Fast and flexible multilegged locomotion using learned centroidal dynamics. *ACM Transactions on Graphics (TOG)*, 39(4):46–1, 2020.

[42] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

[43] He Li and Patrick M Wensing. Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters*, 5(4):5448–5455, 2020.

[44] Guan-Horng Liu, Hou-Yi Lin, Huai-Yu Lin, Shao-Tuan Chen, and Pei-Chun Lin. A bio-inspired hopping kangaroo robot with an active tail. *Journal of Bionic Engineering*, 11(4):541–555, 2014.

[45] Katja Mombaur. Using optimization to create self-stable human-like running. *Robotica*, 27(3):321–330, 2009.

[46] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012.

[47] Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018.

[48] Quan Nguyen, Matthew J Powell, Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Optimized jumping on the mit cheetah 3 robot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7448–7454. IEEE, 2019.

[49] David E Orin and Ambarish Goswami. Centroidal momentum matrix of a humanoid robot: Structure and properties. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 653–659. IEEE, 2008.

[50] David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous robots*, 35(2-3):161–176, 2013.

[51] David E Orin, RB McGhee, M Vukobratović, and G Hartoch. Kinematic and kinetic analysis of open-chain linkages utilizing newton-euler methods. *Mathematical Biosciences*, 43(1-2):107–130, 1979.

[52] Hae-Won Park, Sangin Park, and Sangbae Kim. Variable-speed quadrupedal bounding using impulse planning: Untethered high-speed 3d running of mit cheetah 2. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5163–5170. IEEE, 2015.

[53] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. In *2015 Robotics: Science and Systems Conference, RSS 2015*. MIT Press Journals, 2015.

[54] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[55] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[56] Brahayam Ponton, Alexander Herzog, Andrea Del Prete, Stefan Schaal, and Ludovic Righetti. On time optimization of centroidal momentum dynamics. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

[57] Brahayam Ponton, Alexander Herzog, Stefan Schaal, and Ludovic Righetti. A convex model of humanoid momentum dynamics for multi-contact motion generation. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 842–849. IEEE, 2016.

[58] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[59] Michael Posa and Russ Tedrake. Direct trajectory optimization of rigid body dynamical systems through contact. In *Algorithmic foundations of robotics X*, pages 527–542. Springer, 2013.

[60] Alessandro Saccon, Silvio Traversaro, Francesco Nori, and Henk Nijmeijer. On centroidal dynamics and integrability of average angular velocity. *IEEE Robotics and Automation Letters*, 2(2):943–950, 2017.

[61] Alla Safonova, Jessica K Hodgins, and Nancy S Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (ToG)*, 23(3):514–521, 2004.

[62] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Citeseer, 2013.

[63] Luis Sentis and Oussama Khatib. A whole-body control framework for humanoids operating in human environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2641–2648. IEEE, 2006.

[64] Sangok Seok, Albert Wang, Meng Yee Chuah, David Otten, Jeffrey Lang, and Sangbae Kim. Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot. In *2013 IEEE International Conference on Robotics and Automation*, pages 3307–3312. IEEE, 2013.

[65] Benjamin J Stephens and Christopher G Atkeson. Dynamic balance force control for compliant humanoid robots. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pages 1248–1255. IEEE, 2010.

[66] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[67] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[68] Vassilios Tsounis, Mitja Alge, Joonho Lee, Farbod Farshidian, and Marco Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020.

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

[70] Patrick M Wensing and David E Orin. Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *2013 IEEE International Conference on Robotics and Automation*, pages 3103–3109. IEEE, 2013.

[71] Patrick M Wensing and David E Orin. Development of high-span running long jumps for humanoids. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 222–227. IEEE, 2014.

[72] Patrick M Wensing and David E Orin. Improved computation of the humanoid centroidal dynamics and application for whole-body control. *International Journal of Humanoid Robotics*, 13(01):1550039, 2016.

[73] Patrick M Wensing, Albert Wang, Sangok Seok, David Otten, Jeffrey Lang, and Sangbae Kim. Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. *Ieee transactions on robotics*, 33(3):509–522, 2017.

[74] Xiaobin Xiong and Aaron D Ames. Bipedal hopping: Reduced-order model embedding via optimization-based control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3821–3828. IEEE, 2018.

[75] Jun Zhang, Guangming Song, Yuya Li, Guifang Qiao, Aiguo Song, and Aimin Wang. A bio-inspired jumping robot: Modeling, simulation, design, and experimental results. *Mechatronics*, 23(8):1123–1140, 2013.

[76] Victor Zordan, David Brown, Adriano Macchietto, and KangKang Yin. Control of rotational dynamics for ground and aerial behavior. *IEEE transactions on visualization and computer graphics*, 20(10):1356–1366, 2014.