Regularized Predictive Control Framework for Robust Dynamic Legged Locomotion

by

Gerardo Bledt

B.S., Mechanical Engineering Virginia Tech (2015)
S.M., Mechanical Engineering MIT (2018)
S.M., Electrical Engineering & Computer Science MIT (2018)

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2020

Author .....
Department of Mechanical Engineering
January 15, 2020

Certified by .....
Sangbae Kim
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
Nicolas Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

Regularized Predictive Control Framework for Robust Dynamic Legged Locomotion

by

Gerardo Bledt

Submitted to the Department of Mechanical Engineering
on January 15, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Legged robots have the potential to be highly dynamic machines capable of outperforming humans and animals in executing locomotion tasks within dangerous and unstructured environments. Unfortunately, current control methods still lack the ability to move with the agility and robustness needed to traverse arbitrary terrains with the same grace and reliability as animals. This dissertation presents the successful implementation of a novel nonlinear optimization-based Regularized Predictive Control (RPC) framework that optimizes robot states, footstep locations, and ground reaction forces over a future prediction horizon. RPC exploits expertly designed and data-driven extracted heuristics by directly embedding them in the optimization through regularization in the cost function. Well-designed regularization should bias results towards a "good enough" heuristic solution by shaping the cost space favorably, while allowing the optimization to find a better result if it exists. However, designing meaningful regularized cost functions and adequate heuristics is challenging and not straightforward.

A novel framework is presented for automatically extracting and designing new principled legged locomotion heuristics by fitting simple intuitive models to simulated and experimental data using RPC. Statistically correlated relationships between desired commands, robot states, and optimal control inputs are found by allowing the optimization to more exhaustively search the cost space during offline explorations when not subjected to real-time computation constraints. This method extracts simple, but powerful heuristics that can approximate complex dynamics and account for errors stemming from model simplifications or parameter uncertainty without the loss of physical intuition.

Nonlinear optimization-based controllers have shown improved capabilities in simulation, but fall short when implemented on hardware systems that must adhere to real-time computation constraints and physical limits. Various methods and algorithms critical to the success of the robot were developed to overcome these challenges. The controller is verified experimentally using the MIT Cheetah 3 and Mini Cheetah robot platforms. Results demonstrate the ability of the robot to track dynamic ve-

locity and turn rate commands with a variety of parametrized gaits, remain upright through large impulsive and sustained disturbances, and traverse highly irregular terrains. All of these behaviors are achieved with no modifications to the controller structure and with one set of gains signifying the generalized robustness of RPC. This work represents a step towards more robust dynamic locomotion capabilities for legged robots.

Thesis Supervisor: Sangbae Kim
Title: Associate Professor of Mechanical Engineering

Acknowledgments

Most importantly, I want to share my appreciation for the love, support, and encouragement that parents, Carlos and Marcela, have had, not only throughout my graduate school studies, but throughout my life. Without them, I would not have been able to pursue my degree. My brother, Carlos, was influential in my decision to continue engineering research at a higher level and showed me it was possible to do. This dissertation is dedicated to my family.

My advisor Dr. Sangbae Kim has allowed me the freedom to explore research that I am passionate about and provided the opportunity to work in an incredible lab with some of the best robots in the world. His intuition for robotics makes the lab successful in integrating a largely interdisciplinary field and provided me with direction in my own work. My committee members Dr. Alberto Rodriguez and Dr. Steven Hall for their input on my dissertation research. Their advice on optimization techniques was hugely beneficial to implementation of the controller on the robot and tips on related research defined some of the methods used.

Next, I'd like to thank all of the people involved in the Cheetah robot project throughout my time in the Lab. A huge thanks to Pat Wensing who guided me through the research process and was an endless source of knowledge about robotics and dynamics. None of the hardware results presented would have been feasible without Ben Katz's amazing Mini Cheetah design and his work fixing and debugging the robot platforms. The great work by Jared Di Carlo and Donghyun Kim for the robot software was an incredible help and allowed me to focus on and implement my research. The whole process would have taken much longer without their contributions. Matt Powell, Quan Nguyen, and Chiheb Boussema's contributions to the software and assistance with robot experiments sped up progress and facilitated getting the results.

I also thank everyone who has helped with the other aspects of graduate school besides research. The MIT MechE staff, in particular Leslie Regan has made the process of getting my Masters and PhD a lot more manageable. Albert Wang, João Ramos, Michael Chuah, Daniel Carballo, and the other members of the MIT Biomimetic

Robotics Lab that made the time at the lab enjoyable are greatly appreciated. Finally, a thanks to all of my friends from Madison, Virginia Tech, and Boston that have made my time outside of the lab enjoyable and filled with good memories. Often times, having a small distraction from research is exactly what is needed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotics is a field of research with immense potential to transform the world we live in by creating systems with capabilities surpassing humans in performing dangerous and tedious tasks. For years, robots have been designed to carry out repetitive, monotonous tasks in controlled environments such as manufacturing and packaging. However, when entering the real world full of uncertainty and lack of structure, they fall short of realizing their potential. Robotic systems are not yet at the level required to reliably work alongside first responders in ambiguous disaster scenarios or to autonomously navigate a populated urban setting. To operate in uncontrolled environments, robots need to be able to comfortably manage physical interactions with objects in their environment in new ways that they may not have been specifically programmed to handle. Animals and humans move through unfamiliar, cluttered environments, while manipulating complex objects and surroundings with ease, but it is precisely this general ability that robots currently lack.

It is this challenge that is addressed in this dissertation. A controller is designed for general blind locomotion that is capable of robust dynamic maneuvers over unstructured terrain. Abilities are demonstrated using hardware systems and require no specific design or changes to the controller structure to accomplish each of them. A control method capable of handling large disturbances and traversing any terrain is necessary for widespread adoption of robots.

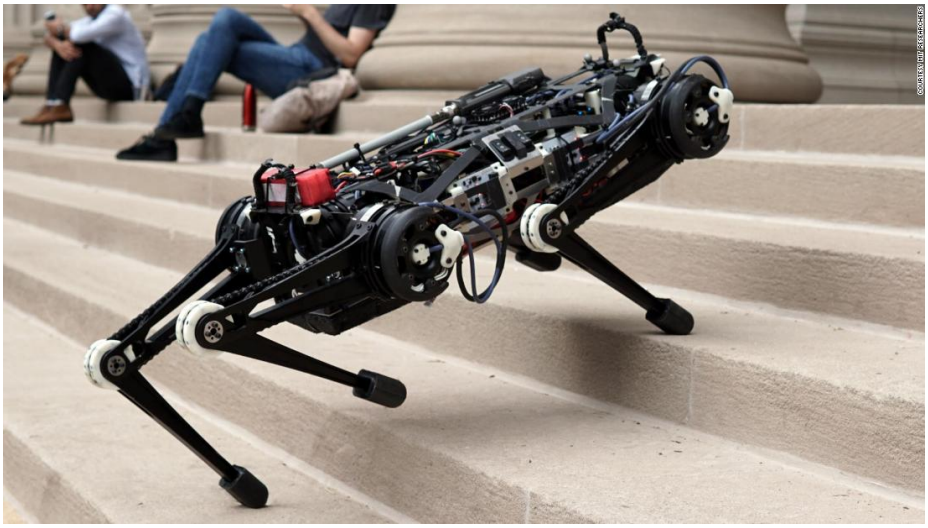


Figure 1-1: **Stair Climbing with Legged Robot.** As opposed to wheeled or tracked vehicles, legged systems have the capability to navigate many man made obstacles such as stairs.

1.1 Motivation for Legged Robot Research

Legged locomotion in particular is an area of research that has potential to make robots that take on dangerous, tedious, or unwanted tasks that are currently done by humans. A major motivation for building legged robots is disaster response scenarios where it is too risky to send a human responder, the terrain is often impossible to traverse with wheels or tracks, and quadrotors or gliders cannot carry loads or manipulate the environment in a way that a ground vehicle can. Often, the first responders being called to assist these situations are putting their lives in as mush risk as the victims of the tragedy itself. Being able to use robotic systems for dangerous situations, either in place or alongside trained first responders, would reduce the risk to human lives and the number of fatalities suffered.

Legs allow for access to areas that are inaccessible for wheeled or tracked vehicles such as climbing stairs or traversing extremely rough, unstructured terrains. While there are non-legged robots that are able to climb stairs or jump over obstacles, they are often designed uniquely for the specific task with the tradeoff of losing other abilities. Legged designs are powerful in that they provide the versatility needed to accomplish various tasks with no need for specific design. This is particularly useful in

(a) High Speed Cheetah Turn (b) Baby Goat Climbing (c) Goat Pushing Dog

Figure 1-2: **Dynamic Legged Animals.** Legged animals possess natural abilities to dynamically move around challenging environments that currently outperform the most sophisticated robotic systems.

a world largely designed by humans with legs, meaning that often, urban environments feature discrete terrain changes such as stairs and curbs, as well as surface changes from concrete, wood, dirt, and grass. Figure 1-1 shows the MIT Cheetah 3 robot adeptly climbing over stairs built for human pedestrians. By directly interacting with their surroundings, they are able to carry heavy loads and manipulate objects to change their surroundings.

We can imagine that a robot designed for use in disaster response scenarios may need to quickly get to the location of the incident as response time is often crucial in these situations. It will likely need to traverse non-smooth, unstable terrain such as rubble or natural environments with unexpected conditions. Obstacles that it cannot simply walk over may require the robot to leap to clear a gap or fallen structure. These are all abilities that legged robots should be capable of doing, but have not yet been demonstrated reliably in real-world situations.

Many animals with legs are able to accomplish all of these tasks naturally without any specific training. Figure 1-2 shows pictures of animals naturally performing highly dynamic maneuvers reliably in their everyday habitats. Cheetahs are able to run over $100\frac{km}{h}$ while making sudden quick turns (1-2a). Mountain goats can climb over rocky, near vertical cliffs at ease without falling (1-2b). Dogs and goats run are able to play by pushing each other roughly without falling and quickly regaining their balance (1-2c). While these are only three specific examples, legged animals are capable of impressively maneuvering many more environments. Even newborn animals already outperform our most advanced robots. With two newly developed robots by the MIT

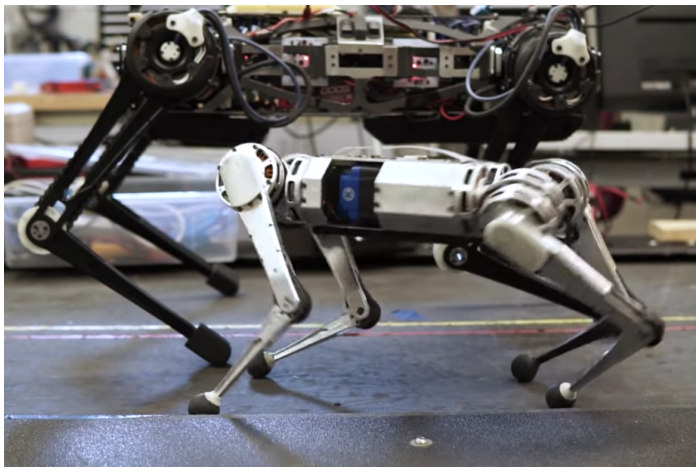


Figure 1-3: **Legged Robot Platforms.** The control framework developed in this dissertation was designed for the MIT Cheetah 3 and Mini Cheetah robots.

Biomimetic Robotics Lab, the MIT Cheetah 3 and the Mini Cheetah pictured in Figure 1-3, the controller developed in this dissertation brings the ability of legged robots closer to these dynamic capabilities.

1.2 Recent Advancements and Challenges

Robotics in general is still a young field with many open questions remaining, many of which have numerous different solutions. As a largely interdisciplinary field, robots need to have both good hardware design as well as robustly capable control systems. There is not one single way to control a robot and each has advantages and limitations. Despite the encouraging results in robotics, we do not see them operating at the level of animals or even with enough reliability to execute tasks in uncontrolled environments. Quadruped robots have seen a surge in popularity because of the increased accessibility to good hardware, appropriate actuation methods, and the considerable capabilities being demonstrated by new controllers. Hydraulic actuators have proven successful in Big Dog by Boston Dynamics [67] and IIT's HyQ quadruped [73]. The robots take advantage of hydraulic actuation system's ability to output large joint forces. ANYmal at ETH Zürich [35] has an extensive range of motion for use in real-world autonomous tasks with Series Elastic Actuators (SEAs). Properties of SEAs

are typically good impact mitigation properties and excellent force control capacity. The MIT Cheetah 2 made use of custom proprioceptive actuators with high impact mitigation, force control, and position control capabilities. This design enabled it to autonomously jump over obstacles [60] and bound at high speeds of 6m/s [61], but had a limited range of motion constraining it to sagittal plane locomotion. The MIT Cheetah 3 [6] and Mini Cheetah [41] robots are the latest platforms featuring proprioceptive actuators that address the issues present in their predecessor. With hardware designs improving, there has been a number of recent advancements in control strategies.

1.2.1 Simplified Model Heuristic Control

Approaches using simple template models for locomotion have been used to reduce the complexity of the problem to analyze or optimize over [28, 47]. One of the earliest, most influential, and frankly impressive demonstrations of early legged robot capabilities by the MIT Leg Lab demonstrated hopping and acrobatic bipeds that moved over different terrains without the need for large optimizations or perception of its environment [68]. With simple, but well informed heuristics, the robots were able to run, be pushed, and even accomplish flips while running. Another physics-based heuristic that has proven to be highly influential in legged systems is the capture point heuristic, based off of treating the robot like a linear inverted pendulum and finding the best point to place the bottom point in order to negate the momentum by the time the mass at the top reaches the apex [65].

The inverted pendulum dynamics have been extensively analyzed and provide a good simplified model for systems with legs. reduce the problem of a high dimensional legged system to a small number of well-understood states. The Spring-Loaded Inverted Pendulum (SLIP) model is widely used as a simplified template [9, 17]. It is used to embed known physics principles into control frameworks like Hybrid Zero Dynamics (HZD) [63] and to generate limit cycles [14]. Simple centroidal momentum dynamics representations enable graceful recoveries to disturbances in real-time simulation environments [85, 17].

More recently, these simple template model approaches have been applied to working higher-dimensional systems. Using a model of a quadruped robot that includes a lumped mass model with no legs and only the CoM dynamics, instantaneous ground reaction forces are calculated to walk along sloped terrain [26]. The HyQ robot demonstrated the ability to heuristically plan traversals over rough terrain with degraded perception systems and under external disturbance [27]. Other work shows emerging gaits through a simplified model of a quadruped [62]. These examples demonstrate that it is not absolutely necessary for control methods to have absolutely accurate models for locomotion under disturbances and rough terrain.

1.2.2 Whole-Body Control (WBC)

A commonly used control method called Whole-Body Control (WBC) features a dynamically consistent formulation in which the entire system's dynamics and their joint torques and external forces are described and optimized over. WBC frameworks can prioritize tasks that need to be executed in a hierarchical manner that carries out the most important ones, such as balancing, with a secondary consideration to others such as arm placements [74]. Simulation results show the Atlas robot executing complex dynamic maneuvers such as the salmon ladder, monkey bars, and jumps with centroidal dynamics and full kinematics [15]. An WBC implementation on hardware uses adaptive-sliding mode control and dynamics estimation to overcome the need for an accurate model of the robot [48]. An efficient prioritized WBC implementation successfully controls a passive-ankle biped in both simulation and experiment by using smooth contact transitions and formulating the problem as a Quadratic Program (QP) on the centroidal momentum dynamics [44]. A similar simplification method is used to make WBC tractable for control of a quadruped [49].

Though basic WBC optimizes joint torques over a complete models of robotic systems, it finds only instantaneous joint torques and forces. This means that it has no inherent awareness of future events without a higher-level planner that executes a prediction. Gaits with flight phases or long periods of underactuation, such as galloping will not work with pure WBC. Often the use of Zero Moment Point (ZMP)

planning methods can allow for less quasi-static movements by passing in a planned reference trajectory [55]. Trajectory optimizations and generators are used to plan out a set of footsteps and body trajectory on the HyQ robot and a WBC executes the movements to play back the plan in simulation and hardware [51]. The robot is able to successfully walk over ramps, stairs, and other uneven terrain. This class of controllers is proficient in calculating instantaneous commands, but depends on the provided trajectory to track.

1.2.3 Model Predictive Control (MPC)

Simple prediction models for short, finite time horizons have long been thought to play a role in animal locomotion strategies [43]. Model Predictive Control (MPC) techniques attempt to act in a similar manner by solving an optimization for some control sequence over a receding planning horizon to track a desired trajectory [83]. MPC has shown promise in a variety of legged systems such as bipeds [64, 1] and quadrupeds [57, 12, 16]. MPC simulations also show an ability to reject push disturbances by adjusting footstep locations for bipedal robots with simplified representations of the kinematics and dynamics [32, 75].

However, many of these implementations rely on solving strictly convex optimizations on linear systems. While powerful and able to solve in real-time, they are limited in that they require several separate optimizations to decouple nonlinearities and therefore are not informed about inherent interactions between important coupled dynamics. Results shown on the HRP-2 Humanoid robot platform demonstrate a whole-body MPC, but notes the limitations of this approach lie in the computation requirements, nonlinear local minima, and algorithm convergence [45].

1.2.4 Nonlinear Trajectory Optimization

Nonlinear optimization-based controllers are powerful as they take advantage of more complex and coupled dynamics, which ultimately results in a better solution. Recently, they have become more tractable to compute in real-time as general com-

puting power is increasing. Quadcopters in particular have shown impressive results using fast, online Nonlinear MPC (NMPC) [56, 40]. This class of controllers is particularly difficult in legged systems due to the hybrid nature resulting from discrete mode changes that completely change the cost space. A simple hopping robot system showed NMPC over rough terrain [72]. Simulation of nonlinear trajectory optimizations shows the ANYmal robot jumping over gaps and climbing on slanted walls, but has not year been realized on hardware [86]. A direct predecessor leading to the controller presented in this dissertation showed footstep and ground force optimization for various gaits on a model of the MIT Cheetah 2 in purely in simulation [8, 3]. A version of a whole-body NMPC was demonstrated on a quadruped in hardware [58] where the ANYmal robot is able to stably stand, move quasi-statically, and reject minor disturbances. Nonlinear optimization in particular has proven difficult to realize in practice due to timing constraints and undesirable local minima. The whole-body nonlinear MPC hardware implementation in [58] notes that computation requirements and real-world limitations still exist and will need to be overcome to fully take advantage of the NLP.

Designing cost functions is not a straightforward process. Particularly as robot models improve and become more complex with nonlinearities and prediction horizons, meaning more decision variables. Assuming the optimization converges, the result is only optimal for the cost function provided to the solver. There are many techniques to design cost functions [31], but there is no guarantee that the cost function is the best choice and may require tedious manual tuning and expert design to even achieve adequate results. In legged robots, it is especially difficult to tune and adapt cost function parameters because instantaneous mode shifts from contact changes cause discontinuous gradients. Early work determined it was best to find a working cost function manually and adapt withing a local region around the successful parameters [71]. Others have attempted to replicate naturally seen behavior by inferring cost function weights through inverse optimal control [69], but this requires having a good guess for the structure and variables in the cost. Neural Networks have also been employed to attempt to imitate human motions [53]. However, the design

of meaningful cost functions remains an open problem that is application-specific.

1.2.5 Machine Learning

In recent years, there has been interest in using neural networks to control legged systems with some functioning results. Machine learning has been able to solve many extremely difficult tasks in other areas of robotics [76], such as computer vision [38, 25], manipulation [46, 20], and multi-agent cooperation [30, 66]. As a result, it has now gained popularity as an approach to tackle issues in legged locomotion. In [37], ANYmal learns to stand up from an fallen position with a learned policy and to walk forward with no specified gait with another policy. Both policies were learned in simulation and were successfully ported over to the real robot. Impressively, the neural network learns to infer contact with its limbs and body from the joint trajectories purely in simulation, and uses its many contact points to realize the standing. Other work on ANYmal shows a framework for planning gaits and footstep locations using a trained network [79]. Work from DeepMind has shown the ability to learn to move in complex environments on various different dynamic systems [34, 77, 87]. However, as is often a drawback of neural networks, it is not always clear how to modify specific policy parameters in order to tune the behaviors without the need to retrain the entire network on a different set of explorations.

In the case of legged robots, the problem is low dimensional enough where many of the existing techniques discussed already provide good results while simultaneously allowing the control designer to maintain intuition over each part of the algorithm. For this reason, while powerful and promising, we elect to forgo using deep neural networks to carry out end-to-end learning for locomotion. However, in the future it may be worth using these methods for parameter adaptation or to solve isolated highly nonlinear modeling issues that may not be approximated easily or require near perfect accuracy such as SEA motor characterization shown in [37]. Either way, it is acknowledged that there is no singular method that will prove to be the only technique adept at robust dynamic legged locomotion and the work presented is intended to provide one method that has improved the current state of robot capabilities.

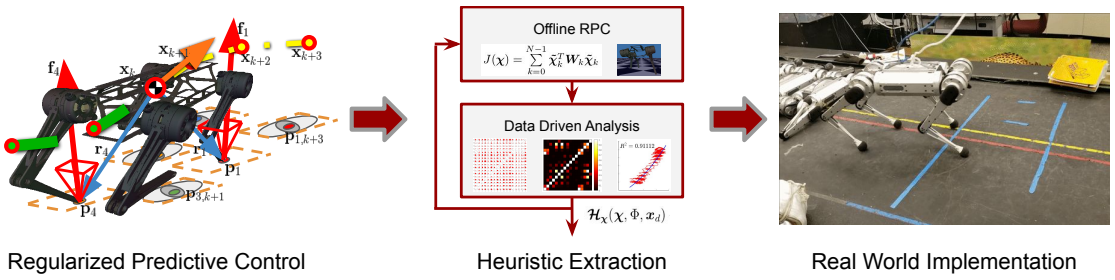


Figure 1-4: **Main Contributions.** The contributions described range from the theoretical Regularized Predictive Control framework, to the extraction of regularization heuristics from simulation data, to the implementation on the hardware.

1.3 Contributions

The work presented represents advancement in the field of legged robotics. The overall goal is to design a robust control framework capable of robust dynamic locomotion. Figure 1-4 shows the three main contributions are highlighted throughout this dissertation which are:

1. Regularized Predictive Control: A control framework that exploits simple physics-based and data-driven heuristics through nonlinear optimization to simultaneously pick future footstep locations and ground reaction forces for locomotion.

2. Heuristic Extraction: A novel framework for extracting legged locomotion heuristics through simulated and experimental data. Developing meaningful, general cost functions can be challenging. The automated method provides a powerful tool for discovering new regularization heuristics to embed in the cost function.

3. Real World Implementation: Various algorithms and techniques to overcome the challenges of online nonlinear optimization in real-time. The controller is implemented on the MIT Cheetah 3 and Mini Cheetah robot platforms.

To determine the success of the controller, a set of goals is selected describing the general capabilities that the robot should be able to accomplish reliably. The control framework described is meant as a mid level blind balance and locomotion controller meaning it should be capable of tracking commands and rejecting disturbances such

as pushes, slips, and various terrains. The general capability areas that will be tested are outlined:

1. Basic Capabilities:
    - Track velocity and turn rate commands
2. Locomotion Robustness:
    - Robust to unexpected disturbances
    - Traverse unstructured terrain
    - Flexible framework

Since legged locomotion has no clearly agreed upon metric of stability in the classical sense, we determine the robot to successfully accomplish the tasks if the robot has not fallen or diverged through a number of gait cycles for steady state behaviors and is repeatable over many trials with a high success rate for disturbance tests. The specific metrics of success are determined more directly on a case by case basis.

1.4 Dissertation Organization

The rest of this dissertation is organized is as follows. Chapter 2 details the quadruped systems, the simulation environment used for development, and the overlying control architecture that the robot operates on. Apart from the controller detailed in the dissertation, operating the robot requires many other parts such as higher-level trajectory planning, state estimation, and gait scheduling, all of which contribute to the successful locomotion of the robot.

Chapter 3 develops the Regularized Predictive Controller in detail. A cost function is that exploits simple physics and data-driven heuristics through regularization is designed. The impact of regularization heuristics on the cost function is described to show the importance of the novel control scheme. Constraints for physical feasibility are written to make the solution realizable on the robot. This section lays out the

bulk of the basic nonlinear optimization-based control framework that makes up the major contribution of this work.

The next two sections build on the theory behind the RPC framework. Chapter 4 discusses a novel framework that uses RPC to systematically extract new heuristics. It is challenging to design a meaningful cost function for predictive control algorithms in all cases, so this chapter presents a method to help design heuristics from experimental and simulation data. Next, Chapter 5 describes several implementation details that were developed to successfully get the robot hardware to run the controller. Due to the challenges of real-time constraints, it is not enough to simply run the optimization and expect that solutions will be returned as fast as needed while synchronizing the results of the optimization problem and the scheduled contact state. Chapter 6 demonstrates a series of robustness experiments that test the robot's ability to operate in the real world and handle unexpected disturbances. Various experiments deal with cases that were not considered when designing the controller, but demonstrate its general applicability.

Finally Chapter 7 discusses the importance of the results and the implications of the work developed. The controller itself is a new implementation that shows stable locomotion and rejection to a variety of unexpected disturbances. In addition to the results presented, this section describes the meaning of the techniques used. This represents a step towards using nonlinear optimization-based controllers that find footholds and forces in practice on hardware. The method outlined for heuristic extraction is a powerful tool that is generalized to both improve performance and increase knowledge of legged systems in cases that may be unintuitive or difficult to analyze with currently existing methods.

Appendices contain supplementary information that may be helpful for the reader. Appendix A concisely defines the nomenclature for many of the important variables used throughout the dissertation. Appendix B defines the RPC optimization problem concisely with the cost function and all of the constraints as used on the robot. Appendix C lists the analytically designed heuristics and the data-driven extracted heuristics used in the final implementation in separate tables. Appendix D notes sev-

eral publications that are directly related to the work done as part of this thesis, as well as publications of related work by the MIT Biomimetics Robotics Lab that represent parallel research that may serve to guide future directions and complimentary methods.

Chapter 2

Theory and System Modeling

This section largely represents a summary of the general design background and control architecture used and developed by the MIT Biomimetic Robotics Lab for legged robot control. The MIT Cheetah 3 and the Mini Cheetah robots were both custom designed in the lab and operate using the same software as they were built to be similar to each other at different scales. Brief descriptions of both robot platforms, their respective kinematics, physical parameters, and actuator design is provided. A comprehensive system architecture description for the quadruped control system used is described in [6, 41], with the more important relevant parts concisely reviewed in this chapter.

2.1 Robot Platforms

Throughout this dissertation, a control framework for legged robots is developed and implemented on the MIT Cheetah 3 and Mini Cheetah quadruped robot platforms. While the framework is designed generally for any number of legs and for any system, many design choices were taken specifically with these hardware platforms in mind. Both use similar design choices for the kinematics and actuation principles. For this reason, the frameworks are general to both robots and can be used interchangeably by simply switching the model.

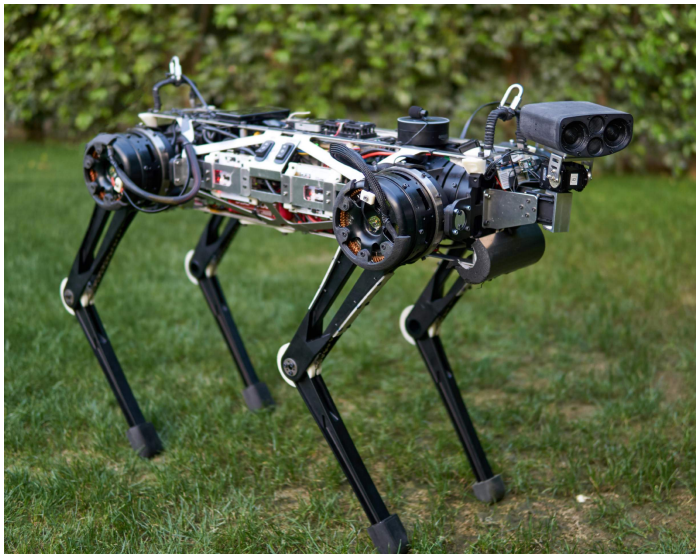


Figure 2-1: **MIT Cheetah 3.** The MIT Cheetah 3 quadruped robot platform is an electrically powered robust robot capable of untethered 3D locomotion in uncertain terrains.

2.1.1 Hardware Design: MIT Cheetah 3

The MIT Cheetah 3 is a quadruped robot designed to use simple control strategies using its high bandwidth proprioceptive actuators to manage contact interaction with its environment as pictured in Figure 2-1. Using the experience gained from the MIT Cheetah 2 [60, 61], its predecessor, the new platform employs many key improvements. The most important are a larger range of motion, higher force production abilities, and an added proprioceptive actuator controlling the abduction / adduction (Ab/Ad) degree of freedom which allows the robot to control ground reaction forces in 3D. Cheetah 2 was designed for mainly planar motion, while Cheetah 3 is able to move in arbitrary directions.

The robot uses high torque density electric motors with backdriveable single-stage planetary gear reductions with low inertia legs. The proprioceptive actuators allow it to control ground reaction forces without the use of any force sensors and possess a high impact mitigation factor (IMF) [84]. The three actuators are assembled compactly as shown in in Figure 2-2. The first Ab/Ad actuator moves the hip and knee actuator modules together through a linkage, then the hip actuator rotates the entire leg link and knee actuator combination. Finally, the knee actuator rotates the

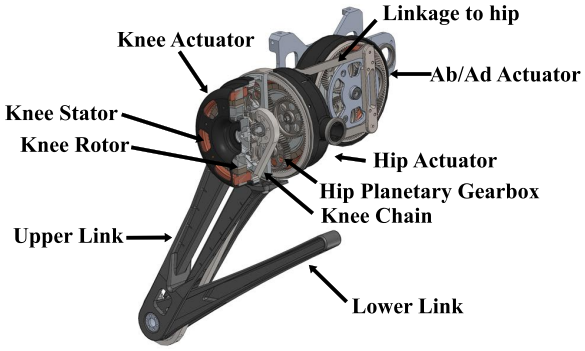


Figure 2-2: **Cheetah 3 Leg Design.** One leg showing the 3 actuators, with a cutaway view of the knee actuator.

Table 2.1: MIT Cheetah 3 Physical Robot Parameters

Parameter	Symbol	Value	Units
Mass	m	45	kg
Body Inertia	I_{xx}	0.35	$\text{kg} \cdot \text{m}^2$
	I_{yy}	2.1	$\text{kg} \cdot \text{m}^2$
	I_{zz}	2.1	$\text{kg} \cdot \text{m}^2$
Body Length	l_{body}	0.600	m
Body Width	w_{body}	0.256	m
Body Height	h_{body}	0.200	m
Leg Link Lengths	l_1, l_2	0.34	m

bottom shank at the knee through the use of a chain. The reason for this is to have the motor modules, which are heavy, be as close to the body as possible and reduce the amount of inertia in the legs. This minimizes the effects of the leg inertia as they swing around and allows the use of simplified models for controlling the robot. Table 2.1 lists the main physical parameters for Cheetah 3 needed for the control model.

2.1.2 Hardware Design: Mini Cheetah

The Mini Cheetah robot was designed using the same design principles as Cheetah 3, but at a smaller scale. Actuators were designed to be low-cost, compact modules without sacrificing performance and are extensively described in [42]. The small scale means that it is harder to break parts as impact forces are smaller when falling from

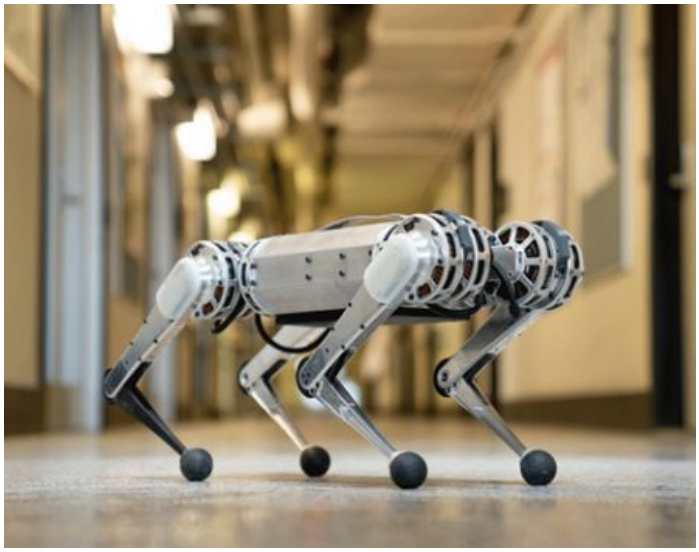


Figure 2-3: **Mini Cheetah Platform.** Small scale, electrically actuated, low cost, high performance quadruped robot.

a lower CoM height and has a smaller magnitude of momentum when crashing. Having also been built after Cheetah 3, improvements were made taking the experience gained from experiments running the larger robot. The detailed design can be found in [41]. As the robot was designed with the same philosophy and similar physical characteristics as Cheetah 3, the controllers and software were able to be used for locomotion control in the exact same way.

The important physical quantities that are needed for the controller are listed in Table 2.2. The smaller robot is about 5× less massive, but stands around 60% as

Table 2.2: Mini Cheetah Physical Robot Parameters

Parameter	Symbol	Value	Units
Mass	m	9	kg
Body Inertia	I_{xx}	0.07	$\text{kg} \cdot \text{m}^2$
	I_{yy}	0.26	$\text{kg} \cdot \text{m}^2$
	I_{zz}	0.242	$\text{kg} \cdot \text{m}^2$
Body Length	l_{body}	0.380	m
Body Width	w_{body}	0.203	m
Body Height	h_{body}	0.102	m
Leg Link Lengths	l_1	0.21	m
	l_2	0.19	m

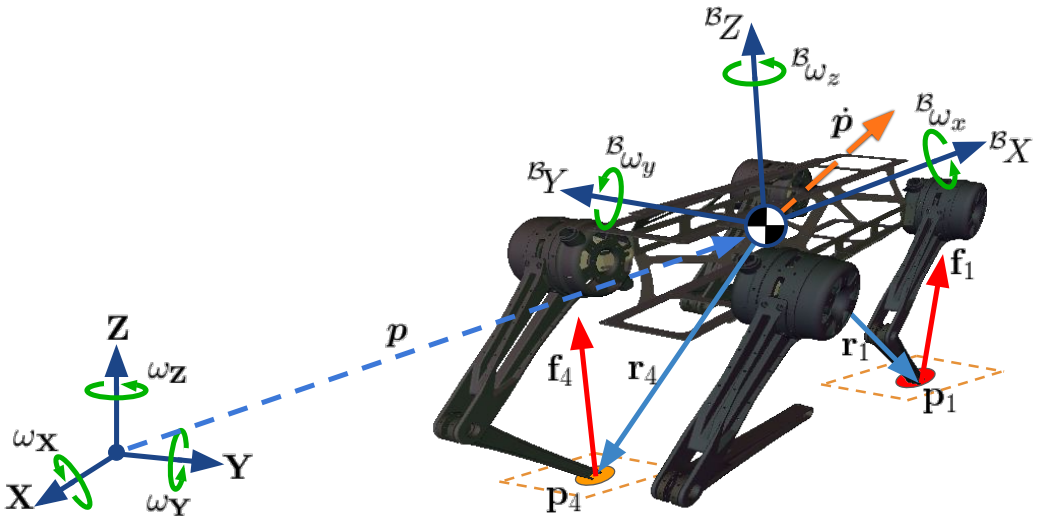


Figure 2-4: **Robot Coordinate System Definitions.** A body fixed coordinate frame, $\mathcal{B}$, originates at the robot's base center, from which the vectors to the footstep locations and forces can be defined.

tall. It is also worth noting that the upper and lower leg links are slightly different lengths. Diagrams of the actuators and leg kinematics, which are slightly different than Cheetah 3's, can be seen in [42]. The small-scale design is geared towards conducting rapid and frequent experiments.

2.2 Robot System Model Definitions

First it is important to clearly define the definitions for important physical quantities that describe the robot system. Both robot platforms are intended to have similar design principles meaning that the systems can be modeled the same with only the physical parameters and kinematics needing to be updated for the correct platform. In general, physical quantities, vectors, and matrices will be defined in the Inertial frame as denoted by a leading superscript, $^{\mathcal{I}}(\cdot)$, which originates at an arbitrarily defined, fixed origin, $^{\mathcal{I}}\boldsymbol{O}$. Usually this is set to the robot's initial position when initialized. Quantities expressed in the robot's body frame are signified by the leading superscript, $^{\mathcal{B}}(\cdot)$. Quantities where the frame is not explicitly defined can be assumed to be described in the Inertial frame.

Figure 2-4 depicts the floating base model of Cheetah 3 where the CoM is the important point of interest. Translational position and velocity are straightforward to define and follow standard practices. The position of the CoM is denoted by the vector from the origin,

$$\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{2.1}$$

where subscripts denote the respective (x, y, z) coordinates. The rate of change of this position vector denotes the velocity of the CoM in the Inertial frame,

$$\dot{\boldsymbol{p}} = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix}, \tag{2.2}$$

while the velocity of the CoM in the body frame is defined by

$${}^{\mathcal{B}}\dot{\boldsymbol{p}} = \begin{bmatrix} {}^{\mathcal{B}}\dot{p}_x \\ {}^{\mathcal{B}}\dot{p}_y \\ {}^{\mathcal{B}}\dot{p}_z \end{bmatrix}. \tag{2.3}$$

and the two velocity definitions are related by $\dot{\boldsymbol{p}} = {}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}}{}^{\mathcal{B}}\dot{\boldsymbol{p}}$, which is a coordinate frame transformation. The rotation matrix ${}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}}$ signifies a rotation from $\mathcal{B}$ to $\mathcal{I}$. Rotation matrices will be specified in the form where the leading superscript is the resultant coordinate frame and the subscript is the original coordinate frame.

Robot body orientation is less straightforward to define and there are a variety of ways to represent orientation. Each different way has benefits and limitations and are therefore used at different points throughout the robot system where appropriate.

The first is Euler angle representation

$$\mathbf{\Theta} = \begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix} \tag{2.4}$$

where the roll, θ, pitch, ϕ, and yaw, ψ, are successive rotations about the instantaneous axis. The first rotation is yaw, then pitch, then roll in this particular definition. Euler angle rates are written

$$\dot{\mathbf{\Theta}} = \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} \tag{2.5}$$

This is the most intuitive definition, but suffers from gimbal lock when $\phi = \frac{\pi}{2}$. To counteract this, the quaternion representation is sometimes used as well. The standard quaternion is $\mathbf{q} \in \mathbb{R}^{4\times 1}$ and ordered as

$$\mathbf{q} = \begin{bmatrix} q_r \\ q_i \\ q_j \\ q_k \end{bmatrix} \tag{2.6}$$

with q_r being the scalar portion. The derivative of the quaternion is standard as

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q}^{\mathcal{B}}\boldsymbol{\omega}. \tag{2.7}$$

Finally, the full rotation matrix, ${}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}} \in \mathbb{R}^{3\times 3}$, defined above is often used as well. The derivative for the rotation matrix representation is

$${}^{\mathcal{I}}\dot{\boldsymbol{R}}_{\mathcal{B}} = {}^{\mathcal{I}}\boldsymbol{R}_{\mathcal{B}}[{}^{\mathcal{B}}\boldsymbol{\omega}]_{\times} \tag{2.8}$$

where ${}^{\mathcal{B}}\boldsymbol{\omega}$ contains the instantaneous angular velocities around the robots body co-

ordinate frame axis and $\boldsymbol{\omega}$ contains the instantaneous angular velocities around the Inertial frame. Since the measurements from a gyroscope in an IMU give instantaneous angular rates about the IMU's local coordinate frame, it is often convenient to use this representation.

Leg kinematics are measured by encoders at each of the actuators for the Ab/Ad, $q_{Ab/Ad}$, hip, q_{hip}, and knee, q_{knee} in order. The angles for each are stacked in a vector for each leg

$$\boldsymbol{q}_i = \begin{bmatrix} q_{i,Ab/Ad} \\ q_{i,hip} \\ q_{i,knee} \end{bmatrix} \tag{2.9}$$

where $i \in \{FR, FL, BR, BL\}$ signifies the corresponding leg, front right (FR), front left (FL), back right (BR), and back left (BL). Forward kinematics gives the position of the end of the foot as

$$\boldsymbol{p}_i = \begin{bmatrix} p_{i,x} \\ p_{i,y} \\ p_{i,z} \end{bmatrix} \tag{2.10}$$

Since the legs are designed to be light compared to the body of the robot, the center of the base body and the CoM are approximately equal despite having a slight variation dependent on the leg configurations. Often they can be used interchangeably, but will be explicitly stated where necessary. These definitions are some of the more important physical descriptors that will be used throughout the dissertation and are listed in Appendix A for reference. Any further quantities that are used in a small number of specific sections are defined locally as needed.

2.3 Control Architecture

The control architecture used for both of the robots is depicted in Figure 2-5 and thoroughly described in [6]. Higher level trajectory commands are given by an operator

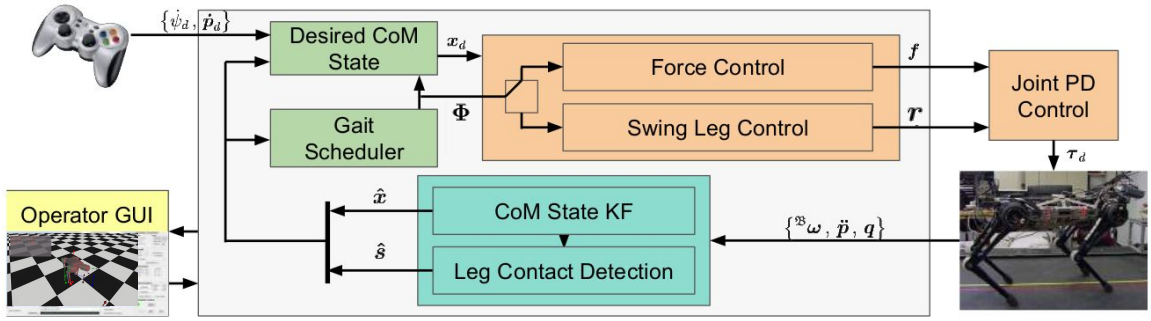


Figure 2-5: **System Architecture Block Diagram.** The user sends velocity and turning commands as well as general tunable parameters to the main computer. The main Cheetah control computer is composed of three main parts: higher-level planning (green), leg and body control (red), and state estimation (blue). The force and position commands are sent to the microcontrollers for each leg that relay the motor command to the robot.

through joystick commands or a predefined trajectory. These should be in the form of a desired translational velocity and turning rate where a separate planner can modify the commands to account for obstacles if a vision system is present and to give a full desired CoM state position, orientation, velocity, and angular rate trajectory to the locomotion controller. A legged gait is defined by the sequence of foot contact state changes which is flexibly defined by a gait scheduler and passed into the locomotion controller.

2.3.1 State Estimation

A state estimator determines the robot's state from the sensors in two decoupled parts. The sensor fusion algorithm first uses its IMU to estimate body orientation and angular rates, while the second estimates the body position and velocity from the accelerometer and body leg kinematics. The two parts combined provide an estimate for position, orientation, velocity, and angular rate for the body, while the encoders provide the angle of the leg joints.

Orientation is estimated either directly from proprietary algorithms provided by the IMU itself, or with a simple orientation filter on the raw gyroscope and accelerometer data similar to [50]. The gyroscope is used to estimate the high-frequency orientation dynamics, letting the accelerometer de-drift the estimate due to the gravity

bias at a lower frequency. The orientation estimate for the body relative to the Inertial frame, ${}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}}$, updates as

$$ {}^{\mathcal{I}}\dot{\hat{\boldsymbol{R}}}_{\mathcal{B}} = {}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}} \left[\boldsymbol{\omega} + \kappa \boldsymbol{\omega}_{corr}\right] \times \tag{2.11} $$

where $\kappa > 0$ is a correction gain and $\boldsymbol{\omega}_{corr}$ is a correction angular velocity to align the accelerometer reading ${}^{\mathcal{B}}\boldsymbol{a}$ with the gravity direction

$$ \boldsymbol{\omega}_{corr} = \frac{{}^{\mathcal{B}}\boldsymbol{a}}{\|{}^{\mathcal{B}}\boldsymbol{a}\|} \times {}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}}^{T} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.12} $$

The de-drifting time constant from this term can be approximated by κ^{-1}, which is in practice heuristically decreased during highly-dynamic portions of the gait where $\|{}^{\mathcal{B}}\boldsymbol{a}\| >> \boldsymbol{g}$ with

$$ \kappa = \kappa_{\text{ref}} \max(\min(1, 1 - ||{}^{\mathcal{B}}\boldsymbol{a} - \boldsymbol{g}||/\boldsymbol{g}), 0) \tag{2.13} $$

where $\boldsymbol{g}$ is the acceleration of gravity and κ_{ref} is chosen as $\kappa_{\text{ref}} = 0.1$. Roll and pitch are well-corrected, but yaw cannot be accurately de-drifted without knowledge of the absolute position relative to the surroundings such as with a magnetometer or a vision and LiDAR system [19].

The second part of the state estimation uses ${}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}}$ in addition to leg kinematics to estimate base position and velocity. By decoupling the two stages, the position and velocity estimates can be found using a standard linear Kalman Filter rather than an Extended Kalman Filter as in [10]. The reason for this is the simplicity of tuning and analysis as will as guaranteeing that the equations will not diverge in finite time.

The process equations for the body position, velocity, and foot i position in con-

tinuous time are modeled as

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{2.14}$$

$$\dot{\boldsymbol{v}} = {}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}}{}^{\mathcal{B}}\boldsymbol{a} + \boldsymbol{g} + \boldsymbol{w}_{\boldsymbol{v}} \tag{2.15}$$

$$\dot{\boldsymbol{p}}_i = \boldsymbol{w}_{\boldsymbol{p}_i} \quad \forall i = \{1, \ldots, 4\} \tag{2.16}$$

where white noise terms $\boldsymbol{w}_{\boldsymbol{v}}$ and $\boldsymbol{w}_{\boldsymbol{p}_i}$ characterize the noise in the accelerometer and any variability in the foot positions due to possible slipping. The foot process noise is set to a high value during swing to negate the effects of swing foot kinematics. With ${}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}}{}^{\mathcal{B}}\boldsymbol{a} + \boldsymbol{g}$ as an input to the system, the equations represent a linear time invariant process, and allowing us to use the regular Kalman Filter. The continuous time model is changed to discrete time with 1ms time steps using the same method as in [10].

Measurements of the position vector for each foot relative to the body are given by leg kinematics from encoder data assumed to be correct. With the relative foot position, $\boldsymbol{p}_{rel}(\boldsymbol{q}_i, {}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}})$, the measurement residual is given by

$$\boldsymbol{e}_{p,i} = (\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}) - \boldsymbol{p}_{rel}(\boldsymbol{q}_i, {}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}}) \tag{2.17}$$

Similarly, the foot velocity relative to the body is computed from leg angles, velocities, and body orientation and angular velocity by $\dot{\boldsymbol{p}}_{rel}(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, {}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}}, {}^{\mathcal{B}}\boldsymbol{\omega})$. Using an assumption that the foot positions are fixed, the residual is computed as

$$\boldsymbol{e}_{v,i} = (-\hat{\boldsymbol{v}}) - \dot{\boldsymbol{p}}_{rel}(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, {}^{\mathcal{I}}\hat{\boldsymbol{R}}_{\mathcal{B}}, {}^{\mathcal{B}}\boldsymbol{\omega}). \tag{2.18}$$

Finally, the assumed contact height h_i for each foot gives the measurement residual

$$\boldsymbol{e}_{h_i} = \left([0,\ 0,\ 1]^T \hat{\boldsymbol{p}}_i\right) - h_i. \tag{2.19}$$

Measurement errors for the residuals are described by random multivariate Gaussians. As with the process noise $\boldsymbol{w}_{\boldsymbol{p}_i}$, measurement covariances for foot are set to a

high value during swing so that swing leg measurements in the fusion are effectively ignored. While the state estimator itself is outside the scope of this work, we assume it will be adequate for controlling the robot.

2.3.2 Leg Control

Leg control can be achieved with several methods. For the purposes of our basic locomotion architecture, the swing feet are controlled by desired Cartesian positions for each foot that are converted into torques and the stance feet are controlled by desired ground reaction forces at the stance feet that are also converted into torques. Swing feet are given a Bezier trajectory originating at the takeoff location and ending at the desired next step location and described in the corresponding hip frame. To track the Cartesian position of the foot in space, the actuators are controlled by a combination of feedforward, $\boldsymbol{\tau}_{\mathrm{FF},i}$, and feedback torques, $\boldsymbol{\tau}_{\mathrm{FB},i}$, that are linearly combined to find the final torques for each actuator

$$\boldsymbol{\tau}_{\mathrm{FF},i} = \boldsymbol{J}_i^T \boldsymbol{\Lambda}_i ({}^{\mathcal{B}}\boldsymbol{a}_{i,\mathrm{ref}} - \dot{\boldsymbol{J}}_i \dot{\boldsymbol{q}}_i) + \boldsymbol{C}_i \dot{\boldsymbol{q}}_i + \boldsymbol{G}_i \tag{2.20}$$

$$\boldsymbol{\tau}_{\mathrm{FB},i} = \boldsymbol{J}_i^T [\boldsymbol{K}_p ({}^{\mathcal{B}}\boldsymbol{p}_{i,\mathrm{ref}} - {}^{\mathcal{B}}\boldsymbol{p}_i) + \boldsymbol{K}_d ({}^{\mathcal{B}}\boldsymbol{v}_{i,\mathrm{ref}} - {}^{\mathcal{B}}\boldsymbol{v}_i)] \tag{2.21}$$

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_{\mathrm{FF},i} + \boldsymbol{\tau}_{\mathrm{FB},i} \tag{2.22}$$

where $\boldsymbol{J}_i$ is the foot Jacobian, $\boldsymbol{\Lambda}_i$ is the operational space inertia matrix, ${}^{\mathcal{B}}\boldsymbol{a}_{i,\mathrm{ref}}$ is the reference acceleration for the swing trajectory, $\boldsymbol{q}_i$ is a vector of joint configurations, $\boldsymbol{C}_i$ is the Coriolis matrix, $\boldsymbol{G}_i$ is the torque due to gravity, and $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are diagonal matrices of proportional and derivative gains. These gains are scaled according to their configuration in order to ensure stability

$$K_{p,j} = \omega_{\mathrm{des}}^2 \Lambda_{jj}(\boldsymbol{q}) \tag{2.23}$$

where $K_{p,j}$ is the j-th diagonal entry in $\boldsymbol{K}_p$, ω_{des} is the desired natural frequency, and Λ_{jj} is the j-th diagonal entry in the operational space inertia matrix.

Using the proprioceptive actuators means that we can reasonably expect that

the torque commanded by the controller is the torque that will be produced with a small margin of error. Consequently, if the kinematics are described correctly, the force produced at the foot will also be close to the expected force. Therefore, the force controlled robot has several mid-level controllers that calculate desired ground reaction forces for the stance feet in order to track a body state or trajectory. Once appropriate forces are found, the actual torques required for the motors is found by

$$\boldsymbol{\tau}_i = \boldsymbol{J}_i^T \boldsymbol{f}_i. \tag{2.24}$$

This gives an accurate enough method for force control on the robots. As such, the calculation of the appropriate forces can be done in several ways. Previously, the MIT Cheetah 3 platform showed an instantaneous QP balance controller implementation in [6] which is similar to the one presented in [26], as well as a convex MPC for calculating current and future ground reaction forces for dynamic locomotion [12]. The RPC controller described throughout the remaining chapters focuses on calculating future footstep locations for the swing controller in addition to current and future ground reaction forces.

2.3.3 Gait Scheduling

Legged locomotion gaits are characterized by sets of discrete contacts as legs enter and leave stance and swing periods. The more legs in the system, the more contact modes exist. This is determined by 2^F, where F is the number of legs, so for a quadruped, there are $2^4 = 16$ discrete contact modes that can be achieved. For the purposes of the presented controller, the gait is defined outside and assumed to be achievable using a gait scheduling map such as the one pictured in Figure 2-6. The phase progresses as

$$\boldsymbol{\Phi}_i = \text{mod}\left(\frac{t - t_{0,i}}{T_P}, 1\right) \tag{2.25}$$

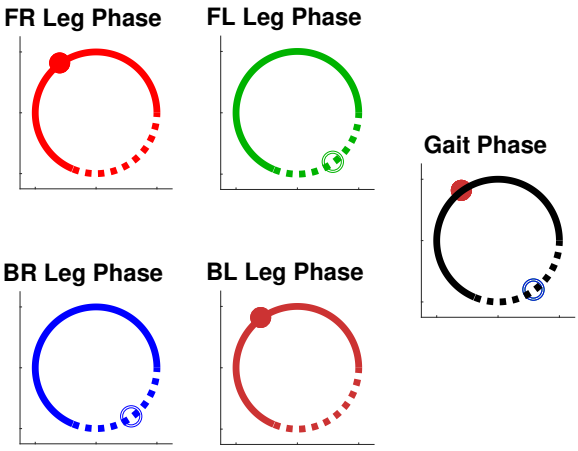


Figure 2-6: **Gait Scheduler Phase Map.** An overall phase variable, Φ_0, controls the gait cycle with each leg having an individual offset. The cycle includes contact (solid) and swing (dashed) states.

where the total gait period, T_P, is defined by the user. The scheduled contact states are defined to be $\boldsymbol{s}_\Phi \in \{1 : contact, 0 : swing\}$, where a discrete switch happens

$$\boldsymbol{s}_\Phi = \begin{cases} 0, & \text{if} \quad \boldsymbol{\Phi}_i > \boldsymbol{\Phi}_{i,c\to\bar{c}} \\ 1, & \text{if} \quad \boldsymbol{\Phi}_i \leq \boldsymbol{\Phi}_{i,c\to\bar{c}} \end{cases} \tag{2.26}$$

where $\boldsymbol{\Phi}_{i,c\to\bar{c}}$ is a user selected switching phase between 0 and 1. The defined gait map can be modified at each control iteration to provide flexible gait scheduling.

To avoid having the legs stay on the ground too long while in stance, the gait period is continuously adjusted by

$$T_P = \frac{2h\tan(\alpha_{max})}{\dot{\boldsymbol{p}} + \boldsymbol{\psi} \times \boldsymbol{r}} + T_{\bar{c}} \tag{2.27}$$

where α_{max} is a maximum leg angle from the hip to the foot, $T_{\bar{c}}$ is a predefined nominal swing time, and $\boldsymbol{\psi} = [0, 0, \psi]^T$. The resulting T_P is bounded my a maximum and minimum so as to not have too long or too short of a period. Too short of a stance period means the robot may not be able to have enough control authority to properly balance itself, while too long of a stance period will mean a traveling center of mass may exit the support polygon without a new upcoming foothold position.

This means the system will be unfavorably underactuated for a large period of time and be unable to exert needed forces and torques or the leg will no longer be able to reach its intended foot position due to extending itself past its kinematic limits.

Although the robot operates on a fixed schedule at each iteration, there are no guarantees that the robot will actually be in stance or swing when scheduled. Since neither robot has a direct force torque sensor, work was done to estimate ground contact at the feet through fusion of various probabilistic contact model priors [7]. The fusion is carried out using a standard linear Kalman Filter despite highly nonlinear dynamics by framing the predictions and measurements as probabilities of contact. The standard prediction equations for the Kalman filter are

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{A}_k \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}_k \boldsymbol{u}_k \tag{2.28}$$

$$\boldsymbol{\Sigma}_{k|k-1} = \boldsymbol{A}_k \boldsymbol{\Sigma}_{k-1} \boldsymbol{A}_k^T + \boldsymbol{\Sigma}_{w_k}. \tag{2.29}$$

Given the current phase of a leg, $\boldsymbol{\Phi}_i$, the gait scheduler provides an expected contact state, $\boldsymbol{s}_{\boldsymbol{\Phi},i}$, of each leg. Under ideal conditions, the expectation is that the actual contact state of the leg, $\boldsymbol{s}_i \in \{1 : contact, 0 : swing\}$, and $\boldsymbol{s}_{\boldsymbol{\Phi},i}$ are equivalent at all times. However, the robot is subject to possible timing delays in its control system or disturbances that may cause its leg to take off late, as well as early or late contacts due to inaccurate swing leg trajectory tracking or unforeseen ground height.

To handle these issues, a probabilistic model is built for the expectation of contact given the scheduled leg state and subphase during stance, $\boldsymbol{\Phi}_c$, or swing, $\boldsymbol{\Phi}_{\bar{c}}$. The subscripts denote contact, c, and no contact, $\bar{c}$. The probability of contact given the current contact state and subphase percent is chosen to take the form

$$\begin{aligned} P(c|\boldsymbol{s}_{\boldsymbol{\Phi}}, \boldsymbol{\Phi}) = \frac{1}{2} \Bigg(& \boldsymbol{s}_{\boldsymbol{\Phi}} \left[\operatorname{erf}\left(\frac{\boldsymbol{\Phi} - \mu_{c_0}}{\sigma_{c_0}\sqrt{2}} \right) + \operatorname{erf}\left(\frac{\mu_{c_1} - \boldsymbol{\Phi}}{\sigma_{c_1}\sqrt{2}} \right) \right] + \\ & \bar{\boldsymbol{s}}_{\boldsymbol{\Phi}} \left[2 + \operatorname{erf}\left(\frac{\mu_{\bar{c}_0} - \boldsymbol{\Phi}}{\sigma_{\bar{c}_0}\sqrt{2}} \right) + \operatorname{erf}\left(\frac{\boldsymbol{\Phi} - \mu_{\bar{c}_1}}{\sigma_{\bar{c}_1}\sqrt{2}} \right) \right] \Bigg) \end{aligned}$$

where $\boldsymbol{s}_{\boldsymbol{\Phi}}$ activates one half of the model depending on the state. The means, μ, encode the expected subphase value at which the contact state switches, while the

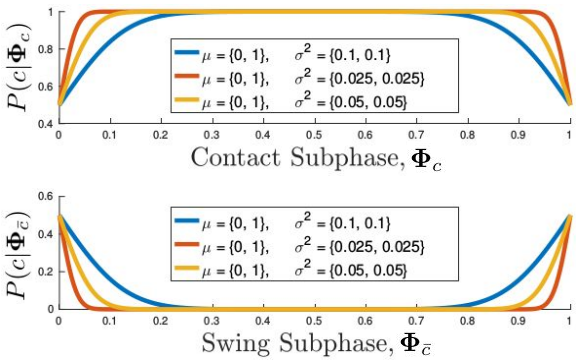


Figure 2-7: **Phase-Based Contact Model.** Various parameters defining the probability of being in contact given the scheduled state and percent of progress through the state subphase.

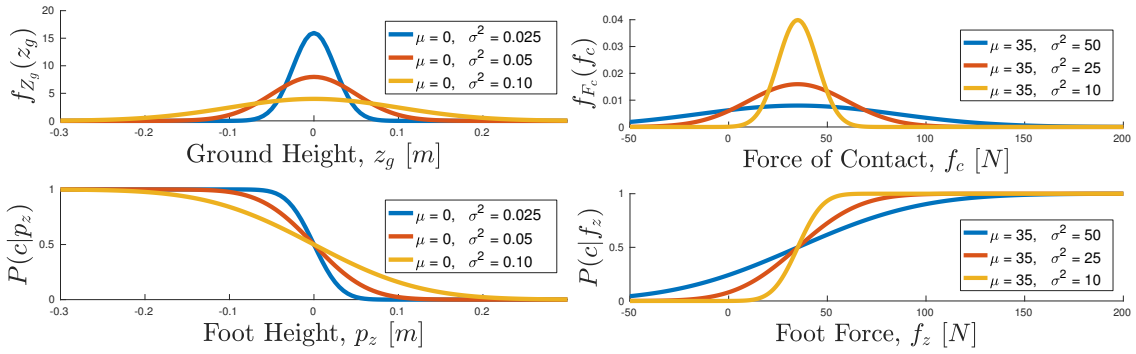
variance parameters, σ^2, are determined by the variation in the subphase value when contact actually occurs. Figure 2-7 shows the two curves for the phase and scheduled state-based probability of contact for three different chosen variances. Towards the beginning and end of the contact phase it is less certain that the leg is truly in contact, while it is more likely that the leg is in contact towards the beginning and end of swing. This model can therefore be used as the input to the system at the instantaneous timestep, k, as

$$\boldsymbol{u}_k = \begin{Bmatrix} P_1(c|\boldsymbol{s}_{\mathbf{\Phi}}, \mathbf{\Phi}) \\ \vdots \\ P_F(c|\boldsymbol{s}_{\mathbf{\Phi}}, \mathbf{\Phi}) \end{Bmatrix}_k \tag{2.30}$$

which is stacked for each of the robot's legs using each of their individually scheduled states and phases. Roughly, the following covariance matrix describes the amount of trust that we place on the accuracy of the robot to adhere to the phase-based switching model as

$$\mathbf{\Sigma}_{w_k} = \begin{bmatrix} \sigma^2_{\mathbf{\Phi},1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma^2_{\mathbf{\Phi},F} \end{bmatrix}_k . \tag{2.31}$$

Since we are simply considering instantaneous contact detection by fusing the cur-



(a) Foot Height Contact Model (b) Foot Force Contact Model

Figure 2-8: **Probabilistic Contact Model Measurement Priors.** Models describe the probability of contact as the foot swings above the ground, (2-8a), and as the robot estimates force at the foot (2-8b). Feet are more likely to be in contact as the swing height decreases and as estimated force increases.

rently available measurements rather than relying on previous contact probabilities, the state and input matrices are defined to be

$$\boldsymbol{A}_k = \mathbf{0}_F \quad \text{and} \quad \boldsymbol{B}_k = \mathbf{I}_F. \tag{2.32}$$

In the absence of an environmental perception system, ground height, z_g, can be modeled probabilistically. The random variable for the height of the ground is drawn from a Normal Gaussian distribution as $Z_g \sim \mathcal{F}(\mu_{z_g}, \sigma^2_{z_g})$. The defining parameter for average ground height, μ_{z_g}, is set to be zero with no external information since we cannot make assumptions about the terrain. Similarly, the variance loosely corresponds to the "roughness" of the terrain. As such, we can use the ground height model to create a belief for the height at which we expect to be in contact with the ground given the position of the foot in the vertical direction, p_z. The probability of contact given foot height is defined as

$$P(c|p_z) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{\mu_{z_g} - p_z}{\sigma_{z_g}\sqrt{2}}\right)\right] \tag{2.33}$$

Figure 2-8a shows both the Gaussian model of the ground, as well as the inferred probability of contact given foot height. Each is shown for different parameters corresponding to various roughness estimates.

Eventually, the values of μ_{z_g} and $\sigma_{z_g}^2$ can be adapted as we gain information from other sensors or historical footsteps. This model provides the first correction measurement with an associated covariance matrix

$$\tilde{\boldsymbol{z}}_{1,k} = \begin{Bmatrix} P_1(c|p_z) \\ \vdots \\ P_F(c|p_z) \end{Bmatrix}_k \quad \boldsymbol{\Sigma}_{v_{1,k}} = \begin{bmatrix} \sigma_{p_{z,1}}^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_{p_{z,F}}^2 \end{bmatrix}_k \tag{2.34}$$

as a measure of our belief in the validity of the contact height model.

The only true indicator of contact is external force felt at the foot. However, the robot is not currently equipped with a direct force sensor. With the force estimate presented in [7], we can create another simple probabilistic model for the force of contact with Gaussian random variable $f_c \sim \mathcal{F}(\mu_{f_c}, \sigma_{f_c}^2)$. Here μ_{f_c} is the average force sensed at the initiation of contact and $\sigma_{f_c}^2$ is a measure of the noise on the estimate. This gives the probability of contact given the estimated foot force as

$$P(c|f_z) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{f_z - \mu_{f_c}}{\sigma_{f_c}\sqrt{2}}\right)\right]. \tag{2.35}$$

The model and associated contact probability given vertical foot force are shown in Figure 2-8b. This force based probability provides our second measurement

$$\tilde{\boldsymbol{z}}_{2,k} = \begin{Bmatrix} P_1(c|f_z) \\ \vdots \\ P_F(c|f_z) \end{Bmatrix}_k \quad \boldsymbol{\Sigma}_{v_{2,k}} = \begin{bmatrix} \sigma_{f_{z,1}}^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_{f_{z,F}}^2 \end{bmatrix}_k \tag{2.36}$$

The two sets of individual measurements are stacked to form the observation measurement vector used in the Kalman Filter. Similarly, the covariance matrices of each measurement form an overall block diagonal covariance matrix as follows

$$\tilde{\boldsymbol{z}}_k = \begin{bmatrix} \tilde{\boldsymbol{z}}_{1,k} \\ \tilde{\boldsymbol{z}}_{2,k} \end{bmatrix} \qquad \boldsymbol{\Sigma}_{v_k} = \begin{bmatrix} \boldsymbol{\Sigma}_{v_{1,k}} & \mathbf{0}_F \\ \mathbf{0}_F & \boldsymbol{\Sigma}_{v_{2,k}} \end{bmatrix}. \tag{2.37}$$

The output matrix, $\boldsymbol{H}_k$, is formed as two stacked $F \times F$ identity matrices where F

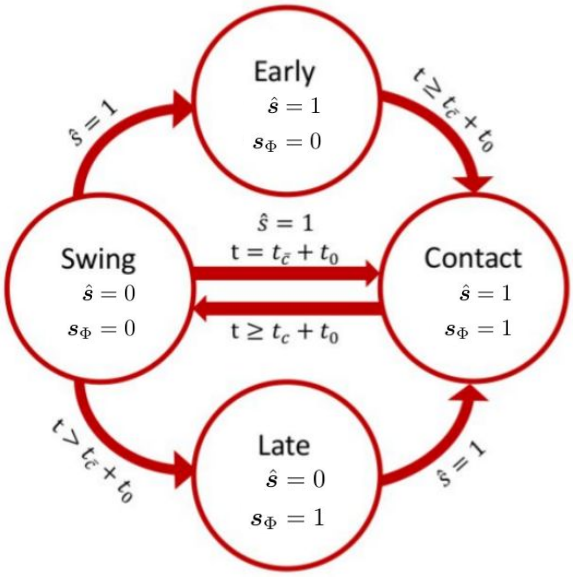


Figure 2-9: **Event-Based Gait Scheduling.** A finite state machine notifies the robot whether the robot is functioning under normal scheduled conditions or has encountered unexpected early or late contact.

is the number of legs.

$$\boldsymbol{H}_k = \begin{bmatrix} \mathbf{I}_F \\ \mathbf{I}_F \end{bmatrix} \tag{2.38}$$

By fusing the gait scheduler state expectations with the measurement model probabilities, we can get a better guess for the actual contact state of each leg. Owing to the fact that the Kalman filter is implemented with $\boldsymbol{A}_k = \mathbf{0}$, this fusion process could be derived through a static likelihood maximization with Bayes law. Yet, this process is one pre-embedded in standard Kalman updates, which may be more accessible in traditional robotics frameworks.

With accurate ground contact estimates, we can use an event-based gait modification finite state machine (FSM). The FSM, pictured in Figure 2-9, notifies the robot when an early contact has occurred, or if a scheduled contact is late and has not yet happened. If a leg has touched the ground early, it can now be used to produce ground reaction force rather than continue swinging into the object of contact, whereas if it is late, the controller can be aware not to rely on it to balance.

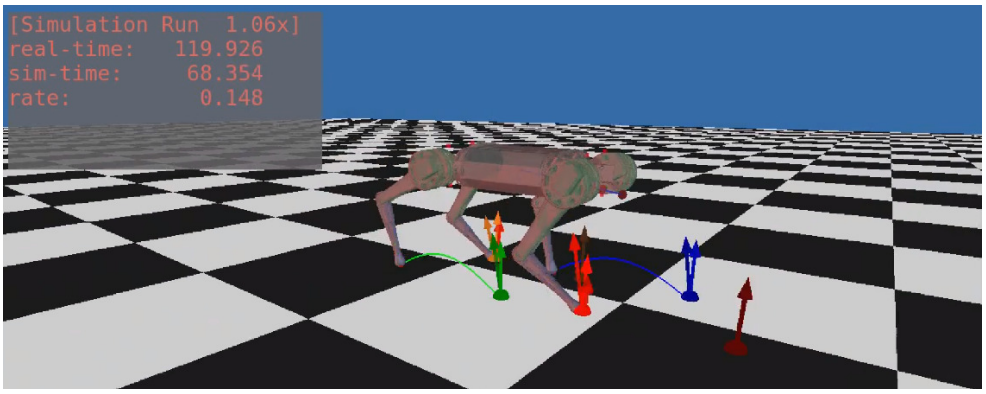


Figure 2-10: **Simulation Software Environment.** A custom realistic dynamics simulation software was designed to allow for fast, risk free controller development.

2.4 Simulation Environment

Experimentation on robotic hardware platforms can be tedious and risky. An accident during experiments can cause damage to the robot or operator. Damage to the robot can hinder development progress while the hardware is being repaired and parts can be costly to replace or manufacture. In robotics, it is often common practice to develop controllers and operating logic frameworks in simulation where there is no risk to the operator or hardware, and where the physics can be slowed down, paused, or sped up. Visualizations can also show information such as future step locations or trajectories as shown in Figure 2-10. Being able to control the environment and compare expected results for physical quantities that we can only estimate in the real world also makes it simple to test controllers directly without interference from other parts of the control architecture.

A realistic, open source simulation environment was developed by the MIT Biomimetic Robotics lab [54] in order to simulate the robot dynamics. It allows rapid, safe development and testing of controllers. Fast calculation of the dynamics is simulated using Featherstone's Articulated Body Algorithm (ABA) which has $\mathcal{O}(N)$ complexity [24]. ABA calculates the forward dynamics of the rigid body subtrees by making use of 6D spatial vector algebra [21, 22]. The simulation software is written in C++, making use of common libraries and software packages such as Eigen [33], Qt [13], and Featherstone's Spatial v2 [23].

Two different options are available for contact dynamics. The first is a nonlinear spring-damper that interacts with user defined points of contact along the robot. While effective, it requires using very small forward integration timesteps in order to ensure stability of the simulation. The second method is an optimization-based contact solver similar to the one presented in [36]. With this solver, approximate contacts are realistic enough to simulate the dynamics, but fast enough where simulations can be run at around $5\times$ real-time on most modern CPUs.

An interactive Graphical User Interface (GUI) allows online changes to both general simulation parameters, such as ground stiffness and damping, as well as any changes to the user defined controller parameters, such as gains or control modes. With this framework in place, the tuning and debugging processes were greatly accelerated and several different controllers were developed in parallel. The option on the GUI allows the user to select to run the simulation or deploy the code on the robot with a simple click, rather than developing for each type separately. Because of this, it is simple to test new algorithms in simulation and immediately be able to run the algorithm on the robot. Having an easy to use, physically realistic simulator is valuable to designing robots and algorithms.

Chapter 3

Regularized Predictive Control

The Regularized Predictive Controller is a nonlinear optimization-based control scheme that optimizes future robot states, footstep locations, and ground reaction forces by directly embedding regularization heuristics to exploit known physics and simple derived heuristic models. As opposed to traditional MPC techniques, RPC places a heavy emphasis on the regularization heuristics, while using highly simplified dynamics as a rough feasibility constraint rather than optimizing with a high-fidelity model. The heuristics are designed to simplify the complex cost space and find physically realizable solutions quickly. Rather than treating the problem of robot locomotion as a black box optimization, simple physics-based heuristics are encoded into the cost function and constraints to bias the optimization towards a sensible solution while remaining free to explore the surrounding cost space for the possibility of a better result.

Figure 3-1 shows a more straightforward version of the block diagram in 2-5 where the RPC handles the calculation of force control, as well as swing foot placements. It acts as the main locomotion and balance controller. Due to the modular architecture, it can easily be swapped in place of the previous convex MPC and QP-based balance controllers with purely heuristic foot placements. The other parts of the control architecture remain exactly the same and do not need to have any information about the type of controller being used for balance.

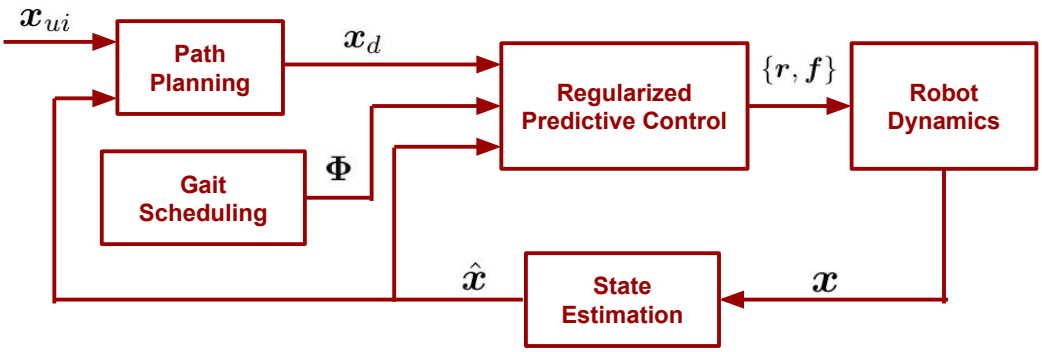
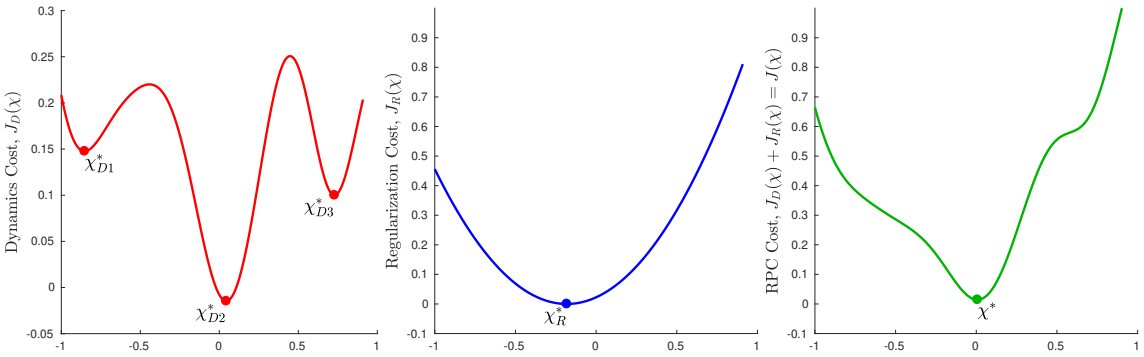


Figure 3-1: **Simplified Control Framework.** Simplified block diagram showing the general flow of information throughout the control framework. In theory, the RPC block is simply the optimization with no modifications needed.

3.1 Regularization in Optimization

Regularization is a technique used in various fields of mathematics and statistics in order to inject useful information to ill-posed problems that may be prone to overfitting [2]. Robot locomotion is particularly well suited for benefiting from regularization as it has been studied for many years and a wealth of knowledge is available to use as regularization heuristics. The philosophy behind RPC is to use our knowledge of robotic and legged systems to inform the optimization. There is no point in letting the optimization do extra iterations to find a more globally optimal solution when a "good enough" heuristic will be a sufficient initial guess. If a working control law has been found for a certain situation, there is no need for the optimization to have to find it again each time.By shaping the cost function with the regularization heuristic, we bias results towards solutions we know may work, while allowing the optimization to find solutions for cases that are not well studied or have no simple description.

Regularization is a common, well-studied field, often used to simplify complicated cost spaces that are prone to many local minima and overfitting in optimization and machine learning. Figure 3-2a shows a normalized nonlinear cost function that has several local minima and is ill conditioned, meaning that depending on the initial guess, a completely different solution will be found. Occasionally, this may be desirable, but if we have a guess for the general region where we expect the solution can be found, we can create a regularization function, as seen in Figure 3-2b. This is often



(a) $J_D(\boldsymbol{\chi})$ (b) $J_R(\boldsymbol{\chi})$ (c) $J(\boldsymbol{\chi})$

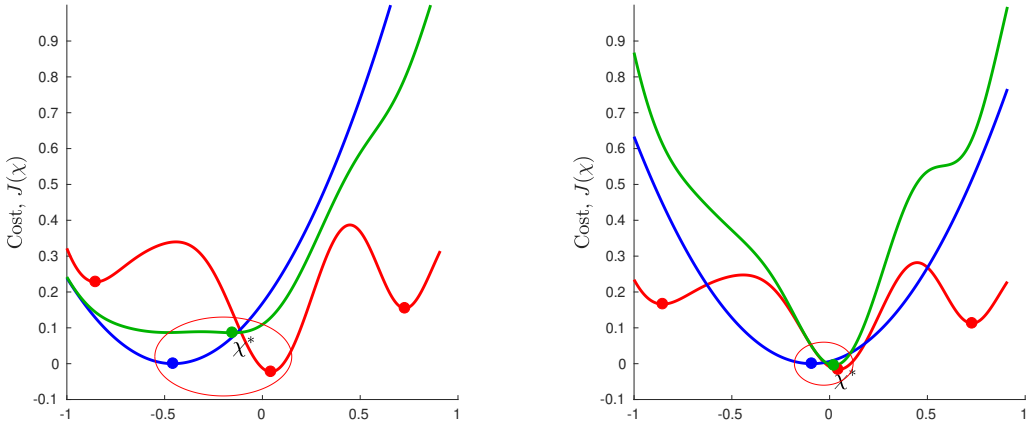
Figure 3-2: **Regularizing Nonlinear Cost Functions.** A generic nonlinear cost function (3-2a) with multiple local minima is additively regularized with a simple quadratic cost function (3-2b). The resulting cost function (3-2c) captures a smoothened nonlinear function with only one minimum.

a simple, convex function with a known solution. By adding it to the nonlinear cost function, the combined regularized cost becomes less nonlinear, more convex, and may even eliminate some extraneous local minima, as shown in Figure 3-2c. While a simple concept. designing the regularization function in a way that does not overpower the underlying cost function or that smooths the resulting function favorably is difficult and not always clear.

Qualitatively, a heuristic is good if it causes the system to behave as desired when executed and shapes the cost function favorably. Quantitatively, a set of heuristics is considered good if the optimal solution of the dynamics portion and the heuristic regularization portion of the cost function are approximately equal to each other, as well as the combined cost optimal for all states within the set of desired reasonable locomotion states written as

$$\boldsymbol{\chi}^*_R \approx \boldsymbol{\chi}^*_D \approx \boldsymbol{\chi}^*, \forall\{\hat{\boldsymbol{x}} | \hat{\boldsymbol{x}} \in \boldsymbol{X}\} \tag{3.1}$$

This roughly states that the heuristic regularization is well suited to be a solution for adequate robot behavior given the current state and the desired behavior while smoothing out the cost function to be more convex. This viable operating space, $\boldsymbol{X}$, is intentionally defined vaguely to allow for different desired performance metrics.



(a) Bad Regularization (b) Good Regularization

Figure 3-3: **Designing Regularization Functions.** Regularization functions should smooth the dynamics function while not having a significant effect on the optimal solution.

For example, the set of heuristics for quasi-static locomotion may not be adequate for more dynamic movement. What may be considered failure for one robot, may be perfectly within reasonable locomotion for another robot. While the individual parts of the cost function may look very different from each other, the optimal solutions for each part should be near each other in most cases.

The choice of regularization function on the same nonlinear cost can have drastically different effects. Figure 3-3a shows an example of bad regularization as the cost is shifted over, reshaping the combined function. The resultant optimal solution is in between the nonlinear and the regularized ones and the function has a large flat section. This makes it more difficult for the solver to find the minimum and when it does, the minimum is actually midway between the local minimum and maximum of the original cost. The well-designed regularization function in Figure 3-3b results in the combined optimal solution near to the original nonlinear optimal, but shapes it favorably. This is an example of the goal when creating regularization functions.

In practice, we don't know the shape, nor the solution of the original cost function. If we did, then there would be no need for optimization. However, we can make educated guesses as to where good local minima are from our analysis of legged locomotion principles. By embedding the possible guesses, we can guide the opti-

mization requiring less iterations to solve. The regularization has no guarantees of being placed near the global optimal, but should find a minimum that is at least good enough temporarily.

3.2 Simplified Dynamics Control Model

The full robot system model is a complex, highly nonlinear 18 DoF system described by 37 states. This creates a large amount of nonlinear terms to solve in the force control optimization. Optimization based controllers often use simplified template models to reduce the amount of computation needed. Both robots were specifically designed to have light limbs to mitigate the effects of leg inertias. Because of this, the control model is chosen to model massless legs, with the body mass and inertia absorbing the actual mass of the legs. With this lumped mass control model, the full robot state is defined to be

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{\Theta} \\ \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\Theta}} \end{bmatrix} \tag{3.2}$$

where the position in the the world frame as before, $\boldsymbol{p} = [x, y, z]^T$, orientation is described by Euler angles, $\boldsymbol{\Theta} = [\theta, \phi, \psi]^T$, signifying roll-pitch-yaw, (θ, ϕ, ψ), translational velocity also in the world frame, $\dot{\boldsymbol{p}} = [\dot{x}, \dot{y}, \dot{z}]^T$, and rotational velocity as the Euler angle rates, $\dot{\boldsymbol{\Theta}} = [\dot{\theta}, \dot{\phi}, \dot{\psi}]^T$. The state describing the dynamics of the robot is therefore simplified as, $\boldsymbol{x} \in \mathbb{R}^{12\times 1}$.

For the force controlled legged robot model, inputs of the system are chosen to be the vectors from the CoM to the footstep positions and their respective ground

reaction forces for each of the feet stacked for all feet as

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{r}_F \\ \boldsymbol{f}_F \end{bmatrix}. \tag{3.3}$$

where $\boldsymbol{r}_i = [r_{x,i}, r_{y,i}, r_{z,i}]^T$ and $\boldsymbol{f}_i = [f_{x,i}, f_{y,i}, f_{z,i}]^T$ are the 3D foot position vector from the CoM and ground reaction force vector respectively for foot i. Both the footstep vectors and forces are described in the inertial world frame. The framework is created in a way that can solve for any number of feet, F, where $F = 4$ for the implementation of the quadruped model. As shown previously, ground reaction forces and footstep locations are sufficient for control of the robot.

Finally, we arrive at the simplified discrete dynamics written

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}(\Delta t_k)\boldsymbol{x}_k + \boldsymbol{B}(\Delta t_k)h(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\Phi}_k) + \boldsymbol{d}(\Delta t_k) \tag{3.4}$$

where Δt_k is the time taken between timesteps k and $k+1$. Due to the choice of states and model simplifications, the state transition, $\boldsymbol{A}(\Delta t_k)$, input, $\boldsymbol{B}(\Delta t_k)$, and external disturbance, $\boldsymbol{d}(\Delta t_k)$, matrices can be written as time-varying, linear matrices

$$\boldsymbol{A}(\Delta t_k) = \begin{bmatrix} \mathbf{I}_6 & \Delta t_k \mathbf{I}_6 \\ \mathbf{0}_6 & \mathbf{I}_6 \end{bmatrix} \qquad \boldsymbol{B}(\Delta t_k) = \begin{bmatrix} \frac{\Delta t_k^2}{2}\boldsymbol{I}^{-1} \\ \Delta t_k \boldsymbol{I}^{-1} \end{bmatrix} \qquad \boldsymbol{d}(\Delta t_k) = \begin{bmatrix} \frac{\Delta t_k^2}{2}\boldsymbol{a}_g \\ \Delta t_k \boldsymbol{a}_g \end{bmatrix} \tag{3.5}$$

with the disturbance being the acceleration due to gravity, $\boldsymbol{a}_g = [\boldsymbol{g}^T, \mathbf{0}_{3\times1}^T]^T$.

As inputs to the system, the robot is searching for footstep locations for the swing feet and ground reaction forces for the stance feet which cannot be written in the typical linear dynamic system state space representation with the chosen states. As such, the inputs must be written in terms of forces and torques on the CoM as the

nonlinear input relating footstep locations and forces as

$$h(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\Phi}_k) = \begin{bmatrix} \boldsymbol{f} \\ {}^{\mathcal{B}}\boldsymbol{\tau} \end{bmatrix} = \sum_{i=1}^{4} \begin{bmatrix} \mathbf{I}_3 \\ \boldsymbol{R}_z(\psi_k)^T \begin{bmatrix} \boldsymbol{r}_{i,k} \end{bmatrix}_\times \end{bmatrix} \boldsymbol{s}_{\Phi,i,k} \boldsymbol{f}_{i,k} \tag{3.6}$$

with $\boldsymbol{R}_z(\psi_k)$ being the rotation matrix to describe the rotation of the robot into the yaw rotated body frame with no roll, $\theta = 0$, or pitch, $\phi = 0$. While the robot, may still have a roll and pitch, it is expected that they should be small under normal locomotion conditions. The orientation dynamics are simplified to reduce the amount of nonlinearity in the model for better computation times, while capturing the key components describing the robot state.

3.2.1 Justification for Simplified Orientation Dynamics

As mentioned in equation (3.6), the deliberate choice was made to represent the orientation of the robot in the control model only taking into consideration the yaw rotation of the body, ψ, rather than include the roll, θ, and the pitch, ϕ. in cases where the roll and pitch are close to zero, the full rotation matrix, $\boldsymbol{R}(\boldsymbol{\Theta})$, and the yaw rotation matrix, $\boldsymbol{R}_z(\psi)$, are equal

$$\lim_{(\theta,\phi)\to(0,0)} \boldsymbol{R}(\boldsymbol{\Theta}) = \boldsymbol{R}_z(\psi). \tag{3.7}$$

Since the robot is never expected to have a large roll or pitch during normal operation for locomotion which is the intent of this controller, it is sufficient to make this approximation. This is valid as long as the body fixed ${}^{\mathcal{B}}z$ axis remains within a reasonably bounded cone around the world $\mathbf{Z}$ axis.

The torque around the CoM in the robot body frame gives the angular acceleration of the body in the body frame as

$${}^{\mathcal{B}}\dot{\boldsymbol{\omega}} = {}^{\mathcal{B}}\bar{\boldsymbol{I}}^{-1}{}^{\mathcal{B}}\boldsymbol{\tau}, \tag{3.8}$$

where the velocity product terms ${}^{\mathcal{B}}\boldsymbol{\omega} \times {}^{\mathcal{B}}\bar{\boldsymbol{I}}\,{}^{\mathcal{B}}\boldsymbol{\omega}$ are neglected since they are generally

small in practice. Since our robot orientation angular rate is defined as the Euler angle rates, we can convert the body frame fixed angular rates according to

$$\dot{\boldsymbol{\Theta}} = \boldsymbol{B}(\boldsymbol{\Theta})\, {}^{\mathcal{B}}\boldsymbol{\omega} \tag{3.9}$$

where $\boldsymbol{B}(\boldsymbol{\Theta}) \in \mathbb{R}^{3\times 3}$ is the Euler rate matrix from [82]. This has a complex form that introduces many nonlinearities into the problem. The rational behind the small pitch and roll assumption to create equation (3.7) is used again to approximate this matrix as

$$\lim_{(\theta,\phi)\to(0,0)} \boldsymbol{B}(\boldsymbol{\Theta}) = \mathbf{I}_3. \tag{3.10}$$

With this we can arrive at the crucial approximation that the Euler angle rates are approximately equal to the body frame fixed angular rates

$$\dot{\boldsymbol{\Theta}} \approx {}^{\mathcal{B}}\boldsymbol{\omega}. \tag{3.11}$$

The approximations are applied to the dynamics meaning that the robot is aware of its orientation and tracks it over time. However, the instantaneous As such, all nonlinearities disappear from the calculation of torques on the robot body besides equation (3.6), which allows the simplified discrete dynamics to use a linear transition matrix, $\boldsymbol{A}(\Delta t)$, defined in (3.5).

3.3 Nonlinear Optimization Formulation

The decision variables in the optimization at each timestep, k, are the stacked vector of the robot states and inputs at the corresponding timestep

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{\Theta}_k \\ \dot{\boldsymbol{p}}_k \\ \dot{\boldsymbol{\Theta}}_k \end{bmatrix} \qquad \boldsymbol{u}_k = \begin{bmatrix} \boldsymbol{r}_{1,k} \\ \boldsymbol{f}_{1,k} \\ \vdots \\ \boldsymbol{r}_{F,k} \\ \boldsymbol{f}_{F,k} \end{bmatrix} \qquad \boldsymbol{\chi}_k = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix}. \tag{3.12}$$

This gives the total number of decision variables at k to be $n_{\boldsymbol{\chi}_k} = n_{\boldsymbol{x}_k} + n_{\boldsymbol{u}_k}$, which for the quadruped with simplified dynamics is $36 = 12 + 24$.

Each of the decision variables is regularized by error terms that bias the solution towards regularization heuristic functions. The heuristic regularization error terms are defined to be

$$\tilde{\boldsymbol{\chi}} = \boldsymbol{\mathcal{H}}_{\boldsymbol{\chi}}(\boldsymbol{\chi}, \Phi, \boldsymbol{x}_d) - \boldsymbol{\chi} \tag{3.13}$$

and are dependent on the states, inputs, gait phase, and desired commands. A comprehensive list of the active heuristic functions used can be found in Appendix C. These include analytically designed heuristics from simple physics analysis, as well as data-driven extracted heuristics.

The full Regularized Predictive Control nonlinear optimization problem is written in the general form as

$$\begin{aligned} \min_{\boldsymbol{\chi}} \quad & J(\boldsymbol{\chi}) = \sum_{k=0}^{N-1} \tilde{\boldsymbol{\chi}}_k^T \boldsymbol{W}_k \tilde{\boldsymbol{\chi}}_k \\ \text{subject to} \quad & \hat{\boldsymbol{x}}_{k+1} = f(\Delta t_k, \boldsymbol{\chi}_k, \boldsymbol{s}_{\Phi,k}) \\ & \boldsymbol{\zeta}_k(\boldsymbol{\chi}_k, \boldsymbol{s}_{\Phi,k}) \leq 0 \\ & \boldsymbol{\zeta}'_k(\boldsymbol{\chi}_k, \boldsymbol{\chi}_{k+1}, \boldsymbol{s}_{\Phi,k}, \boldsymbol{s}_{\Phi,k+1}) \leq 0 \end{aligned} \tag{3.14}$$

where instantaneous constraints are denoted with $\boldsymbol{\zeta}_k$ and transition constraints be-

tween decision variables at the current and next time step are $\boldsymbol{\zeta}_k'$. A full description of the cost function and constraints is clearly laid out in Appendix B. The total number of decision variables at k is represented as $\boldsymbol{\chi}_k \in \mathbb{R}^{n_{\boldsymbol{\chi}_k} \times 1}$, giving the total decision variables for the N timesteps, $n = \sum_{k=0}^{N-1} n_{\boldsymbol{\chi}_k}$ as $\boldsymbol{\chi} \in \mathbb{R}^{n \times 1}$. Similarly, the number of constraints, m, is the number of constraints at each timestep, $\boldsymbol{\zeta}_k(\boldsymbol{\chi}_k) \in \mathbb{R}^{m_{\boldsymbol{\zeta}_k} \times 1}$, summed over N timesteps, $m = \sum_{k=0}^{N-1} m_{\boldsymbol{\zeta}_k}$, resulting in $\boldsymbol{\zeta}(\boldsymbol{\chi}) \in \mathbb{R}^{m \times 1}$. Each of the heuristic regularization error terms is weighted by a block diagonal matrix of decision variable weights

$$\boldsymbol{W}_k = \begin{bmatrix} \boldsymbol{Q}_k & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_k \end{bmatrix} \tag{3.15}$$

which is composed of the cost weights at the timestep k for the states, $\boldsymbol{Q}_k \in \mathbb{R}^{n_{\boldsymbol{x}_k} \times n_{\boldsymbol{x}_k}}$, and inputs, $\boldsymbol{R}_k \in \mathbb{R}^{n_{\boldsymbol{u}_k} \times n_{\boldsymbol{u}_k}}$ resulting in $\boldsymbol{W}_k \in \mathbb{R}^{n_k \times n_k}$. These values are user selected based on the relative importance of each decision variable.

The optimization is carried out using the freely available nonlinear program (NLP) solver IPOPT [81] which requires the calculation of gradients and allows the user to define the Hessian of the Lagrangian. The Jacobian of the constraints is matrix $\mathbb{J}_{\boldsymbol{\zeta}}(\boldsymbol{\chi}) \in \mathbb{R}^{m \times n}$ is designated by

$$\mathbb{J}_{\boldsymbol{\zeta}}(\boldsymbol{\chi}) = \nabla \boldsymbol{\zeta}(\boldsymbol{\chi})^T$$

which is dependent on all the decision variables over the prediction horizon. The Hessian of the Lagrangian, $\mathbb{H}_{\mathcal{L}}(\boldsymbol{\chi}, \boldsymbol{\zeta}(\boldsymbol{\chi})) \in \mathbb{R}^{n \times n}$ takes the form

$$\mathbb{H}_{\mathcal{L}}(\boldsymbol{\chi}, \boldsymbol{\zeta}(\boldsymbol{\chi})) = \sigma_f \nabla^2 J(\boldsymbol{\chi}) + \sum_{j=0}^{m-1} \lambda_j \nabla^2 \zeta_j(\boldsymbol{\chi}) \tag{3.16}$$

where σ_f is an objective factor and λ_j is the Lagrange multiplier for constraint j and is part of the vector $\boldsymbol{\lambda} \in \mathbb{R}^{m \times 1}$.

By linking only the current and next future timestep together, the optimization constrains dynamics to be continuous between consecutive timesteps, which in turn

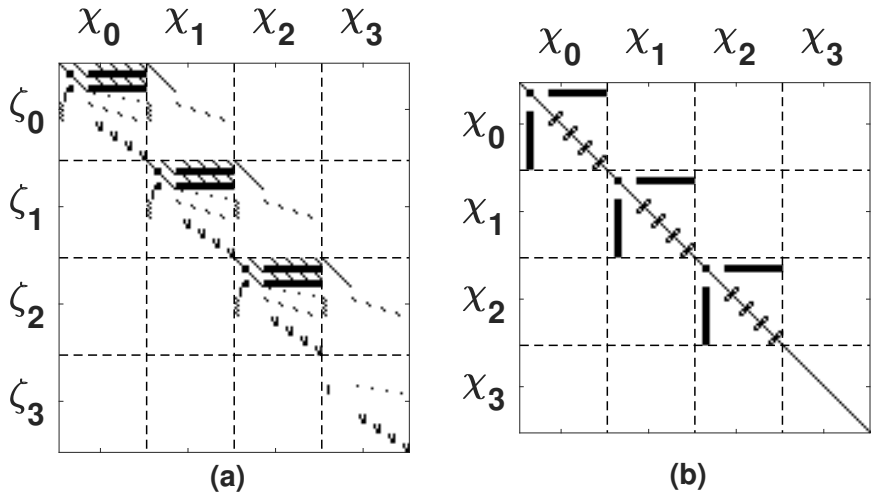


Figure 3-4: **Sparsity Patterns.** Both the constraint Jacobian (a) and Lagrangian Hessian (b) feature a highly sparse, repeated structure throughout the matrix which contributes to the flexibility and speed of the algorithm.

implicitly results in continuous dynamics throughout the prediction horizon without explicitly writing every timestep state as dependent on every other timestep. This reduces the amount of computations needed in the constraint Jacobian and the Hessian of the Lagrangian. Figure 3-4 shows the sparsity pattern for the constraint Jacobian and the Hessian of the Lagrangian. The dark spots signify entries that may be nonzero, while the white areas are guaranteed to have zero values. The subscripts denote the prediction timestep number, k. In this example $N = 4$, but in practice it can be chosen to be anything due to the repeated structure. By design, the matrices are both sparse and although the size of the Constraint Jacobian and Lagrangian Hessian grow proportional to N^2, the non-zero entries of each grows linearly as $\mathcal{O}(N)$.

For this problem, only about 10% of the entries in each of the matrices are possibly nonzero depending on the chosen prediction horizon number, N, and the number of stance feet active at each timestep in the prediction horizon as defined by the gait scheduler. It is also worth noting that since the Hessian of the Lagrangian is symmetric, in practice only the lower triangular portion needs to be calculated which reduces the actual calculations actually carried out significantly.

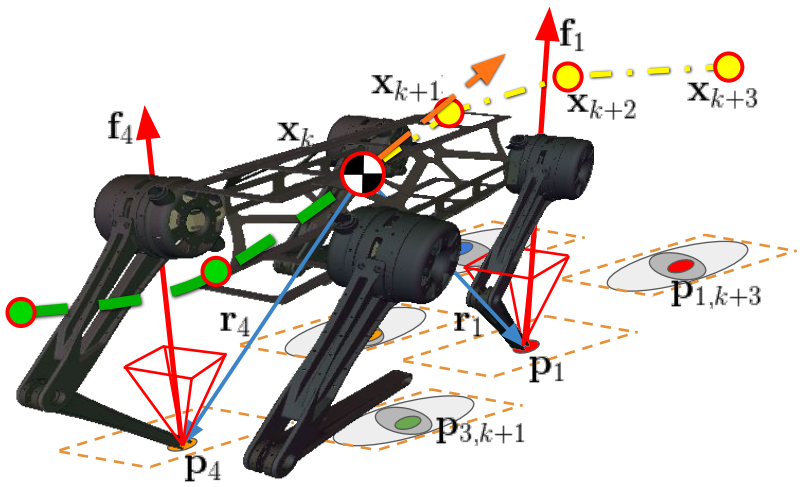


Figure 3-5: **RPC Optimization.** Physically realistic constraints are enforced for the simplified dynamics, footsteps locations, and ground reaction forces.

3.4 Physical Feasibility Constraints

The cost function itself finds solutions for future states and inputs that best cause the robot to achieve the desired behavior. However, it does not guarantee that the solution found is realizable by the robot. Therefore, the optimization is subject to a set of constraints to encode the physical feasibility of the solution. Constraints are depicted in Figure 3-5 and represent a minimum viable representation of physical feasibility that describes legged locomotion.

As with most MPC techniques, the predicted states are constrained to the system dynamics. An equality constraint is written that states the solution must adhere to the simplified discrete dynamics

$$\boldsymbol{x}_{k+1} - (\boldsymbol{A}(\Delta t_k)\boldsymbol{x}_k + \boldsymbol{B}(\Delta t_k)h(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{s}_{\Phi,k}) + \boldsymbol{d}(\Delta t_k)) = 0 \tag{3.17}$$

between timesteps k and $k+1$. This is using the heavily simplified dynamics that treats the robot as having massless legs and considers only the lumped mass CoM dynamics as described in Section 3.2. Because this simplified model does not account for coupled dynamics and the effects of leg inertia, this constraint is designed to be a rough feasibility constraint, rather than an accurate dynamic representation of the full

robot. The dependence on regularization heuristics, rather than an accurate model is what differentiates RPC as a subset of traditional MPC techniques.

Next, the constraints must enforce that the optimized footstep locations are physically possible. Contact feet as defined by the gait scheduler must be in contact with the ground throughout the stance period

$$\boldsymbol{s}_{\Phi,i,k} \left(\hat{z}_g(p_{x,i,k}, p_{y,i,k}) - p_{z,i,k} \right) = 0 \tag{3.18}$$

where $\hat{z}_g(p_{x,i}, p_{y,i})$ is the estimated ground height at the footstep location coordinates projected in the XY plane. In the absence of terrain environment from a perception system, the ground height is set to be 0 by default so $\hat{z}_g(p_{x,i,k}, p_{y,i,k}) = 0$. In the case of perceived rough terrain, the ground height would need to be described with a smooth representation to allow for gradients to be written.

Footsteps must be within the kinematic limits of the robot legs. If this in not ensured, solutions may be found with footsteps in locations where the robot cannot physically reach. The kinematic constraint is approximated to

$$\boldsymbol{s}_{\Phi,i,k} \left(\left\| \boldsymbol{r}_{i,k} - {}^{\mathcal{B}}\boldsymbol{r}_{h,i,k} \right\| - p_{z,k} \tan(\beta) \right) \leq 0 \tag{3.19}$$

where ${}^{\mathcal{B}}\boldsymbol{r}_{h,i}$ is the vector from the CoM to the corresponding hip in the body frame and β is a maximum kinematic leg angle that the robot can reach. This can be a user defined maximum or set to be dependent on the combined leg length and CoM height off the ground.

A constraint must be enforced that states the footstep location cannot change between iterations if the foot is in contact for both timesteps signifying a no slip condition. The binary contact state for timesteps k and $k+1$ are multiplied together

$$\boldsymbol{s}_{\Phi,i,k+1} \boldsymbol{s}_{\Phi,i,k} \left(\boldsymbol{p}_{i,k+1} - \boldsymbol{p}_{i,k} \right) = 0 \tag{3.20}$$

This links the footsteps between consecutive timesteps where the foot is in contact. If the foot is lifted from stance at $k+1$, the constraint is automatically satisfied as

$\boldsymbol{s}_{\Phi,i,k+1} = 0$ which means that the discrete contact state switching is automatically handled and does not require explicitly writing a different optimization for each gait sequence. The no slip constraint does not mean that the solution falls apart if slip does occur as $\boldsymbol{p}_{i,0}$ is updated at the beginning of each solve with the actual location of the feet.

Finally, the ground reaction forces must be physically realizable and cannot pull on the contact surface. Ground reaction forces must be positive with respect to the ground surface normal at the step location

$$-\boldsymbol{s}_{\Phi,i,k}\left(\boldsymbol{f}_{i,k}\cdot\nabla\hat{\mathbb{G}}(\boldsymbol{p}_{i,k})\right)\leq 0. \tag{3.21}$$

Again, in the absence of a perception system to estimate ground surface normals, the ground is assumed to be flat with a positive surface normal pointing directly upwards along the Inertial $\mathbf{Z}$ axis, $\nabla\hat{\mathbb{G}}(\boldsymbol{p}_{i,k}) = [0,0,1]^T$. Despite this simplified assumption, the controller is able to walk over slopes and curved terrain.

Forces cannot be arbitrarily applied at the contact feet as too large of a lateral force may cause slipping of the surface. A constraint is written to avoid violating the friction constraints

$$\begin{aligned} -\mu\boldsymbol{f}_{z,i} \leq \boldsymbol{f}_{x,i} \leq \mu\boldsymbol{f}_{z,i} \\ -\mu\boldsymbol{f}_{z,i} \leq \boldsymbol{f}_{y,i} \leq \mu\boldsymbol{f}_{z,i}. \end{aligned} \tag{3.22}$$

The use of a pyramid representation instead of a circular cone allows describing simple linear constraints. Though by using the pyramidal representation of lateral friction, a region of the actual friction cone is cut off, it is small and reasonable in practice. The coefficient of friction, μ, is user defined if no perception system or sensor is in place to estimate ground friction.

In reality, the robot has actuator and operating limits that it cannot exceed. All of the decision variables are subjected to maximum and minimum limits

$$\boldsymbol{\chi}_{min} \leq \boldsymbol{\chi} \leq \boldsymbol{\chi}_{max}. \tag{3.23}$$

For example, the pitch and roll orientation of the robot is set to be less than a magnitude of $\frac{\pi}{2}$ in either direction to avoid gimbal lock in pitch and keep the robot with a mostly flat orientation. A minimum ground reaction force is enforced to keep the contact feet pushing on the ground. While maximum force is dependent on actuator torque limits and leg configuration, a sensible maximum force is used to approximate this during normal locomotion conditions. By bounding the decision variables, the search space is also kept withing a small manageable region. This set of constraints is chosen to be the simplest, minimum description for approximating physical feasibility of a legged locomotion system.

Chapter 4

Extracting Legged Locomotion Heuristics

The key idea behind RPC is to regularize the decision variables towards heuristics that describe locomotion in the optimization cost function. Injecting simple physics-based heuristics into the optimization has proven to be beneficial. However, designing these heuristics can be challenging and unintuitive. Using the basic RPC implementation, Figure 4-1 describes a methodology for extracting and designing principled heuristics for legged locomotion. By allowing a simulation to fully explore the cost space offline, certain states and actions can be constrained or isolated. Data is fit with simple models relating the desired commands, optimal control actions, and robot states to identify new heuristic candidates. Basic parameter learning and adaptation laws are then applied to the heuristics online. This method extracts simple, but powerful heuristics that can approximate complex dynamics and account for errors stemming from model simplifications and parameter uncertainty without the loss of physical intuition while generalizing the parameter tuning process. Found model fits relating states, inputs, and commands are generalizable and robust to model inaccuracies and disturbances. By employing the framework, capabilities of the robot are improved due to the newly extracted heuristics without any modification to the controller structure or gains.

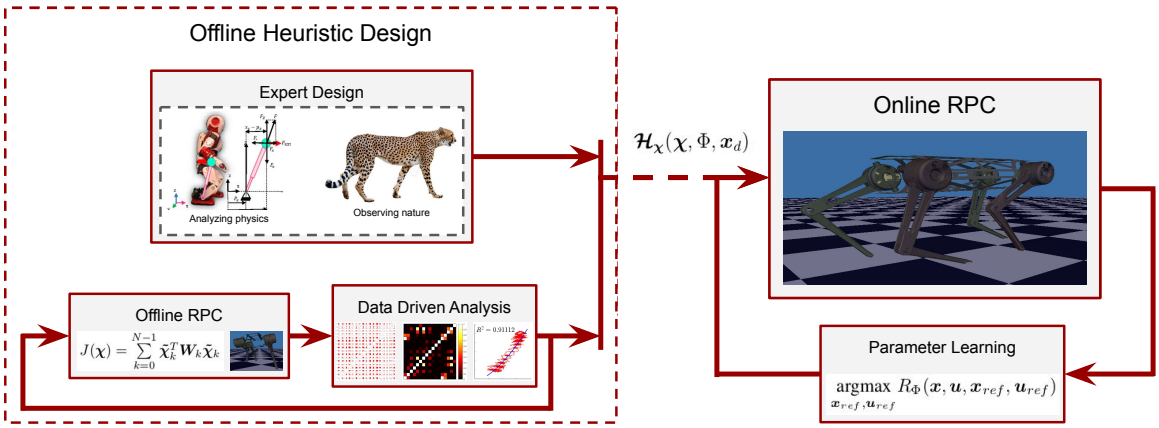


Figure 4-1: **Heuristic Extraction Framework.** Expert design and data analysis from running the RPC offline both yield heuristics that can be used to inform the controller online. Basic Reinforcement Learning adapts parameters to account for model inaccuracies and timing discretization delays that prove to be difficult to model.

4.1 Identifying Heuristic Candidates

The framework pairs expert design with data-driven analysis. Humans provide natural intuition that algorithms may not find as easily. Meanwhile, computers can analyze large amounts of data and recognize patterns automatically that may not be obvious to an expert. The framework seeks to exploit the advantages of each method to build the richness of the regularization heuristics that will be embedded in the robot for better online control. Expert design encompasses observing nature or analyzing simple physics. For example, the notion of vertical impulse scaling [59], or the widely used capture point [65]. These required experts in the field to make clever assumptions and simplifications for force generation and footstep placement in specific cases.

This extraction method provides a more autonomous method to quickly discover these relationships and is presented in [5]. This Chapter focuses on using the data gathered from simulations running the RPC offline in different scenarios. The states, controls, and inputs from the simulations are all set as potential functions of each other. Various simple model fits are applied to the data sets and highly correlated models are identified as candidates for a new heuristic policy. These candidates are further analyzed for repeatability and compared with the results of the predictive

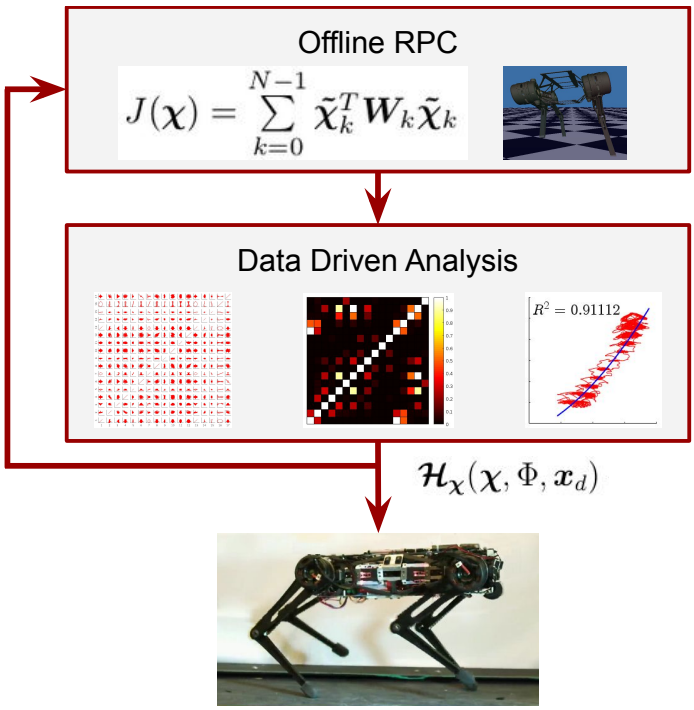


Figure 4-2: **Data-Driven Heuristic Extraction.** Methods presented in this paper improve the robot's capabilities by extracting heuristic models from simulated data.

optimization. We can then determine if the proposed model is a valid extracted heuristic. By embedding them back into the optimization as regularization, shown in Figure 4-2, the process can be repeated to explore new situations and find new models.

Since the solution times do not need to adhere to any real-time constraints offline, the cost space can be fully explored for nonintuitive local minima by providing various initial guesses on decision variables within the limits and tested for generality through validation experiments. Running the controller in a simulated environment also allows restriction of certain states to investigate the performance without some of the nonlinear coupling effects that would be present in the real system. The framework computes model fits on the robot data during simulation where forward, lateral, and turning rate commands are provided.

4.1.1 Data Collection Exploration

As with all data-driven algorithms, the results are highly dependent on the data collected through experiment or simulation. Designing meaningful explorations is just as important as the controller itself. In general, we are interested in heuristics that work well during steady state. Operating conditions of interest are isolated. For example, if the robot is not performing well during lateral locomotion, an exploration can be run for various lateral velocity commands offline with varying initial guesses to see in which conditions the robot's performance improves. Generalizable heuristics can also be found by combining explorations that may not specifically isolate an operating condition. If the heuristics found in isolation work in generalized cases without being detrimental to the performance, then it is considered a good heuristic.

Commands are given to the robot for forward, backward, and lateral velocities, as well as turning rates within a reasonable range set to be

$$\dot{p}_{x,d} = \Big[-2, \quad 2\Big] \frac{m}{s} \tag{4.1}$$

$$\dot{p}_{y,d} = \Big[-2, \quad 2\Big] \frac{m}{s} \tag{4.2}$$

$$\dot{\psi}_d = \Big[-2, \quad 2\Big] \frac{rad}{s}. \tag{4.3}$$

Without any heuristics, the controller is able to keep the robot upright at low velocities and turning rates. Since the interest is primarily to find steady state heuristics, it is important to design explorations in a way that commands the robot allowing it to reach steady state, but also provides enough varied commands within the range in order to find generalizable relationships.

The basic set of variables, $\boldsymbol{V}$, that are tracked during the exploration are composed of the robot CoM states, the footstep vectors, forces, desired commands, gait phases,

and various combinations of each that may be important physical quantities

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^T & \boldsymbol{\Theta}^T & \dot{\boldsymbol{p}}^T & \dot{\boldsymbol{\Theta}}^T \end{bmatrix}^T \tag{4.4}$$

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{r}_1^T & \boldsymbol{f}_1^T & \dots & \boldsymbol{r}_4^T & \boldsymbol{f}_4^T \end{bmatrix}^T \tag{4.5}$$

$$\boldsymbol{x}_d = \begin{bmatrix} \dot{p}_{x,d} & \dot{p}_{y,d} & \dot{\psi}_d \end{bmatrix}^T \tag{4.6}$$

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{x}^T & \boldsymbol{u}^T & \boldsymbol{x}_d^T & \boldsymbol{\Phi}^T & (\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}})^T & t & \dots \end{bmatrix}^T. \tag{4.7}$$

It is important to explicitly add certain combinations of states that may describe physical situations because the framework would not see these relationships unless requested. In the definition provided for the set of variables in (4.7), the term $\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}}$ is defined. This is a term that dominates the dynamics for turning at high speed which becomes important to add to $\boldsymbol{V}$ in order to find any heuristics that may be present only during high speed turning that are not in purely translational or rotational motion. This way, even though the models being fit to the data are simple in form, they can still find implicitly complex relationships. It does however require the designer to add them to the set of variables. The model fitting can be done offline as a post processing, meaning that it is separate from the data collection process and different sets of $\boldsymbol{V}$ can be used on the same data.

Figure 4-3 shows a small subset of the variable relationships of comparing the variable candidates in set $\boldsymbol{V}$. In actuality, all of the states in $\boldsymbol{V}$ are analyzed, but the plot was truncated to 17 variables for clarity. The entries where $i = j$ are ignored as they represent self-correlations between the variables.

4.1.2 Simple Intuitive Model Fitting

Since model fitting is done offline, it is easy to test for any model fit once the data has been collected. Any number of model types with any range of complexity can be tested. However, for this robot system, we are focused on fitting simple, intuitive models that are able to be easily tuned and possibly provide intuition for the heuristic. The fits tested for statistical correlation are polynomial and sum of sines fits where

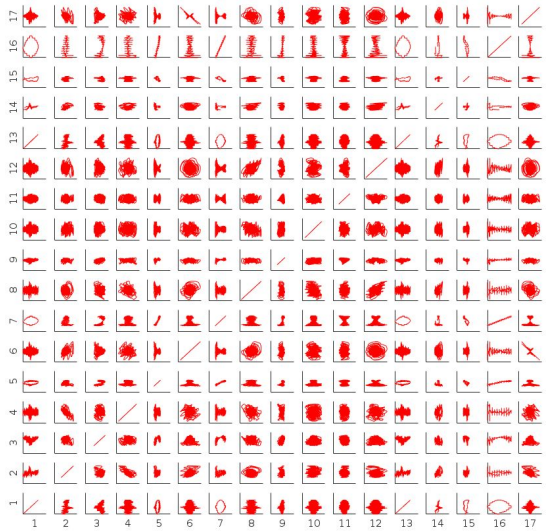


Figure 4-3: **Truncated Variable Relationships.** A subsection of the variable relationships of $\boldsymbol{V}$ from the data gathered from an exploration of varying translational forward velocity commands.

the model is described by

$$\text{Polynomial: } \mathcal{H}_{v_i}(v_j) = \sum_{n=0}^{D\in[0,9]} a_n^{v_i} v_j^n \tag{4.8}$$

$$\text{Sum of Sines: } \mathcal{H}_{v_i}(v_j) = \sum_{n=1}^{D\in[1,8]} b_n^{v_i} \sin\left(c_n^{v_i} v_j + d_n^{v_i}\right) \tag{4.9}$$

with D being the degree of polynomial and the number of summed sine terms respectively. A dependent variable, v_i, is written as a function of the independent variable, v_j. These models were deliberately chosen over more complicated highly nonlinear functions or neural networks because they are easily tunable and can provide intuition for the underlying physical relationships. It is straightforward to determine the effects of varying the parameters of a polynomial or sine wave while approximating highly nonlinear behaviors.

Statistical correlation is found through the the standard formula for goodness of

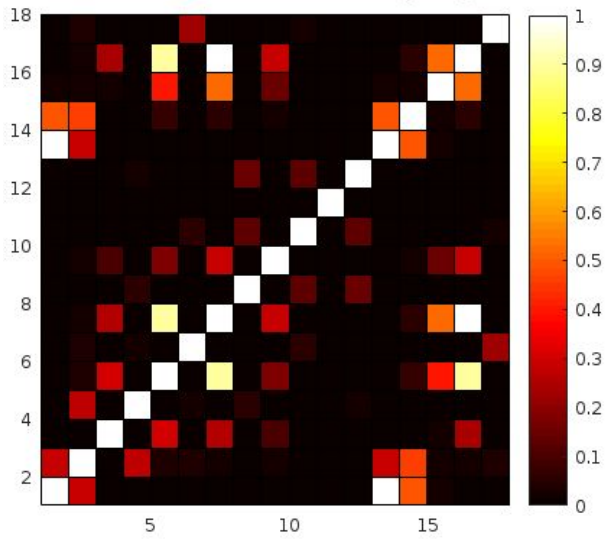


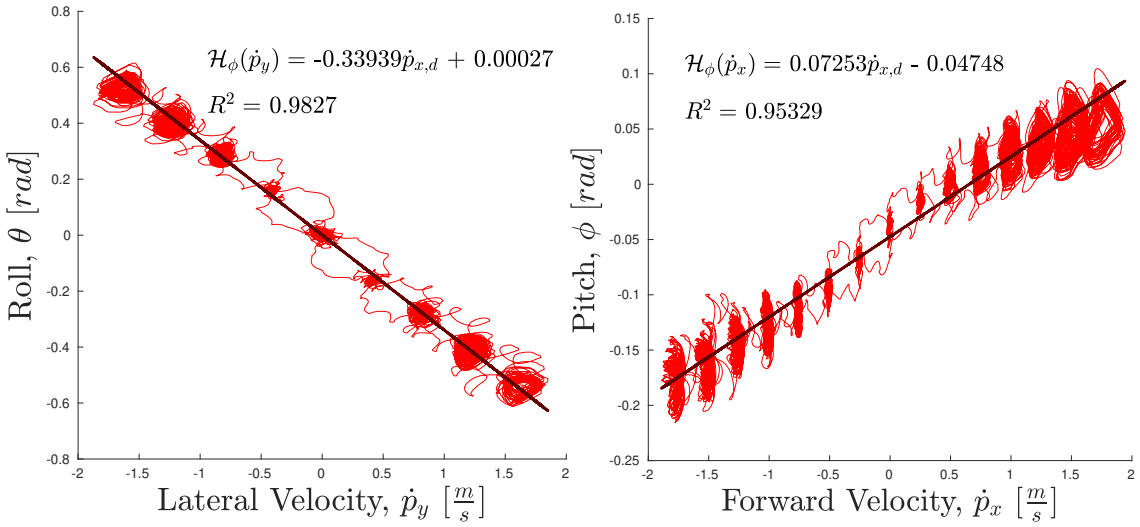
Figure 4-4: **Identifying Heuristic Model Candidates.** Model fits are computed for each of the variable relationships in 4-3 to identify new heuristic candidates. The higher R^2 values for a 1^{st} order polynomial fits signal relationships of interest.

fit between the model fit and the collected data as

$$R^2_{\mathcal{H}_{v_i}(v_j)} = \frac{\sum_{n=1}^{N} w_n \left(\hat{v}_{i,n} - \bar{v}_{i,n}\right)^2}{\sum_{n=1}^{N} w_n \left(v_{i,n} - \bar{v}_{i,n}\right)^2}. \tag{4.10}$$

Good candidates are identified by a high value for $R^2_{\mathcal{H}_{v_i}(v_j)}$, which states that it is statistically likely that v_i is dependent on v_j. The quality of the data used factors into the meaningfulness of the result.

Figure 4-4 shows an example result for the R^2 values for a first order polynomial model fit on data gathered from varying the commanded forward speed of the robot in Figure 4-3. Similar plots were made for the various model fits in (4.8) and (4.9). In the past, some of these relationships were discovered through experimental observation and parameters were hand tuned, but with this framework, the model types and associated parameters automatically discovered. This simplifies the process, while retaining intuition over the robot control.



(a) $\mathcal{H}_\theta(\dot{p}_y)$ (b) $\mathcal{H}_\phi(\dot{p}_x)$

Figure 4-5: **Linear Velocity Induced Orientation.** The framework identified the tendency towards linear orientation offset in pitch and roll with faster forward and lateral trotting.

4.2 Evaluating Heuristic Candidates

Finding a good correlation between variables does not mean the model found is a good heuristic. As stated, the quality of the heuristic is dependent on the exploration itself as well as the choice of set $\boldsymbol{V}$. Once candidates have been identified by the high $R^2_{\mathcal{H}_{v_i}(v_j)}$ value, they must be embedded back in the controller as a regularization heuristic and initial guess. They are then evaluated further in validation simulations or experiments.

Following an exploration where translational forward and lateral velocity commands are varied within the range, clear heuristic candidates emerged. Two of the stronger correlations found were linear dependencies of roll, θ, on the robot's lateral velocity, $\dot{p}_y$, and pitch, ϕ, based on the forward velocity, $\dot{p}_x$. The fits had an R^2 value of 0.9827 and 0.9533 respectively. These were found to be

$$\mathcal{H}_\theta(\dot{p}_y) = a_1^\theta \dot{p}_y + a_0^\theta = -0.339\dot{p}_y + 0.00027 \tag{4.11}$$

$$\mathcal{H}_\phi(\dot{p}_x) = a_1^\phi \dot{p}_x + a_0^\phi = 0.073\dot{p}_x - 0.0475 \tag{4.12}$$

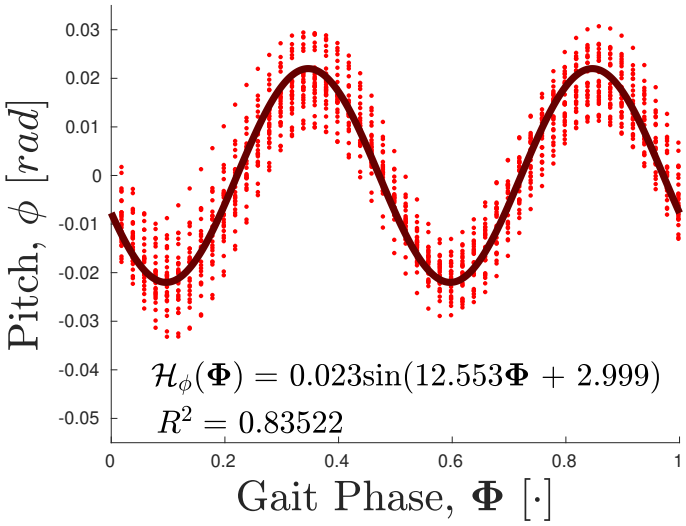


Figure 4-6: **Periodic Pitch Behavior.** The robot exhibits natural pitch dynamics that are be extracted from simulation data.

and shown in Figure 4-5 for both cases. The dominant linear nature of the relationships is clear, although this does not fully fit the data. In general systems approximated as linear are often powerful. Much of classical controls is centered around linearizing a system around an operating point of interest.

The same simulations were run again and analyzed after embedding the velocity-based orientation compensation heuristics, but this time new heuristics were discovered. Namely, a sinusoidal relationship for the periodic pitch behavior as a function of the gait phase with $R^2 = 0.835$ described by the new heuristic

$$\begin{aligned} \mathcal{H}_\phi(\mathbf{\Phi}) &= b_1^\phi \sin(c_1^\phi \mathbf{\Phi} + d_1^\phi) \\ &= 0.023 \sin(12.553\mathbf{\Phi} + 2.999). \end{aligned} \tag{4.13}$$

The results of which are depicted in Figure 4-6. Interestingly, this matches the intuition for a periodic pitch limit cycle during a trot gait where $c_1^\phi = 2\pi f$ and the two diagonal pairs of legs perform complementary actions over a single gait cycle, $f = 2$, giving $c_1^\phi \approx 12.566$.

By combining the two extracted heuristics for pitch dynamics in equations (4.12) and (4.13), we get a better heuristic to approximate the robot's gait phase and forward

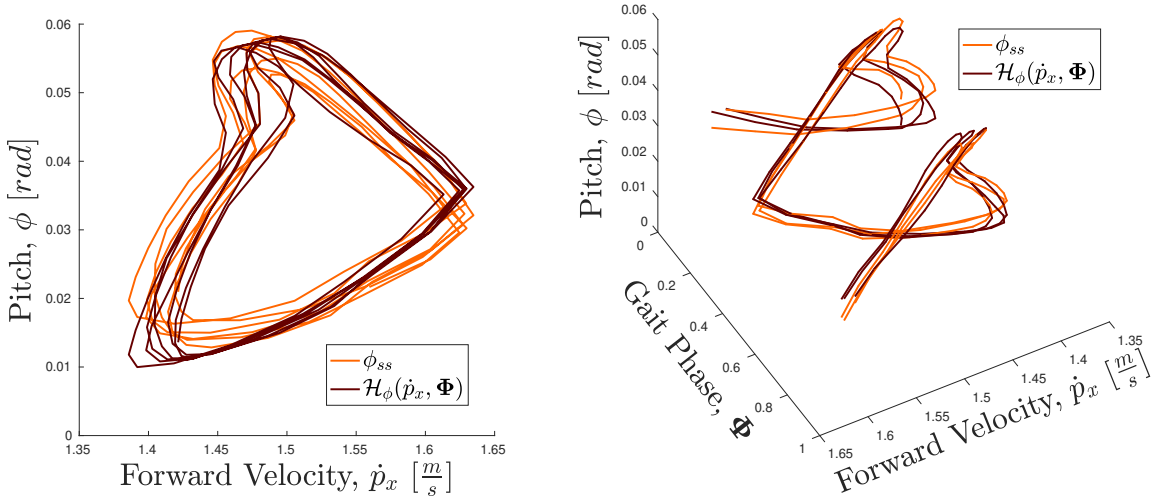


Figure 4-7: **Intuitive Complex Approximation.** The dark red shows the expected pitch during a steady state trot from the simple heuristics and matches closely to the actual pitch of the robot in orange from a commanded $1.5\frac{m}{s}$ forward speed. Although the robot's natural pitch dynamics seen here are complex, the heuristic approximates it well while also allowing easy manipulation of its shape.

velocity dependent pitch dynamics within the operating conditions as

$$\begin{aligned} \mathcal{H}_\phi(\dot{p}_x, \mathbf{\Phi}) &= \mathcal{H}_\phi(\dot{p}_x) + \mathcal{H}_\phi(\mathbf{\Phi}) \\ &= a_1^\phi \dot{p}_x + a_0^\phi + b_1^\phi \sin(c_1^\phi \mathbf{\Phi} + d_1^\phi). \end{aligned} \tag{4.14}$$

Using the heuristic, $\mathcal{H}_\phi(\dot{p}_x, \mathbf{\Phi})$, will inform the controller of the robot's natural motion. The robot's steady state pitch, ϕ_{ss}, in orange in Figure 4-7 is currently too difficult to analyze without data-driven methods. Although the commanded robot speed is $1.5\frac{m}{s}$ with no pitch, the robot's dynamics fluctuate in practice. However with heuristic (4.14) we can estimate the complex dynamics that take place naturally using only the simple intuitive models that are easy to modify. This ability to find combinations of simple models to approximate unknown, nonlinear functions while maintaining intuition over the parameters represents a key result of the heuristic extraction method.

The reason for choosing to describe complex dynamics as a sum of two simple heuristics becomes clear. We can break the parts of $\mathcal{H}_\phi(\dot{p}_x, \Phi)$ back down into the sum

of a linear equation and a sinusoidal function. Looking at the basic linear equation

$$\begin{aligned} y &= mx + b \\ a_1^\phi &= m \\ a_0^\phi &= b \end{aligned} \tag{4.15}$$

it is immediately obvious how the parameters affect the behavior. First parameter a_1^ϕ changes the slope and a_0^ϕ shifts the y intercept.

As with the linear case, the sinusoidal relationship is intuitive to tune. Standardly, the sine wave is written

$$\begin{aligned} y &= A\sin(\omega x + \phi_o) \\ b_1^\phi &= A \\ c_1^\phi &= \omega \\ d_1^\phi &= \phi_o \end{aligned} \tag{4.16}$$

where b_1^ϕ is the amplitude of the pitching during locomotion, c_1^ϕ is the frequency of the wave, and d_1^ϕ is the offset. This provides a clear advantage over methods that may approximate the dynamics better, but result in a model that is almost equally as difficult to understand as the dynamics themselves.

To verify that these heuristics are generalizable to other operating conditions, a validation test set was designed giving the robot cases that were not used during the extraction process. This is done to make sure situational heuristics don't interfere or degrade performance in other operating regions. The desired input boundaries were greatly increased with the added heuristics embedded in the optimization to be

$$\dot{p}_{x,d} = \Big[-3.5, \quad 3.5\Big] \frac{m}{s} \tag{4.17}$$

$$\dot{p}_{y,d} = \Big[-3.5, \quad 3.5\Big] \frac{m}{s} \tag{4.18}$$

$$\dot{\psi}_d = \Big[-4.5, \quad 4.5\Big] \frac{rad}{s} \tag{4.19}$$

If the heuristics hold high correlations over the new validation simulations, then they are deemed to be generalizable, rather than just good fits for the specific exploration data. The control designer can then choose to add them into the controller as regularization heuristics or not. A key aspect of using this method is that it generates heuristic candidates, but does not relinquish control over what the optimization is attempting to regularize to. It is a tool to help design heuristics which are intuitive to the expert designer, rather than force unknown functions to shape the cost function.

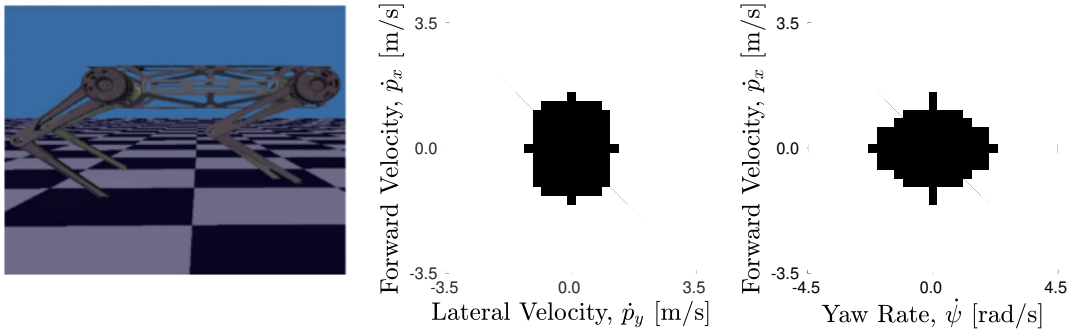
4.3 Performance Improvements

Along with the examples presented in Section 4.2, many other heuristics were found which greatly improved performance of the robot. The extraction method was run several times for different common locomotion cases. We can clearly see the effects of each of the heuristics as they are incrementally added. Heuristics were found with the limited command set (4.3), while, an exhaustive exploration of the commands outlined in (4.19) was used to validate the quality of the heuristic and map the robot's viable operating region. This region simply states that the robot is able to trot with the given commands for several steps without falling, but does not qualify the quality of the locomotion. This qualitative assessment of how "nice" the movements are has no clear description metric so the choice was made to only rate whether the robot falls or not.

Starting with the base case of stable locomotion where the feet were naïvely regularized to be placed under the ground projection of the hips in the world frame

$$\boldsymbol{\mathcal{H}_r}(\boldsymbol{\Theta}) = \boldsymbol{P}^T \boldsymbol{P}(\boldsymbol{R}(\boldsymbol{\Theta})^{\mathcal{B}}\boldsymbol{r}_h), \tag{4.20}$$

the robot is able to stably stand and take a few steps forward, as well as turn in place with low turning rates. However, as soon as the robot begins to move with speed, the leg placement is not adequate and although the optimization attempts to find a better foot placement, it is unable to due to regularizing for a heuristic that is inappropriate



(a) No Heuristics (b) Small Viable Region

Figure 4-8: **No Heuristics.** The base case where no situational heuristics have been added has the footstep placement regularized to be under the hip.

for the situation. Figure 4-8 shows the viable operating region where the robot was able to trot without falling over. Most of the operating region is centered around the stationary position where the leg position relative to the CoM over the stance period does not change too much. This is a case where the regularization is detrimental to the performance of the robot so a new heuristic must be found to shape the cost function favorably in these situations.

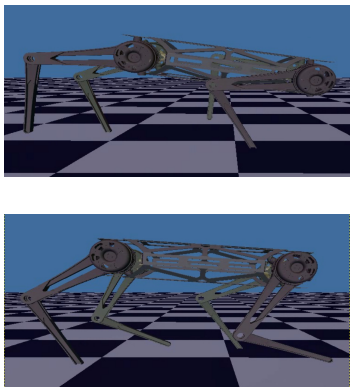
Using the method with an exploration of purely translational velocity commands in the forward, backward, and lateral directions. A linear model fit between the CoM velocities and the footstep locations was found to be highly correlated. A heuristic was found for foot placement in both motion along the x axis, $\mathcal{H}_{r_x}(\dot{p}_x)$, and the lateral y axis, $\mathcal{H}_{r_y}(\dot{p}_y)$, which were then combined to make the foot placement heuristic, $\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}})$, defined in the vector form as

$$\mathcal{H}_{r_x}(\dot{p}_x) = a_1^{r_x}\dot{p}_x + a_0^{r_x} \tag{4.21}$$

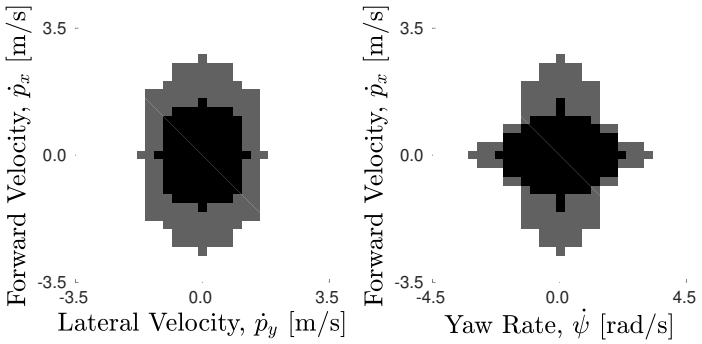
$$\mathcal{H}_{r_y}(\dot{p}_y) = a_1^{r_y}\dot{p}_y + a_0^{r_y} \tag{4.22}$$

$$\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}}) = \boldsymbol{a}_1^{\boldsymbol{r}}\dot{\boldsymbol{p}} + \boldsymbol{a}_0^{\boldsymbol{r}} \tag{4.23}$$

This embeds the common sense knowledge that steps should be taken in the direction of the translational velocity. Figure 4-9 shows the robot moving forward prior to the heuristic addition, as well as after. Before, it took steps directly under the hips which



(a) $\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}})$



(b) Extends Translational Velocity Region

Figure 4-9: **Translational Stepping.** While translating with velocity, footsteps are placed in the direction of the velocity so as to maximize the utility of the contact feet over the stance period. The lighter gray signifies newly achievable operating regions.

meant that as it traveled forwards, they would lag behind and the forces would be biased to making the robot tilt down in the direction of motion, eventually falling as feet reached their kinematic limits during stance. After the translational stepping heuristic was found and added, the robot took steps in along the velocity vector, giving it twice as much stance time capabilities and resulting in even forces on the body as feet were approximately averaged to be under the hips rather than behind them as with the previous case. Maximum velocity capabilities were dramatically increased.

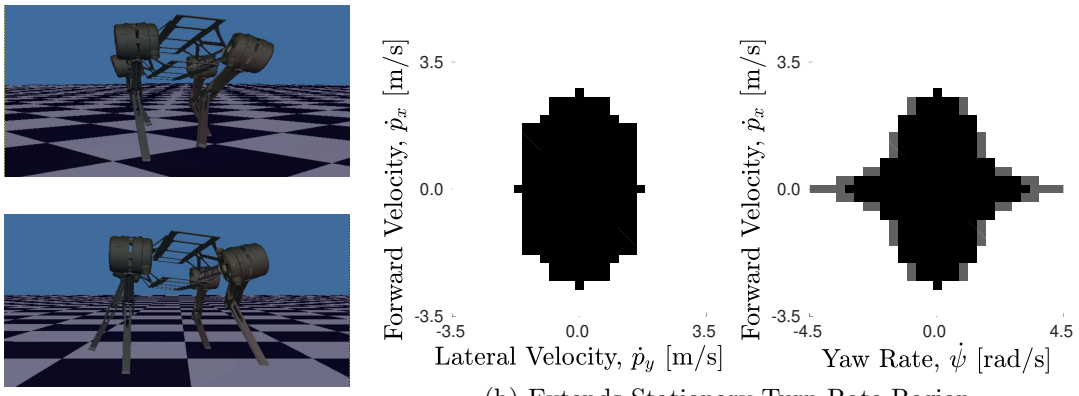
Similarly, the robot had issues when turning with large yaw rate commands as footsteps were suggested to be under the hips if there was no translational velocity. An exploration was run with purely rotational velocity commands at various values between the bounds. Again, a linear relationship was found for the footstep placements

$$\mathcal{H}_{r_x}(\psi) = a_1^{r_x}\psi + a_0^{r_x} \tag{4.24}$$

$$\mathcal{H}_{r_y}(\psi) = a_1^{r_y}\psi + a_0^{r_y} \tag{4.25}$$

$$\boldsymbol{\mathcal{H}_r}(\psi) = \boldsymbol{a}_1^{\boldsymbol{r}}\psi + \boldsymbol{a}_0^{\boldsymbol{r}}. \tag{4.26}$$

As the robot spins faster, the feet are placed along the arc of the turn in the direction



(a) $\boldsymbol{\mathcal{H}_r}(\psi)$ (b) Extends Stationary Turn Rate Region

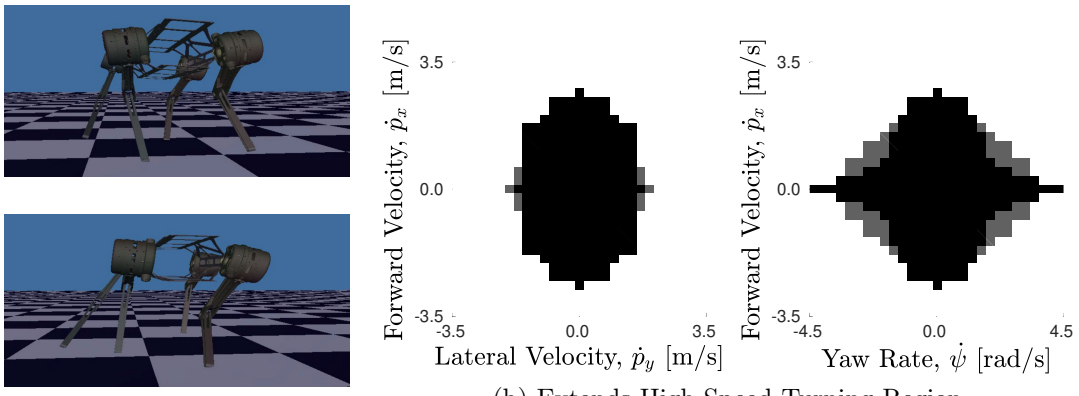
Figure 4-10: **In-Place Turning.** While turning in place, the footsteps tend to be placed in the direction of the turn in a similar manner as the translational velocity stepping case.

of the angular velocity. Figure 4-10 shows analogous behavior to the translational case for the rotations. The robot is given a positive turn rate command and the legs tended to lag behind the hips until they reached the end of the workspace. After the heuristic as regularized to, the foot placements would lead the hips during spinning and gain more workspace to allow faster in-place turning.

The robot still lacked the ability to both move forward and turn at the same time. During high speed turns, the robot tended to fall towards outwards along the turning radius as if it were an object slipping off a spinning plate. The outer leading foot approached the CoM projection and often exceeded its workspace or slipped and the body would tip over as no contact foot could stabilize it. It became clear that the footstep location heuristics up to this point were not adequate for these situations Regular locomotion carries the assumption in this case that there is no negligent vertical velocity, as well as roll and pitch angular velocities. This signifies that the characteristic high speed turning cross product can be simply approximated

$$\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}} \approx \begin{bmatrix} \dot{p}_y\dot{\psi} & -\dot{p}_x\dot{\psi} & 0 \end{bmatrix}^T. \tag{4.27}$$

The elements of the resultant vector are added to the set of variables to test for



(a) $\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}})$

(b) Extends High Speed Turning Region

Figure 4-11: **High Speed Turning.** High speed turning is dominated by the combination of translational and places steps radially outwards along the turning radius.

heuristic candidates.

$$\mathcal{H}_{r_x}(\dot{p}_y\dot{\psi}) = a_1^{r_x}(\dot{p}_y\dot{\psi}) + a_0^{r_x} \tag{4.28}$$

$$\mathcal{H}_{r_y}(-\dot{p}_x\dot{\psi}) = a_1^{r_y}(-\dot{p}_x\dot{\psi}) + a_0^{r_y} \tag{4.29}$$

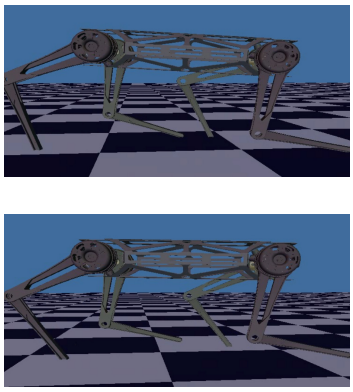
$$\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}}) = \boldsymbol{a}_1^{\boldsymbol{r}}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}}) + \boldsymbol{a}_0^{\boldsymbol{r}} \tag{4.30}$$

In Figure 4-11 we can see an example of a high speed turn where, from the perspective of the reader, the robot is trotting outwards from the page while simultaneously turning to the right with a positive turning rate. The footsteps are biased outwards along the turning radius.

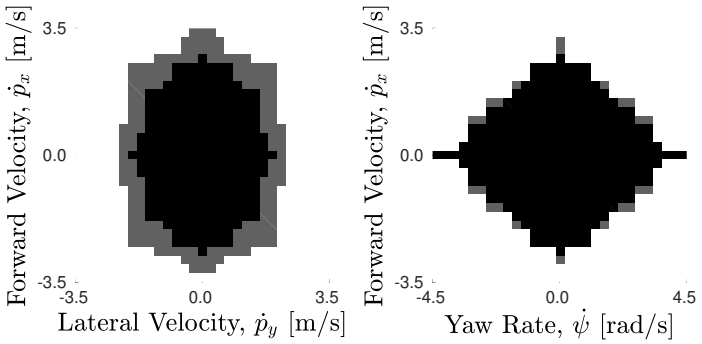
Curious as to why the controller was choosing to place the footsteps towards the outside of the turn, further analysis was done on the system. It turns out that there is likely a physical reason for this. The robot is choosing footsteps that roughly correspond with the angle created by the resultant of the force compensating gravity and the force causing the centripetal acceleration. The lateral deviation can be found from simple analysis of circular motion as

$$\boldsymbol{r} = \frac{p_z}{||\boldsymbol{g}||}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}}). \tag{4.31}$$

This follows what we would observe from looking at animals in nature as they too



(a) $\boldsymbol{\mathcal{H}}_{\boldsymbol{\Theta}}(\dot{\boldsymbol{p}}, \boldsymbol{\Phi})$



(b) Generally Extends Viable Region

Figure 4-12: **Orientation Compensation.** The body of the robot is more parallel to the ground with the orientation heuristics in place and trots in a much more controlled manner.

throw their legs out sideways when turning at high speeds. Finding this heuristic through analytical methods was possible, but had not been considered previously. With the extraction method, it was emergent and once found, it became rather obvious. This highlights a feature of using simple heuristics that can be intuitively analyzed.

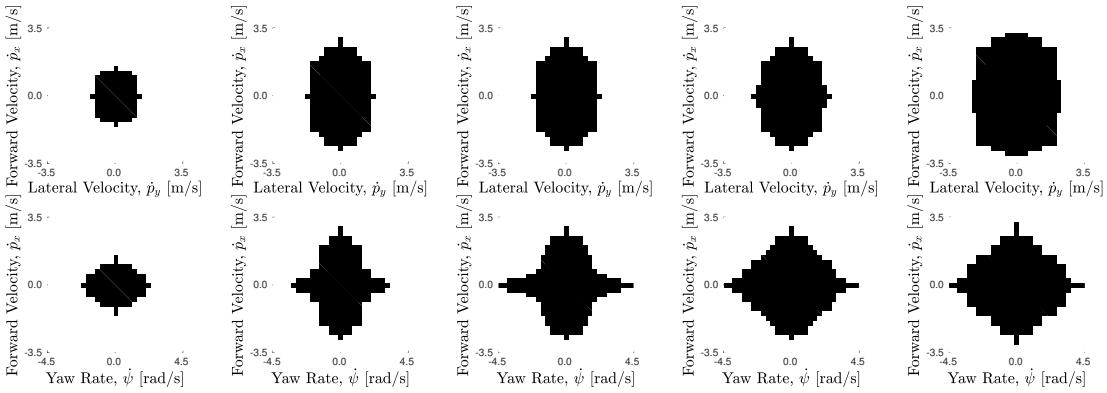
Finally, the velocity induced orientation compensation heuristics for roll and pitch and the periodic orientation cycles from the previous section were implemented. The full heuristic in vector form is

$$\boldsymbol{\mathcal{H}}_{\boldsymbol{\Theta}}(\dot{\boldsymbol{p}}, \boldsymbol{\Phi}) = \boldsymbol{a}_1^{\boldsymbol{\Theta}} \dot{\boldsymbol{p}} + \boldsymbol{a}_0^{\boldsymbol{\Theta}} + \boldsymbol{b}_1^{\boldsymbol{\Theta}} \sin(\boldsymbol{c}_1^{\boldsymbol{\Theta}} \boldsymbol{\Phi} + \boldsymbol{d}_1^{\boldsymbol{\Theta}}). \tag{4.32}$$

A likely reason for the improvement is that at high speeds the orientation is closer to flat, meaning that the simplifications made for the orientation dynamics are more appropriate for the actual robot orientation. Also, at high speeds, the robot pitching will lower the height of the front hips and reduce the workspace of the front legs so they may not have the space needed to swing forward without colliding with the motors. The pitch will raise the back hips and increase the workspace, but reduce the area reachable during stance. The back legs will approach singularity and eventually exceed the workspace while still in stance at high speeds.

Table 4.1: Extracted Heuristics

Heuristic	Extracted Heuristic Model
Translational Stepping	$\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}}) = \boldsymbol{a}_1^{\boldsymbol{r}}\dot{\boldsymbol{p}} + \boldsymbol{a}_0^{\boldsymbol{r}}$
In-Place Turning	$\boldsymbol{\mathcal{H}_r}(\dot{\psi}) = \boldsymbol{a}_1^{\boldsymbol{r}}\dot{\psi} + \boldsymbol{a}_0^{\boldsymbol{r}}$
High Speed Turning	$\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}}) = \boldsymbol{a}_1^{\boldsymbol{r}}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}}) + \boldsymbol{a}_0^{\boldsymbol{r}}$
Orientation Compensation	$\boldsymbol{\mathcal{H}_\Theta}(\dot{\boldsymbol{p}}) = \boldsymbol{a}_1^{\boldsymbol{\Theta}}\dot{\boldsymbol{p}} + \boldsymbol{a}_0^{\boldsymbol{\Theta}}$
Periodic Orientation	$\boldsymbol{\mathcal{H}_\Theta}(\boldsymbol{\Phi}) = \boldsymbol{b}_1^{\boldsymbol{\Theta}} \sin(\boldsymbol{c}_1^{\boldsymbol{\Theta}}\boldsymbol{\Phi} + \boldsymbol{d}_1^{\boldsymbol{\Theta}})$
Height Compensation	$\mathcal{H}_z(\dot{p}_x) = a_2^z\dot{p}_x^2 + a_1^z\dot{p}_x + a_0^z$



(a) No Heuristics (b) $\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}})$ (c) $\boldsymbol{\mathcal{H}_r}(\psi)$ (d) $\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}})$ (e) $\boldsymbol{\mathcal{H}_\Theta}(\dot{\boldsymbol{p}}, \Phi)$

Figure 4-13: **Viable Operating Regions.** Each plot from 4-13a to 4-13e adds one of the heuristics discussed above. As new heuristics are discovered and subsequently injected into the optimization, the set of viable operating region where the robot is in stable locomotion increases. The dark areas signify where a fall did not occur, but does not qualify the movements.

The newly extracted heuristics are laid out in Table 4.1. The periodic orientation and a height compensation heuristic were also added. While they did not directly expand the viable operating region, they made the resulting motion of the robot smoother and more consistent, as well as marginally reducing the computation solve time. As stated, the viable regions were only measures of the robot's ability to walk upright and continue taking steps, but does not qualify the quality of how the movements look. The relative improvement when adding each heuristic is shown in Figure 4-13 as they are subsequently added to the optimization. Initially the robot had maximum viable operating conditions of $|\dot{p}_{x,max}| = 1.6\frac{m}{s}$, $|\dot{p}_{y,max}| = 1.4\frac{m}{s}$ and $|\dot{\psi}_{max}| = 1.8\frac{rad}{s}$ which were increased to $|\dot{p}_{x,max}| = 3.5\frac{m}{s}$, $|\dot{p}_{y,max}| = 2.4\frac{m}{s}$ and $|\dot{\psi}_{max}| = 4.5\frac{rad}{s}$ using the extracted heuristics. From the original naïve case to the

fully informed case, this represents a performance improvement of about four times the operating region area.

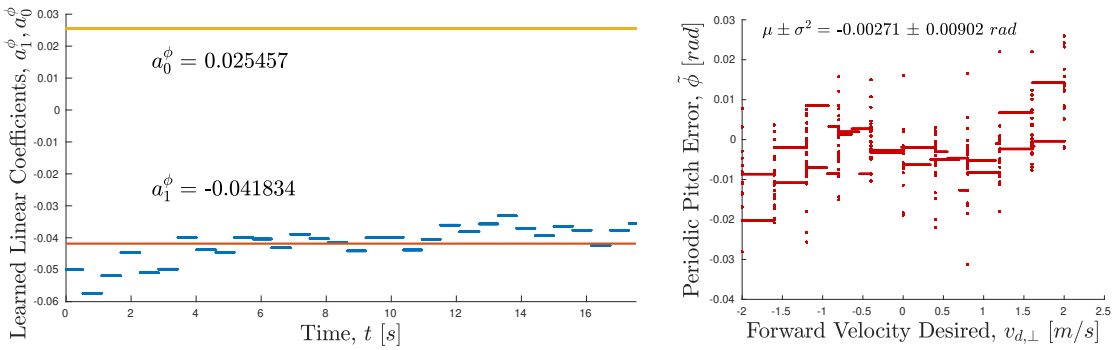
It is worth noting that the heuristics described are left parametrized rather than giving exact values intentionally. The actual values are dependent on the values chosen for decision variable gains, gait frequencies, optimization solve times, and many other factors. However, the model for each heuristic is generalizable to various combinations of these operator selected parameters. Therefore, it is straightforward to manually tune or rerun the extraction process for a different set. It is also simple to expand the variables selected for statistical comparison to find new or combined heuristics. This dissertation is meant to provide a framework for extracting heuristics and presenting important examples implemented on the Cheetah robots rather than finding all possible heuristics.

4.4 Model Parameter Adaptation Learning Laws

Since the data was gathered in simulation without real-time computation restrictions, the parameters used for the heuristics may often differ on the actual system or even on the same simulated model with real-time restrictions in place. Therefore, in order to reduce the labor on the control designer, simple adaptation learning laws are formulated to automatically adapt the model parameters during operation. A reward function for linear heuristics is written to sum the instantaneous errors between the desired and actual states over a full gait cycle and rewards smaller deviations written

$$\mathcal{R}_{\mathcal{H}_{v_i}} = -\left\| \sum_{\Phi} (v_d - v_i) \right\|. \tag{4.33}$$

The reward is evaluated cumulatively over the gait cycle, because the instantaneous error is often canceled out in a complementary portion of the gait cycle so adapting parameters instantaneously would likely cause instability and divergence. The original higher level desired state input is used instead the regularization term because ultimately we want the robot to track the higher level commands achieved through



(a) Parameter Adaptation (b) Learned Error

Figure 4-14: **Learned Pitch Compensation Parameters.** Both parameters a_0^ϕ and a_1^ϕ are learned online to satisfy the reward function and averaged to be constants in the robot's memory. The error during locomotion is virtually eliminated.

the heuristics rather than the heuristics themselves.

For now, a learning law has been implemented for the linear model case by adapting the constant term when the independent variable is under a certain small threshold and the linear parameter when the independent variable is over it. The constant term adapts with stp sizes corresponding to the reward function gradient, while the linear term adapts by taking step sizes with the reward gradient normalized with the independent variable

$$\begin{cases} a_{0,k}^{v_i} = a_{0,k-1}^{v_i} - \nabla \mathcal{R}_{\mathcal{H}_{v_i}} & , |v_j| < v_{j,min} \\ a_{1,k}^{v_i} = a_{1,k-1}^{v_i} - \frac{\nabla \mathcal{R}_{\mathcal{H}_{v_i}}}{v_j} & , |v_j| \geq v_{j,min}. \end{cases} \tag{4.34}$$

In practice, this acts similar to an integral control term where an error summed over the gait cycle in one direction will push the parameter to negate any constant offsets. However, a level of sophistication is introduced by using the gradient of the reward function to make sure step sizes are taken appropriately. Large errors will take larger steps in the parameter adaptation to ensure faster convergence and small errors will have smaller steps to prevent windup or overcompensation.

The learning law is used on the linear parameters for the velocity induced pitch error compensation heuristic, $\mathcal{H}_\phi(\dot{p}_x)$, where the model used for extraction is different from the actual robot where the heuristic is implemented. In this case, the model

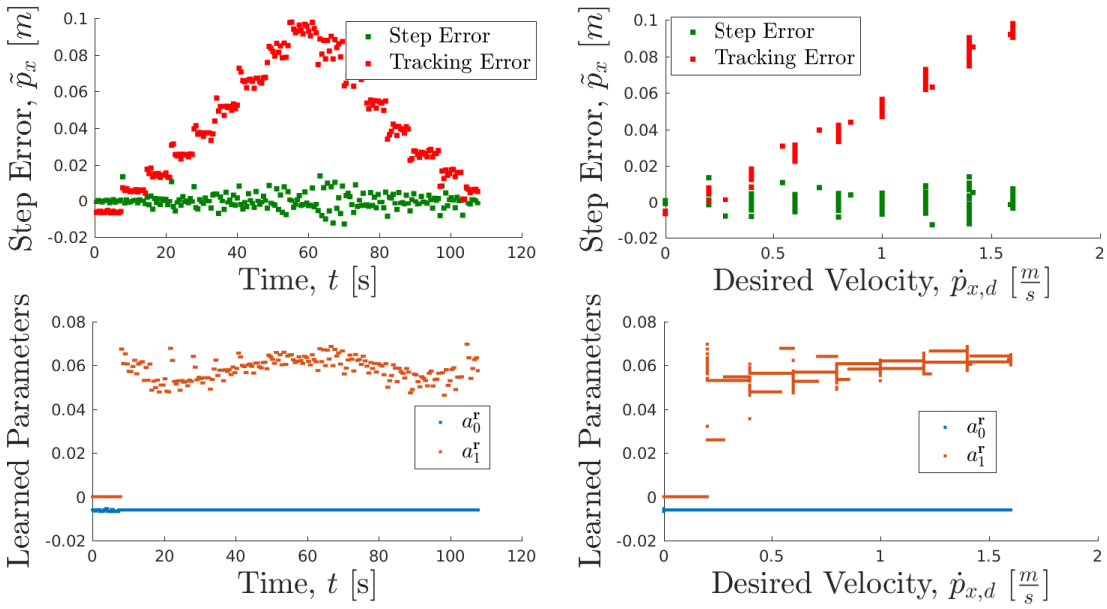


Figure 4-15: **Step Error Parameter Compensation.** Using the parameter compensation, the robot is able to automatically adapt its nominal foot placement to account for poor swing leg tracking. Despite the tracking error, the robot is able to accurately place the feet at the locations calculated by the controller.

used for extraction has different viscous friction in the motors meaning that the actual robot may not need as much compensation while moving. As seen in Figure 4-14, the robot walks forward at various speeds over several gait cycles learning a new a_1^ϕ parameter corresponding to less needed compensation heuristic. With the learned parameters, the robot's pitch error over a trial varying the forward velocity between $-2\frac{m}{s}$ and $2\frac{m}{s}$ is reduced to $-0.0027 \pm 0.0090 rad$, which means that the adapted parameters have virtually eliminated the error.

Similarly, the same adaptive learning law was implemented for the translational stepping heuristic. Swing leg tracking has been a problem plaguing the Cheetah 3 hardware. The cause is not well known, but the problem is apparent. Figure 4-15 shows the footstep placement adaptation for a variety of commanded robot velocities. The foot placement heuristic is originally seeded with parameters of $a_0^r = 0$ and $a_1^r = 0$, but due to the learning law chosen using the derivative of the reward function, it adapts the parameters within the first step they are activated for learning. The tracking error for the robot grows as the swing leg is not able to reach the commanded

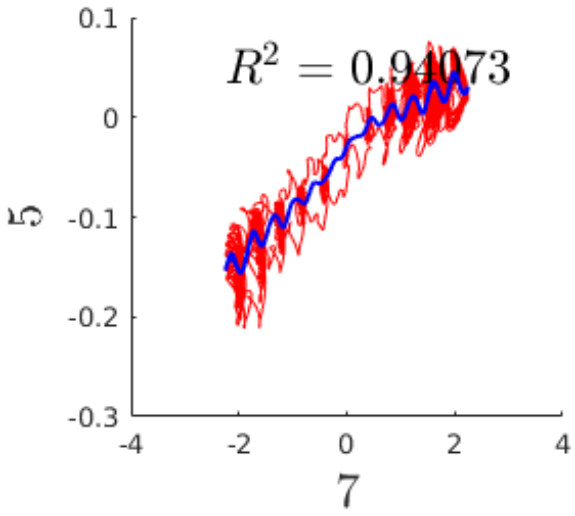


Figure 4-16: **Overfitting.** A highly correlated polynomial model with $D = 9$ was found between the robot's pitch, ϕ, and forward velocity, $\dot{p}_x$, although it likely does not hold as strongly for a different experiment dataset.

location, however, the actual step location error is small and does not vary over the commanded forward velocities. The step location error is the difference between the position of the foot at touchdown and the originally desired placement calculated in the controller, which is the important location. This signifies the adaptation law's ability to compensate for tracking control errors as well.

4.5 Preventing Overfitting and False Dependencies

As with all data-driven methods, we must take care to prevent overfitting the data and finding falsely dependent relationships. This is highly based on the exploration chosen for gathering the data, as well as the variables chosen for analysis. An example of each is depicted in this section to show the dangers of blindly trusting data-driven methods, particularly when being deployed on well understood systems. Figure 4-16 shows a higher order model being fit to the same pitch to forward velocity relationship, $\mathcal{H}_\phi(\dot{p}_x)$. This model has a similar correlation of $R^2 = 0.941$ compared to the linear relationship with $R_2 = 0.953$. However, when validating on other data sets with different explorations, the linear model remained with a high correlation while the higher order model dramatically decreased in correlation as it was likely overfit to the

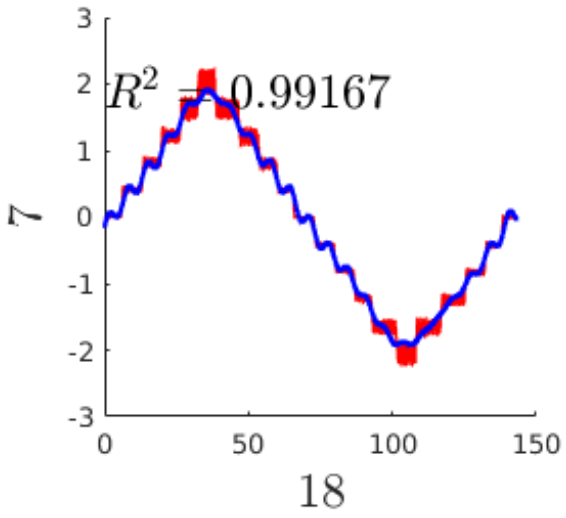


Figure 4-17: **False Dependency.** A sum of sines model with $D = 8$ between forward velocity, $\dot{p}_x$, and experiment time, t, was found to be highly correlated. This is due to the systematic exploration choice when in reality they are in no way correlated.

data. This is also precisely why the simpler models were chosen over higher order fits with potentially slightly higher correlations.

In the case of Figure 4-17, a false dependency is found between the robot's forward velocity and time. A model with 8 summed sine waves is found that has a very high correlation of $R^2 = 0.99$ between the two. We know through common sense that there is absolutely no definitive correlation between a robot's forward velocity and an arbitrary time variable as the robot could be doing anything at any given time on any given experiment. However, this relationship was found during an exploration that had the robot systematically increase the forward velocity command by $0.4\frac{m}{s}$ every $7s$ until it reached $2\frac{m}{s}$ and then reduce by the same amount at the same time interval. This creates the staircase pattern seen. This extracted heuristic is not appropriate for a robot which can be freely commanded any velocity at any time. In validation testing, the heuristic was found to have no correlation as a different pattern of commands was given and therefore discarded. This highlights the importance of both validating heuristic candidates, as well as having an expert designer overlooking the process to avoid reliance on pure data-driven experiments which are prone to poor design. By simply looking at the candidate, the false dependency would have been

obvious without even needing the validation data. This framework is not a blackbox and while it can help find unknown heuristics, intuition should not be ignored.

4.6 Comparison with Other Methods

It is well known that plenty of methods exist for using data to extract information for legged locomotion control from simulation or experiment data [70]. Designing meaningful cost functions for legged locomotion through automated parameter tuning [71], as well as through inferring cost weights from data-driven inverse optimal control [69] is an area that has been explored, but proven difficult due to the complexity and hybrid nature of discrete contact breaks. These methods require expert input to create a controller that can already perform locomotion and or choosing the correct structure of the cost function. Many techniques to approximate nonlinear functions have been used. By iteratively linearizing the dynamics, standard control theory can be used to then control the complex legged system [78]. Others attempt to reduce the dimensionality of nonlinear systems to approximate complex dynamics or identify the underlying equations of motion through a combination of data-driven and model-based methods [29, 80]. Identification of human locomotion was carried out using Gaussian function approximations with smartphones in [18]. All of these techniques rely on knowledgeable input from an expert designer similar to the method presented.

Recently, there have been many advancements in the field of Machine Learning. The use of deep neural networks to develop policy networks for robot control attempts to identify the dynamics of a robot and figure out the best way to use the actuators for control [37, 79, 34, 87]. While, learning to approximate nonlinear systems has been successful from a relatively small amount of data [88], in general the learning process requires large amounts of training data. Deep Learning is powerful and well-suited for problems where there is little to no knowledge of the system and where there no clear path to a partial or full solution. In a well studied and understood field such as legged robot locomotion, it is not necessary to completely leave the problem up to a learned neural network. A main drawback of neural nets is that it becomes

intractable to manually tune or modify them. If the policy network is proficient at control of the system in situations similar to the training data, it is not guaranteed to perform well in an unseen situation and it is usually not possible to simply tune a gain or change a parameter, so the network must be completely retrained.

The method presented is designed specifically to find simple models that maintain the possibility of gaining physical intuition and easily tune or modify. With the simple heuristics extracted through this method, the operator maintains intuition for the parameters in the models and retains control over the behavior of the system. Heuristics are extracted quickly, with small amounts of data. The velocity-induced orientation compensation heuristics were found with less than 3 minutes of locomotion data. However, the need to explicitly define candidate variables in order to find relationships requires having an intuition for what states to consider. For example, the high speed turning foot placement heuristic would not have been extracted without adding the $(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}})$ term to $\boldsymbol{V}$. Sets of heuristics that are found iteratively by building on previous heuristics, which means that a large number of entirely different heuristic sets likely exist depending on the explorations chosen and the order at which they are conducted. While other methods may automatically find these at once through a complex neural network training or a multivariable nonlinear regression, the operator may not have intuition for which situations they are activated in or how to modify them if performance is poor.

In the case of overfitting or false dependencies, these are easy to spot when simple models are used, but may be much more difficult to detect when using a large neural network where each connection may not be fully understood. It is often seen as a negative for a data-driven method to require expert supervision and post-analysis. However, this method encourages the intervention and addition of human knowledge. For a well understood problem such as legged locomotion, discarding the input of an expert is detrimental and counterproductive. With the goal of dynamic legged locomotion in mind, taking advantage of the many years of prior research, while allowing data-driven methods to further the boundaries of our knowledge is a synergy that has proven itself to yield impressive results.

Chapter 5

Implementing Real-Time Nonlinear Optimization

The theory behind the controller has been shown in simulation, but robotics requires engineering to actually implement. Nonlinear optimization for real-time legged locomotion control in particular is one of the techniques that has shown promise, but falls short when implemented on hardware systems subjected to computation limits and sensitive to undesirable local minima. For reference, Figure 5-1 shows the major milestones taken to successfully take the controller from the theoretical derivation of the framework to a robust dynamic implementation of RPC. The implementation details described in this section were important in realizing the theory. Getting the controller successfully running on the full dynamics simulation required further work to be done for hardware implementation to mitigate errors stemming from timing constraints.

While in Figure 3-1 the overall block diagram is presented with a single block representing the theoretically derived RPC, in practice it is not so simple to just run the optimization assuming that it will be solved instantaneously. Therefore work must be done to both ensure the NLP is setup for favorable solutions to be returned, as well as strategies for dealing with cases where the solution is delayed or incorrect. Figure 5-2 extends on the theoretical derivation and shows the algorithms and modifications designed to aide the optimization overcome the real-time implementation challenges.

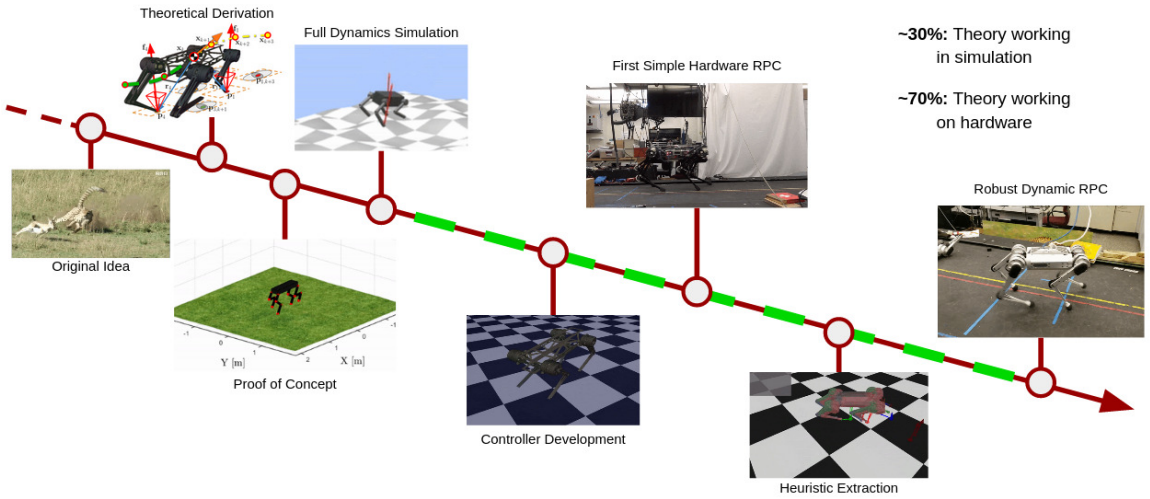


Figure 5-1: **Towards Robust Dynamic RPC.** The implementation details were crucial in the success of the controller. While the theoretical proof of concept and full dynamics implementation were straight forward, a majority of the time developing the framework was comprised of getting the theory implemented on real world hardware depicted by the dashed green line.

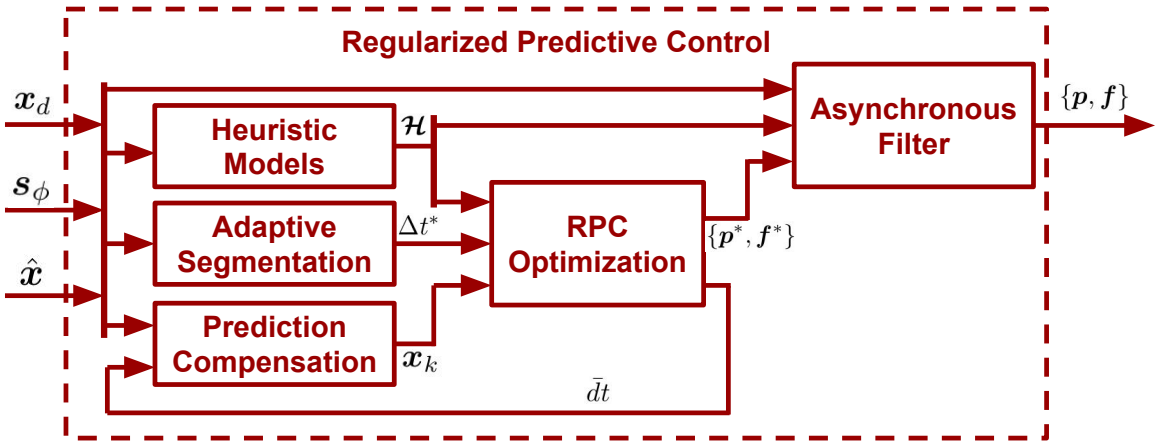


Figure 5-2: **Modified RPC Implementation.** The theoretical RPC optimization is aided by various other algorithms to overcome some of the limitations and difficulties that come with real world implementation.

This section describes the blocks in the diagram and their function in the control framework resulting in successful locomotion on the hardware platform.

5.1 Adaptive Timing Segmentation

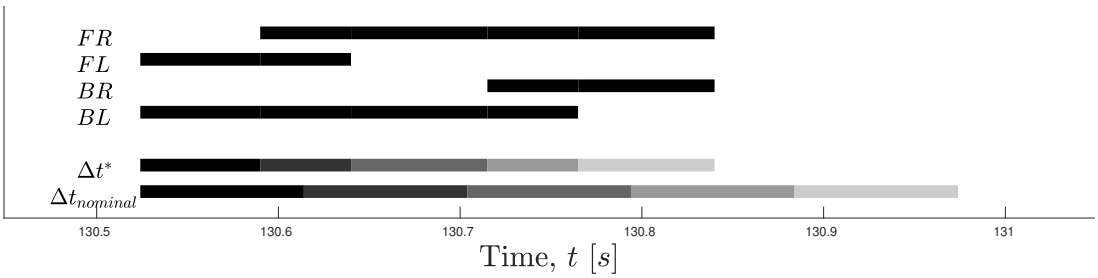
Due to the defined discrete changes between contact and non-contact states of each of the legs, the optimization timestep segmentations must take into account the discrete modes. As the controller is run continuously rather than at a scheduled time, the time step vector to be optimized over cannot simply be prescribed uniformly for every gait at every time. As opposed to continuous systems, which can have a set lookahead prediciton horizon, legged systems should have varying discretizations and horizon lengths. As such, an algorithm must be developed to adaptively segment the time vector regardless of the scheduled gait pattern.

The algorithm takes a nominal discretization timestep vector and attempts to calculate the closest set of timestep segmentations, while respecting the gait schedule determined outside the controller. Discrete scheduled contact switches must signify a new time division as the robot enters a different control mode. Since this often happens irregularly, the algorithm chooses the number of timesteps within the same contact mode and spaces them out evenly. Algorithm 1 outlines this process. This allows for an incredible amount of freedom in choosing gait patterns and frequencies. Instead of constraining the predicted timestep vector to be divided evenly, any gait schedule can be optimized over without needing to make any changes within the optimization.

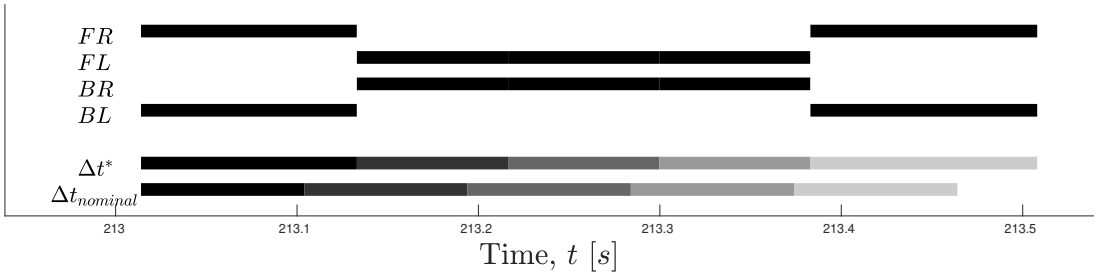
The ambling gait in Figure 5-3a has 8 quick contact mode switches over the full gait cycle and therefore the algorithm shortens the overall prediction horizon from the nominal. The trotting gait in Figure 5-3b has 2 contact switches over the nominal horizon, since diagonal pairs of legs switch simultaneously and can be captured more easily by the nominal prediction horizon. For those more complex gaits that require more contact switches, it is easy to simply increase N, the number of prediction discretizations.

Algorithm 1: Adaptive Timestep Segmentation

nominal timestep: $\Delta t_{nominal}$;
scheduled contact to swing phase: $\mathbf{\Phi}_{c\to\bar{c}} = \frac{T_p - T_{\bar{c}}}{T_p}$;
for $k = 0$ *to NUM_PREDICTIONS-1* **do**
    number of feet in swing: $N_{\bar{c}} = 0$;
    $\Delta t_{max} = \infty$
    for $i = 0$ *to NUM_FEET-1* **do**
        if $\mathbf{\Phi}_{i,k} < \mathbf{\Phi}_{c\to\bar{c}}$ **then**
            foot in contact: $\boldsymbol{s}_{i,k} = 1$;
            time until foot switches to swing: $\Delta t_{switch} = T_p(\mathbf{\Phi}_{c\to\bar{c}} - \mathbf{\Phi}_{i,k})$;
        else
            foot out of contact: $\boldsymbol{s}_{i,k} = 0$;
            time until foot switches to contact: $\Delta t_{switch} = T_p(1 - \mathbf{\Phi}_{i,k})$;
            $N_{\bar{c}}$++;
        if $\Delta t_{switch} < \Delta t_{max}$ **then**
            $\Delta t_{max} = \Delta t_{switch}$;
    choose the current segment duration: Δt_k^*;
    if $N_{\bar{c}} = NUM\_FEET$ **then**
        $\Delta t_k^* = \Delta t_{max}$;
        $flight\_phase = 1$;
    else if $\Delta t_{max} < \Delta t_{nominal}$ **then**
        $\Delta t_k^* = \Delta t_{max}$;
        $flight\_phase = 0$;
    else
        nominally divide time till switch evenly: $\Delta t_k^* = \frac{\Delta t_{max}}{round(\frac{\Delta t_{max}}{\Delta t_{nominal}})}$;
        $flight\_phase = 0$;
    $\mathbf{\Phi}_{i,k+1} = \text{mod}(\mathbf{\Phi}_{i,k} + \frac{\Delta t_k^*}{T_P}, 1)$;



(a) Adapted Amble Gait Segmentation



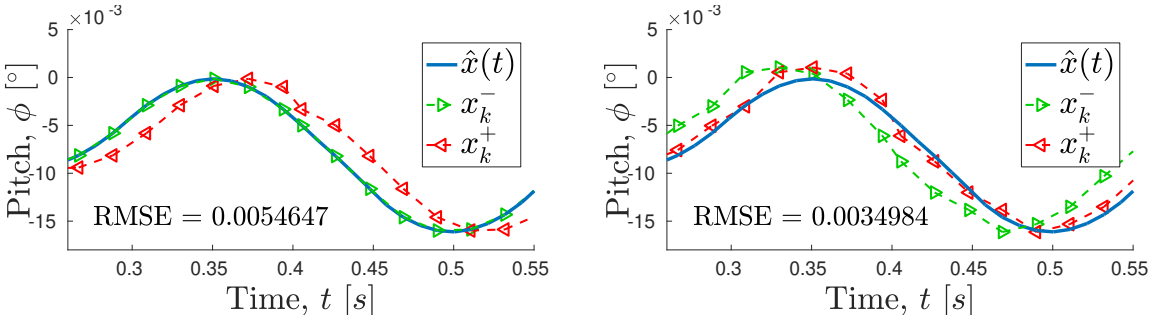
(b) Adapted Trot Gait Segmentation

Figure 5-3: **Adaptive Timestep Segmentation.** Top four bars represent the upcoming gait scheduled contact sequences for each of the feet. The bottom bars are the nominal timestep segmentation and the adapted segmentation based on contact mode switches where each segment gets lighter further along the prediction horizon.

This algorithm is $\mathcal{O}(N)$, as it only requires one predictive pass. It is straightforward and computationally inexpensive to add checks for various boolean flags that get used in the calculations. For example, if a foot is determined to be in contact, we can check if it was previously in contact or not, which signifies a touchdown event. Keeping track of touchdown events lets the optimization initialization know if it should use the heuristic reference footstep location or the footstep location from the previous iteration. The no foot slip constraints will always be enforced so it is not necessary to explicitly flag touchdown events, but initializing and regularizing with already correct footstep logic decreases the number of iterations and complexity needed to converge.

5.2 Prediction Delay Compensation

The optimization solve time may be slower than the main control loop and runs irregularly so often there are timing delays resulting in discrepancies between the



(a) No Prediction (b) With Prediction

Figure 5-4: **Predictive Delay Compensation.** Blue trajectory shows the actual robot states, while the green and red trajectories show the initial state at the beginning and end of the optimization respectively.

current state of the robot with its associated scheduled contact mode and the returned optimal solution. Optimized forces and step locations are only applied after the time a solution is found, dt^*, so the initial optimization states at the beginning and end of the optimization are the same, $\boldsymbol{x}_0^- = \boldsymbol{x}_0^+$, but the actual robot state will likely have changed, $\hat{\boldsymbol{x}}(t) \neq \hat{\boldsymbol{x}}(t + dt^*)$

To compensate this, the initial state given to the solver, $\boldsymbol{x}_0$ is not the current instantaneous state of the robot, but rather a predicted future state that the robot can be expected to be near when a solution is returned, as solved for by the linear dynamics approximation

$$\boldsymbol{x}_0 = \boldsymbol{A}(\bar{dt})\hat{\boldsymbol{x}}(t) + \boldsymbol{B}(\bar{dt})h(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t), \boldsymbol{\Phi}(t)) + \boldsymbol{d}(\bar{dt}) \tag{5.1}$$

where $\boldsymbol{u}(t)$ is the current input from the last solution and the dynamics are integrated forward using $\bar{dt}$, a filtered average optimization solve time. Occasionally, the time taken to solve will be much greater or lesser than the average, but in a large majority of cases, predicting forward proves to be beneficial.

Similarly, the phase of each foot progresses continuously, so by the time the optimization returns a value, the foot may also be scheduled to be in a different contact state than the solver thinks. Accounting for this, each initial leg phase passed into

the controller can be predictively modified according to

$$d\bar{\Phi} = \frac{\bar{dt}}{T_P} \tag{5.2}$$

$$\Phi_k = \text{mod}(\Phi(t) + d\bar{\Phi}, 1) \tag{5.3}$$

where T_P is the nominal gait period time and $\phi(t)$ is the current phase of the foot. This starts the controller optimizing with the contact mode that the robot will likely be in at the time a solution is returned. If a foot is nearing the end of stance, solving for forces assuming that the foot will still be in stance when the optimization converges will lead to unexecutable commands. If there is a mode change before the solution is returned, then the result found is no longer possible as a foot may have been lifted off the ground and no force can be applied, or a foot has touched down and the step location can no longer change. Figure 5-4 shows that the Root Mean Squared Error (RMSE) between the current robot state, $\boldsymbol{x}(t)$, and initial optimization state when a solution is returned, $\boldsymbol{x}_0^+$, is lowered by using prediction.

5.3 Asynchronous Solution Filtering

Despite efforts to compensate for the discrepancies between the optimization solution contact state and the gait scheduled contact state, there is no guarantee that they will be synchronized. To mitigate those effects, an Asynchronous filter, shown in Figure 5-5, is designed to decide appropriate forces and footstep locations in cases where the RPC solution differs from the currently scheduled contact state. It also smoothly filters forces and footsteps to avoid large discrete jumps in solutions as optimization results are returned.

The prediction delay compensation in Section 5.2 starts the optimization acting on a future state and contact mode based on the moving average of the optimization solution time. However, this means that in cases where the actual gait schedule has a differing contact mode than the resulting RPC solution, there must be a way to account for the difference and still execute physically feasible commands. For both

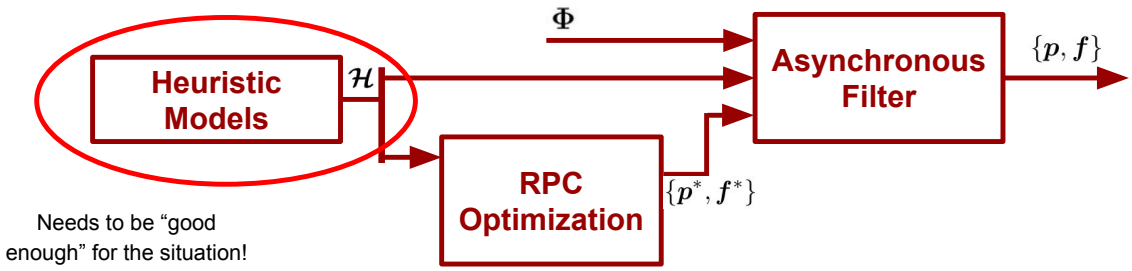


Figure 5-5: **Asynchronous Solution Filtering.** Optimization result contact mode and gait scheduled contact mode may not always align, so the asynchronous filter chooses to use the heuristic models or RPC solution. This is dependent on the heuristic models being adequate enough to provide a "good enough" solution for the situation.

the footstep and ground reaction forces of each leg, the nominal values calculated by the heuristic models discussed in Section 5.5 are used for temporary solutions until a new optimization converges. As stated, the heuristics are designed precisely for this purpose. Therefore in the event that the optimization does not return a solution or cannot converge before the contact mode changes, the robot simply executes the heuristic and is able to temporarily get away with a potentially suboptimal force and foot placement. This temporary control action is designed for idealized conditions and will not stabilize the robot indefinitely. However, it will provide a "good enough" solution to avoid a fall if heuristics are designed well. This makes the design and quality of heuristics described in Chapter 4 even more critical.

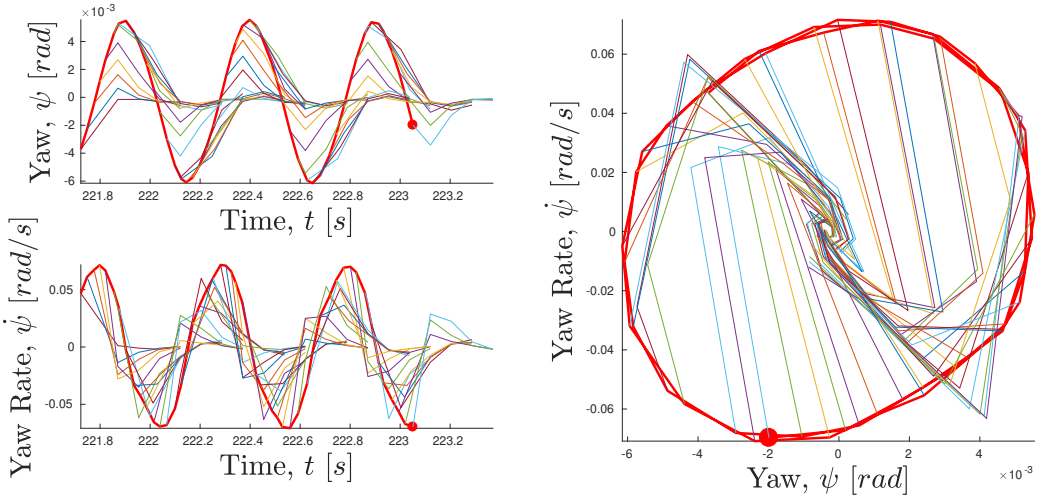
This is especially important in gaits that have very distinct contact modes, such as trotting. In a version of the trot gait, the diagonal pairs of legs are picked up and put down in a perfectly alternating pattern. Even with the prediction delay compensation, it is not guaranteed that the predicted phase and the scheduled phase agree on the same contact state. Therefore, without this heuristic there would be no force command from the stance legs to keep the robot upright. By using the average solve time to predict forward for initial states and gait schedule, it is expected that heuristics will be required at the beginning of half of the contact changes. Only when the optimized solution contact mode and the scheduled gait sequence return to a synchronized state, can the results from the RPC be confidently used again.

5.4 Gain Tuning

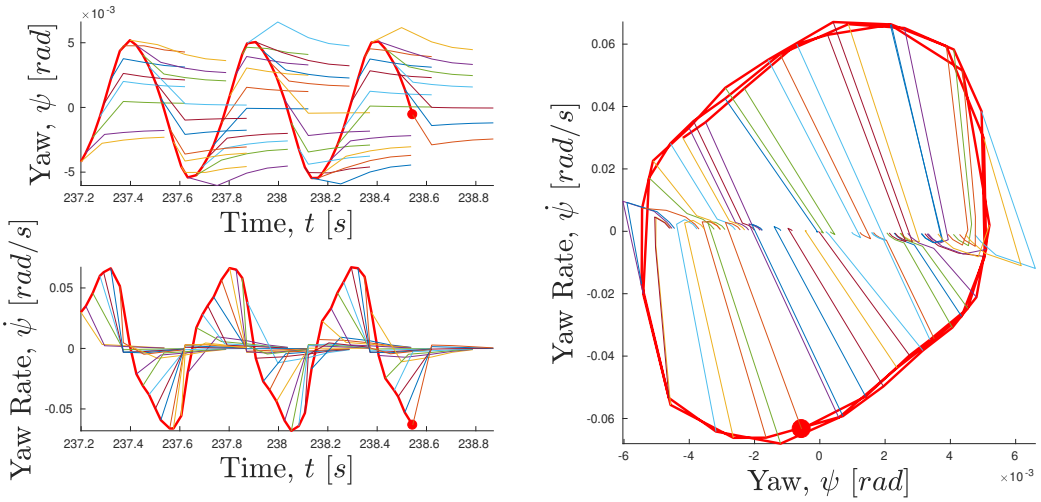
With most controllers of any kind, the tuning of the gains is a critical component to its success. Often this requires many hours of tedious work as researchers manually change each value until performance improves. To tune the gains for RPC, simple desired setpoints were commanded as values for the individual gains for each state and input were automatically swept over. Rough stability bounds for the gains were found manually that allowed the robot to automatically tune its parameters without falling over in simulation by simply finding the gains that produced the best tracking. After allowing the robot to trot for several gait cycles in steady state at each value, the total summed tracking errors were compared and selected. This process was done several times for all gains to ensure changes to each did not deteriorate performance of previously tuned gains significantly.

Figure 5-6 shows the two extreme bounds of tuning the yaw behavior of the robot while trotting. While both are stable as the bounds were chosen to be at values that did not cause the robot to fall, they produce very different behaviors. The red lines are the actual measured states of the robot and the multicolored lines are the predicted states returned from the RPC. Low yaw rate gain, $R_{\dot{\psi}}$, produces a more natural and smooth sinusoidal pattern when not forcing the yaw rate to go to zero as is the case with the high gain. The tracking error for the states is used as a metric for tuning, as is the predicted state error. When the predicted states also have good tracking, then the controller is better conditioned and results produce smoother, less sporadic results. The optimization converges faster when predicted states don't fight against the natural tendencies of the dynamics.

The predicted state trajectory does not line up with the actual state because the prediction model is based on simple uncoupled linear dynamics while the actual robot is a highly nonlinear hybrid system that must adhere to its natural physics regardless of how much the controller fights it. This will be addressed in related work. However, for the purposes of tuning the gains, it is enough to attempt to track as closely to a simple setpoint as possible.



(a) Low Gain, $R_{\dot{\psi}} = 1$



(b) High Gain, $R_{\dot{\psi}} = 500$

Figure 5-6: **Intuitive Gain Tuning.** Red lines indicate the actual motion of the robot and the thin multicolored lines represent the predicted states at each moment in time as the yaw rate gain is swept from $R_{\dot{\psi}} = 1$ in (5-6a) to $R_{\dot{\psi}} = 500$ in (5-6b). In the high gain case, the robot is fighting against the natural dynamics, whereas the low gain case allows the robot to embrace them and therefore converges faster and produces smoother results.

5.5 Extracted Heuristic Models

The heuristics extracted in Chapter 4 are primarily designed to serve as regularization for shaping the cost function favorably. However, in the majority of situations, the heuristics for desired CoM state, foot locations, and ground reaction forces provide a "good enough" solution for locomotion. By simply executing the heuristics, the robot should be able to remain upright for at least a small amount of time. This gives the RPC enough relaxation to return a new solution without falling over. The heuristics are embedded directly into the optimization through the error term in the quadratic cost function. However the same function

$$\boldsymbol{\mathcal{H}}_{\boldsymbol{\chi}}(\boldsymbol{\chi}, \Phi, \boldsymbol{x}_d) \tag{5.4}$$

serves to provide an immediate solution in the event that the optimization has not yet returned a feasible result for the current contact mode. Using the part of (5.4) pertaining to the forces and footstep locations

$$\boldsymbol{\mathcal{H}}_{\boldsymbol{r}}(\boldsymbol{\chi}, \Phi, \boldsymbol{x}_d) \tag{5.5}$$

$$\boldsymbol{\mathcal{H}}_{\boldsymbol{f}}(\boldsymbol{\chi}, \Phi, \boldsymbol{x}_d) \tag{5.6}$$

the robot has a backup input to execute rather than a null solution which would cause the robot to immediately fall to the ground. The importance of adequate heuristics for the inputs increases dramatically as solve frequencies decrease. The longer that the robot waits for a solution, the longer the it will need to use the heuristics for temporary balance. While it is important to design meaningful heuristics for all situations, when the solve frequency is high enough, there is a shorter period of uncertainty and the heuristics do not need to be as immediately capable of stabilizing the robot. However, solve frequency is largely dependent on the quality of the models as well.

In addition, by finding heuristics that approximate the natural dynamics as was done for the robot's velocity and gait phase-dependent pitch dynamics for Figure

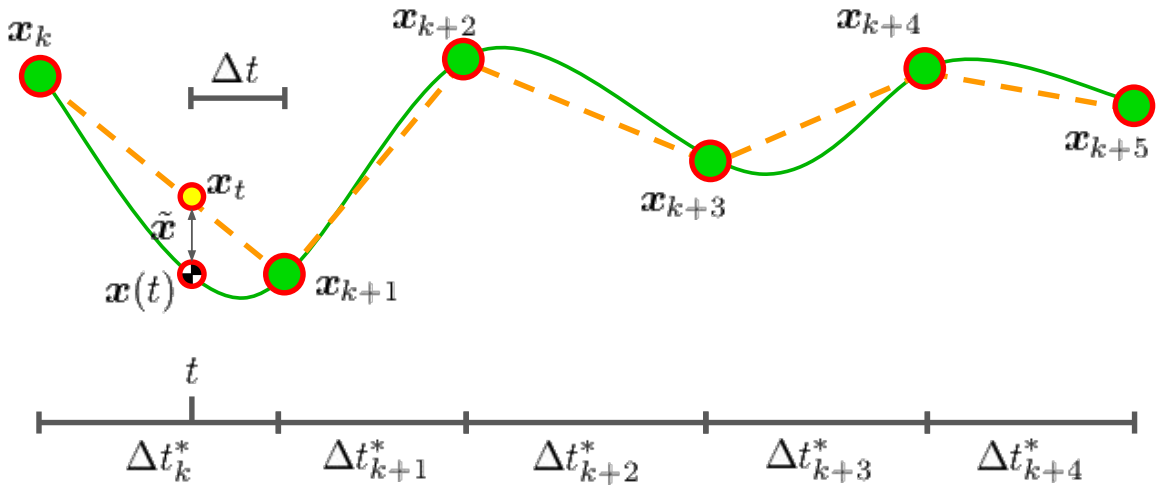


Figure 5-7: **Linear Approximation Error.** Heuristics can help guide the optimization towards a solution that embraces the robot's natural dynamics rather than fights against them despite the simplified dynamics model.

4-7, we can inform the optimization about the coupled nonlinear dynamics through the regularization for the states, $\boldsymbol{\mathcal{H}}_{\boldsymbol{x}}(\boldsymbol{\chi}, \Phi, \boldsymbol{x}_d)$, while continuing to use the simplified model. Figure 5-7 shows that the error of the state between discretization steps is generally reduced if the knot points lie close to the naturally occurring trajectory. By embracing the dynamics rather than fighting against them by attempting to force the solution to an impossible trajectory, the optimization converges faster and reduces the cost on the states to find better, smoother solutions for the inputs.

5.6 Tracking Performance

Despite the difficulties for implementation of nonlinear optimization-based controllers in real-time, using the methods presented, the RPC was implemented successfully on the MIT Cheetah 3 robot platform. The optimization is solved using the freely available IPOPT NLP solver [81] interfaced in C++. It runs on a separate thread, but on the same ADL embedded Quad core PC with a 2nd Gen Core i7 CPU as the main control loop described in [6]. As mentioned throughout the paper, the RPC runs asynchronously to the main control loop with no guarantees on solution times. Therefore while the control loop runs at a fixed $1kHz$, the RPC solution time varies

based on computing power available.

We note that the stability of the robot improves as solve frequency increases, with good stability over $40Hz$, decent stability $20-40Hz$, and unstable under $20Hz$. As dynamic legged locomotion stability has not yet been formally defined, the vague performance measure of stability here simply means that the robot is able to continue taking steps. Since generally the heuristic-based initial guess is designed to provide an adequate solution to at least remain stable, we can limit the maximum solve time to be $50ms$ which corresponds to the lowest stable frequency and rely on the heuristics if a solution was not found. In practice the solver generally runs at $60-80Hz$ on the robot during steady state locomotion and around $40-60Hz$ during worst-case large disturbances as the optimization must search further from the heuristic regularization for better solutions.

The xyz ground reaction forces resulting from the RPC over a short period encompassing three full gait cycles while trotting in place. These are shown in Figure 5-8. The gait schedule contact sequence for the trotting gait is also depicted below for the six footsteps to show the contact mode changes aligned with the forces. When feet are no longer in stance, the optimization and heuristics both correctly constrain the force commands to be $\boldsymbol{f}_i = \mathbf{0}$. Each diagonal pair of legs is capable of stabilizing the robot while trotting in place. A discrete jump in the forces can be seen a short time after the beginning of each new contact mode, signifying the switch from the short period of heuristic force calculation to the RPC solution. This is the result of the asynchronous solution filter described in Section 5.3. Without this filter, there would be no forces commanded to the legs during these periods and the robot would drop. In general, this period is short and is dependent on the solver speed.

In addition to selecting the ground reaction forces at the contact points, future footsteps are also chosen at the same time. The benefit of simultaneously optimizing for both is that footsteps are chosen in the same context as future stabilizing forces. This means that under large disturbances, the tradeoff between using large forces and taking wide steps can be managed. It also means that while in steady state trotting, it already has knowledge of future step locations and can modify them to cause smooth

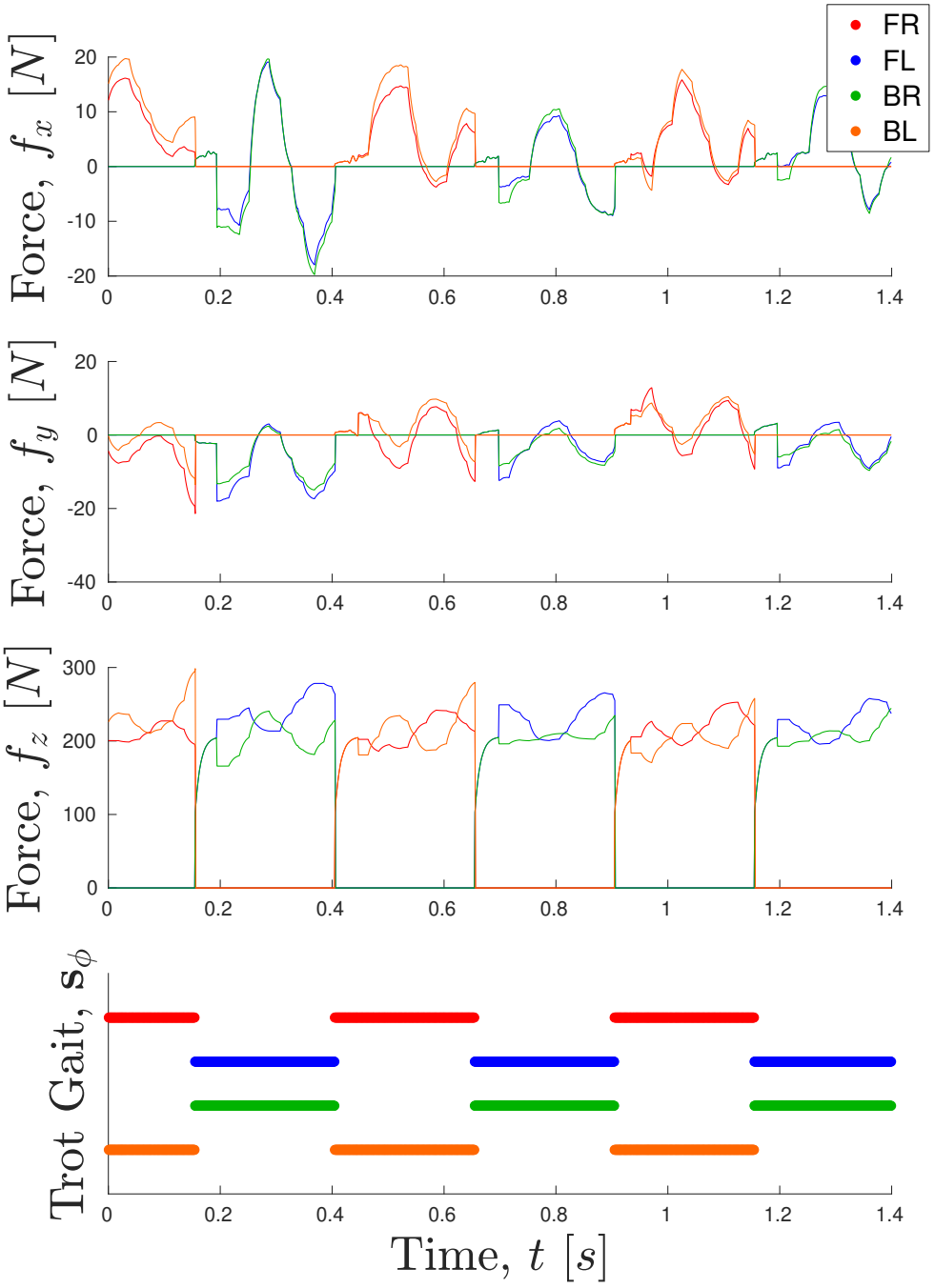
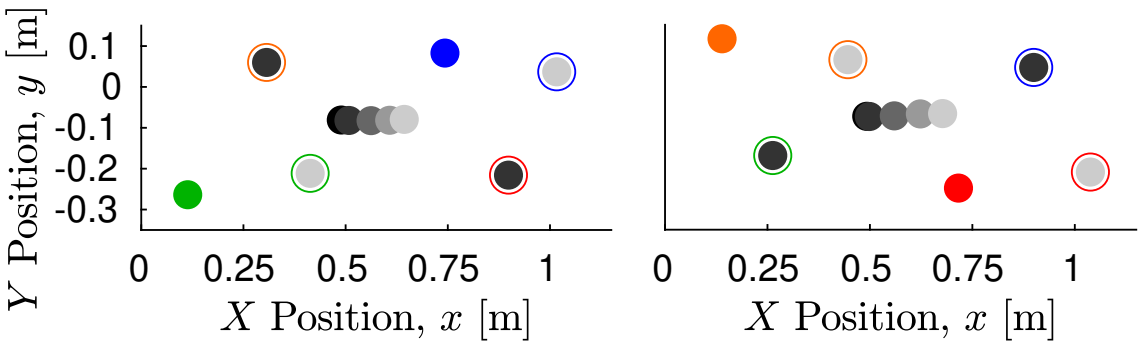


Figure 5-8: **Ground Reaction Forces.** Optimized ground reaction forces during a trotting gait. Forces are smoothly filtered even though the RPC returns discrete constant forces at a lower frequency than the control loop.



(a) Timestep t_1 (b) Timestep t_2

Figure 5-9: **Predicted Footstep Locations.** Solid colored dots represent the current contact foot locations while the colored circles signify the predicted footstep locations. The solid circle inside of the predicted footsteps corresponds to the future time segment of touchdown, where the darker the circle, the closer to the current time. CoM predicted trajectory is shown in the middle with decreasingly dark dots over time.

body trajectories prior to committing to a certain step location for the duration of the stance phase.

Figure 5-9 shows two example predicted footstep optimization results during $1.4\frac{m}{s}$ forward locomotion using the robot with a trotting gait. The figure depicts a 2D projection in the XY plane of the (x, y) coordinates for the CoM, predicted CoM, contact feet, and predicted footstep locations. In Figure 5-9a, the front left (blue) and back right (green) feet are on the ground, as signified by solid colored circles, while the next footstep locations for the front right (red) and back left (orange) feet are solved for. The prediction horizon is long enough such that the next footsteps for the front left and back right feet are also already being found before they have even entered their swing phase. Figure 5-9b shows the results after the feet have changed their contact state. Now the front right and back left are in contact while the front left and back right are swinging to the optimally calculated step locations. The CoM trajectory over five future timesteps is shown in decreasingly dark gray circles.

As stated in Section 1.3, a goal for creating a dynamic locomotion controller is to be able to accurately track velocity and turn rate commands. The robot was able to closely track the desired velocities during an experiment both forwards and backwards. The body's orientation throughout is modulated close to level with minor

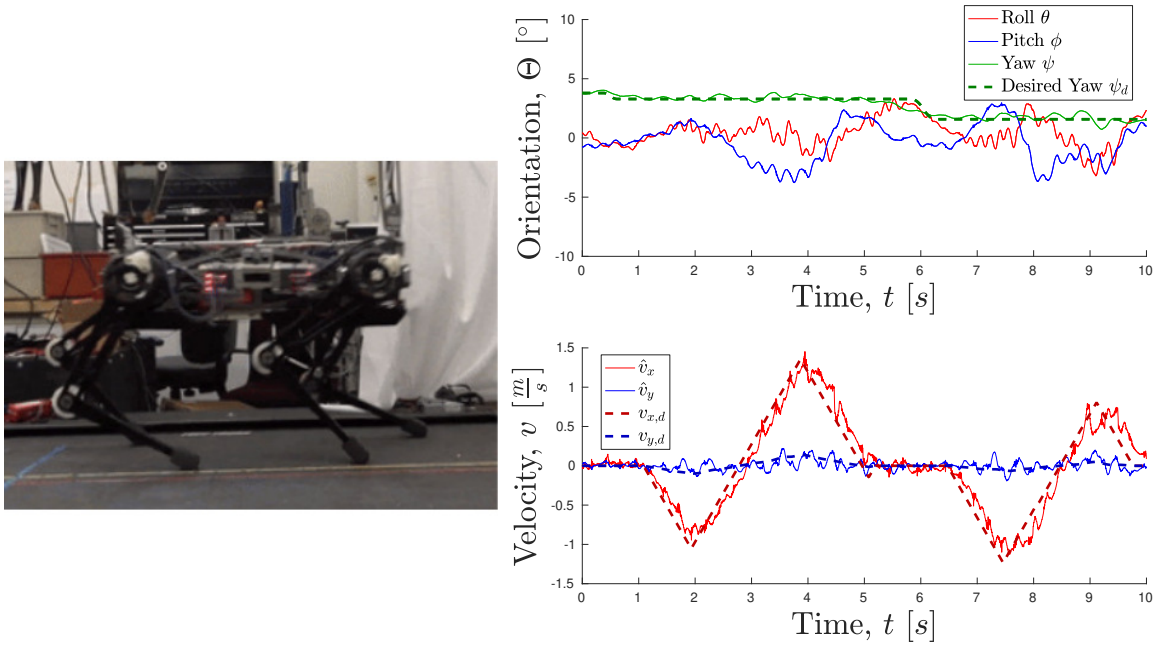


Figure 5-10: **Velocity Tracking.** The robot was given forward and backwards velocity commands in the x direction and was able to closely track the desired values while keeping its body orientation close to flat.

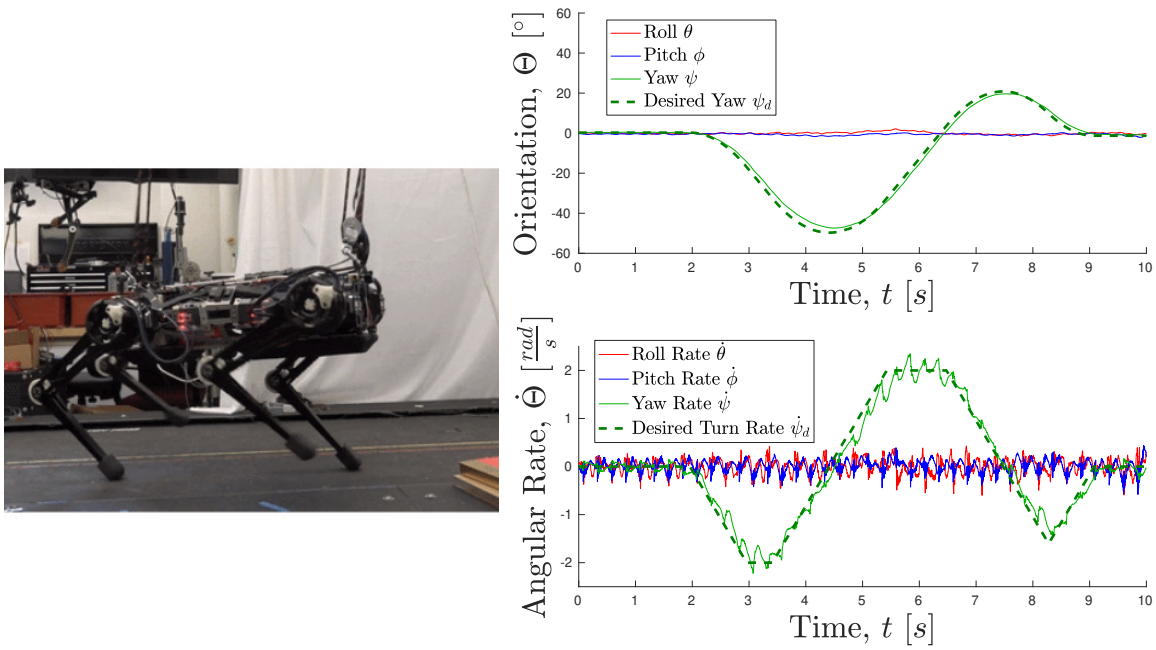


Figure 5-11: **Turn Rate Tracking.** The robot was given a turning rate command which it was able to track with little error over the duration of the test.

fluctuations, never exceeding more than 5°, which usually occurs during the sharp velocity changes. However, it returns flat when the robot is in steady state. Figure 5-10 shows the robot's ability to track commanded translational velocities. The maximum commanded velocity during this test is $1.4\frac{m}{s}$ while trotting. Turning rate commands were also sent to the robot as seen in Figure 5-11. Maximum turning rate given was $2\frac{rad}{s}$ with a mean orientation error in roll of 0.47° and 0.48° in pitch throughout. This demonstrates a successful hardware implementation of the basic capabilities that the robot should be able to perform.

Chapter 6

Robustness Experiments

While previous sections showed good performance in controlled environments with known commands, the controller must be robust to a variety of disturbances to truly be useful. The locomotion controller is not designed purely for operation in controlled environments, but rather for situations where it will constantly need to interact with its surroundings and encounter unexpected terrains and disturbances. Therefore it needs to be robust to a variety of outside disturbances and remain upright without falling over. Disturbance tests were conducted on the Mini Cheetah hardware rather than Cheetah 3 because of the ease in experiment procedure. Being able to run many experiments on the hardware rapidly is invaluable to iterate and debug the controller to characterize its behavior in robustness tests.

6.1 Effects of Injecting Regularization Heuristics

First, it is important to see the effects of injecting the regularization heuristics extracted throughout Chapter 4 on the hardware. By adding heuristics, $\boldsymbol{\mathcal{H}}_{\boldsymbol{\chi}}$, to the initial guess and reference trajectories, $\boldsymbol{\chi}_{ref}$, and enabling regularization, $\boldsymbol{R}$, the capabilities and viable operating regions improve. Figure 6-1 shows the progression from the Naïve nonlinear MPC implementation up until the fully robust dynamic RPC that is running on the robot. It highlights the importance of the contributions outlined in this work. Without well designed heuristics being used as regularization

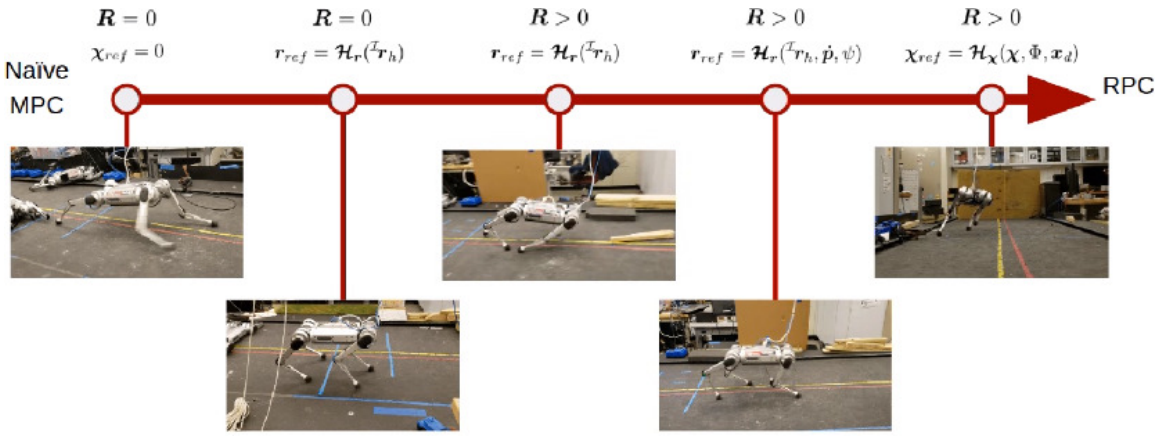


Figure 6-1: **Injecting Heuristic Regularization.** Going from the naïve MPC implementation that does not take an initial heuristic guess and uses no regularization, up until the full RPC implementation, the robots capabilities increase as more regularization heuristics are designed and extracted.

in the controller, the nonlinear optimization would not be able to produce adequate forces and footstep locations for locomotion.

With no informed initial guess, $\boldsymbol{\chi}_{ref} = 0$, and no regularization, $\boldsymbol{R} = 0$, the controller is optimizing footsteps and forces to find solutions that attempt to stabilize the CoM at a given height, $p_{z,d} = 0.25m$, and flat orientation, $\boldsymbol{\Theta}_d = \begin{bmatrix}0, 0, 0\end{bmatrix}^T$. Foot positions are sporadically selected and jump between local minima. Since the controller has no concept of needing to swing the legs or the effect of leg inertia on the body, footsteps found discretely move large distances during swing and the robot is barely able to stand as legs swing around wildly. Next, the controller is given a model of the robot with an initial guess for foot placements to be under the hips in the inertial frame, $\boldsymbol{r}_{ref} = {}^{\mathcal{I}}\boldsymbol{r}_h$. With this, the robot generally takes steps with the feet under the hips, but still jumps around local minima in the area. Since foot placements are less sensitive than the other decision variables in the defined control model, they tend to have a larger variation.

By regularizing the footsteps with the heuristics, $\boldsymbol{R} > 0$, the footsteps are biased towards the reference location under the robot hips. This eliminates the jumping between local minima, but as the robot begins to move forward with larger velocities, the desired step location under the hips becomes inadequate and it falls when moving

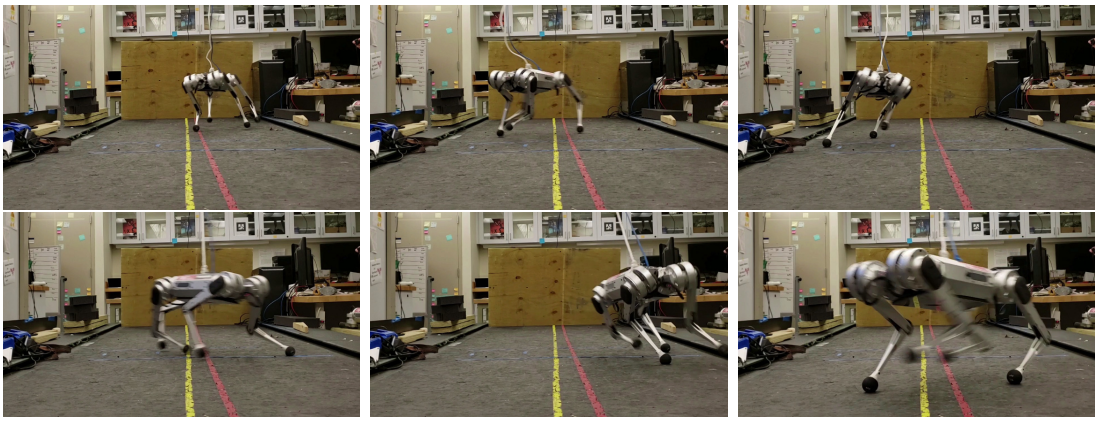


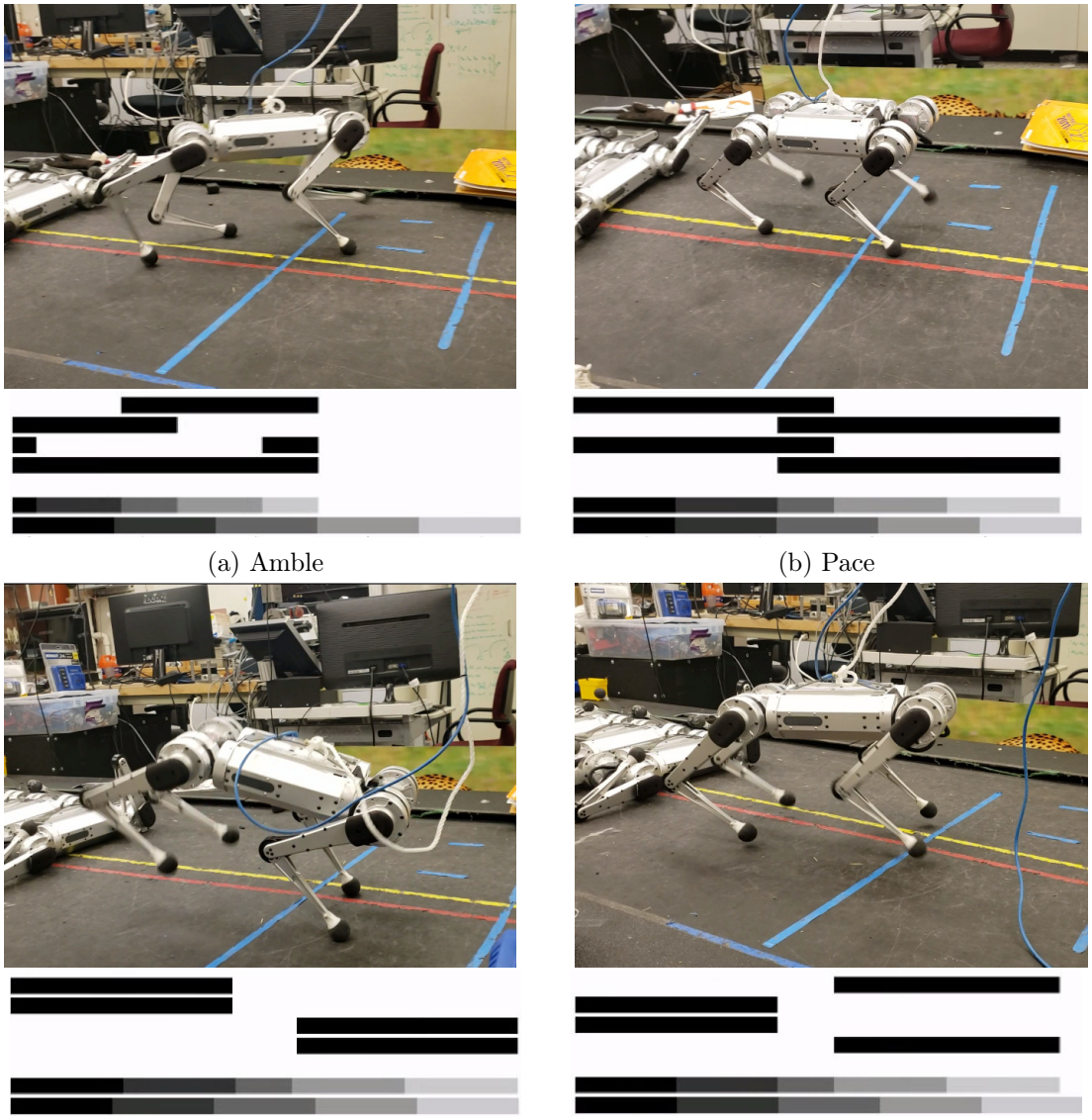
Figure 6-2: **High Speed Turning.** Mini Cheetah is able to take quick, tight turns with a combination of high translational velocity and quickly changing turn rate commands due to the extracted heuristics and robustness of the controller.

in the translational direction or turning quickly. Using the combination of expert designed and extracted heuristics for translational and rotational velocities, $\boldsymbol{r}_{ref} = \boldsymbol{\mathcal{H}}_{\boldsymbol{r}}({}^{\mathcal{I}}\boldsymbol{r}_h, \dot{\boldsymbol{p}}, \dot{\psi})$, Mini Cheetah is able to independently trot forward and turn. These are the basic capabilities that the robot should be able to execute.

The final result of the robust dynamic RPC framework is the high speed turning, for which a the high speed turning extracted heuristic foot placement, $\boldsymbol{\mathcal{H}}_{\boldsymbol{r}}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}})$, was found. With the velocity and turn rate command dependent frequency, the robot is able to trot with a gait period of $T_P = 0.35s$, switching phase of $\Phi_{c \to \bar{c}} = 0.3$. Using the high speed turning heuristic, we could command the robot to trot at $\dot{p}_{x,d} = 2\frac{m}{s}$ with a turning rate of $\dot{\psi}_d = -2\frac{rad}{s}$ and an immediate switch to $\dot{\psi}_d = 2\frac{rad}{s}$. The result is seen in Figure 6-2 as the robot takes tight dynamic turns using a dynamic gait with flight periods. This is a behavior that has not been exhibited in other legged robots and would not be possible without the RPC design and extracted heuristic to deal with the situation.

6.2 Various Gaits

Using the flexible gait scheduling framework, a variety of gaits can be easily parametrized. RPC is designed to be gait agnostic so it does not optimize for contact modes, but also does not need to have a specific pattern. Because of the fast rate of replanning,



(a) Amble (b) Pace

(c) Bound (d) Running Trot

Figure 6-3: **Various Gaits.** RPC works exactly the same for various gait schedules being generated by the gait scheduler and segmented by the Adaptive Timestep Segmentation algorithm, including gaits with overlapping footsteps and flight periods. A few common gaits are shown, but any parametrized gait can be passed to the optimization.

Table 6.1: Parametrized Dynamic Gaits

Gait Name	Period Time	Switching Phase	Phase Offset, $\Phi_{o,i}$			
	T_P	$\Phi_{c\to\bar{c}}$	$\Phi_{o,FR}$	$\Phi_{o,FL}$	$\Phi_{o,BR}$	$\Phi_{o,BL}$
Stand	∞	1.0	0.5	0.5	0.5	0.5
Amble	0.75 s	0.625	0.0	0.5	0.25	0.75
Walking Trot	0.6 s	0.6	0.0	0.5	0.5	0.0
Trot	0.5 s	0.5	0.0	0.5	0.5	0.0
Running Trot	0.5 s	0.2	0.0	0.5	0.5	0.0
Pace	0.4 s	0.5	0.0	0.5	0.0	0.5
Bound	0.4 s	0.4	0.0	0.0	0.5	0.5
Rotary Gallop	0.4 s	0.2	0.0	0.857	0.357	0.5
Pronk	0.4 s	0.3	0.0	0.0	0.0	0.0

gaits con be adapted or switched completely online. The standard gaits designed represent some most commonly found gaits in nature used by four legged animals are parametrized in Table 6.1. The gait scheduler modifies the period times and switching phases dependent on commanded velocities and turning rates.

While trotting has been used as the nominal gait throughout the rest of the chapters, an example of the more common natural gaits seen in nature by four legged animals were parametrized according to the table and executed by the RPC. Screenshots of four examples are seen in Figure 6-3. Ambling in Figure 6-3a is a slower gait that generally has either 2 or 3 feet in contact. It provides static stability at times, but is less stable during disturbances and faster velocities because of its low gait frequency. Pacing seen in Figure 6-3b moves the pairs of legs on each side in an out of contact simultaneously and features a natural tendency to move the body side to side, while the optimization moves the feet slightly inward towards the center of mass to reduce rolling motion. Bounding in Figure 6-3c simultaneously switches contact with the front and back feet as pairs. This causes large periodic pitching motions. Finally, the running trot in Figure 6-3d pairs the diagonal legs together as the regular trot, but features a large flight phase between each switch. It is important to note that the gains and heuristics were generalizable between all of the gaits and the robot was able to remain upright throughout all of them with no need for modification. However, each gait has characteristic motions signifying that it would make sense

Figure 6-4: **Traversing Rough Terrain.** Though the controller assumes flat, stable, rigid ground, it is robust enough to traverse terrains with unstable, slippery surfaces, soft bouncy mats, and varying height objects.

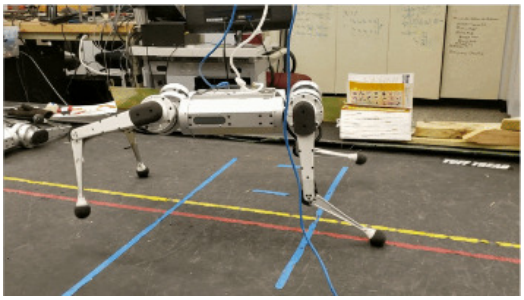
to run separate heuristic extractions for each standard gait for even better results despite the ability of RPC to generalize the trotting behavior through optimization.

6.3 Disturbance Rejection

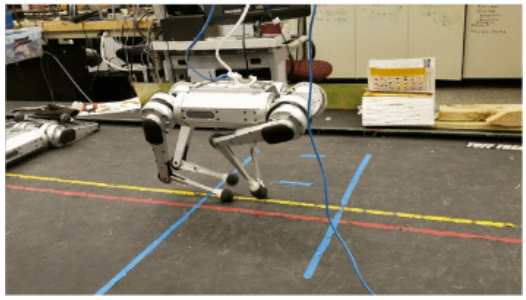
As legged robots are in constant interaction with their surroundings, a locomotion controller should be able to handle a reasonable amount of disturbances from its environment. Designing robots to operate in real world environments means that they will often need to interact with objects and outside forces and must regain balance to continue. Disturbance rejection is highly dependent on the update frequency of the RPC optimization. As the robot is pushed around or encounters objects unexpectedly, the previous prediction model may be far off from the actual, but by replanning quickly, the new accurate state will be used and the previous solution is discarded for the new one that is equipped to stabilize the robot in the unexpected situation. If too much time is taken between updates, the robot will be calculating forces and step locations using incorrect robot states and kinematics. Therefore, the work done in Chapters 3 - 5 to make the optimization run as fast as possible is crucial to robustness.

6.3.1 Traversing Rough Terrain

A key component of a successfully robust locomotion controller is the ability to traverse highly unstructured terrains. To take full advantage of a legged system, it needs to be able to handle environments that wheeled and tracked vehicles cannot. This includes moving over and stepping down from large objects, stepping on ground with



(a) Increased Size Model



(b) Reduced Size Model

Figure 6-5: **Model Changes.** The controller is given a model of the robot that has twice the dimensions as the actual in 6-5a and that is 10% of its actual in 6-5b. However, the controller is able to robustly control the robot despite the model inaccuracies.

different stiffnesses, and recovering from slips and unstable moving footholds. Figure 6-4 shows screenshots from the robot traversing an environment with different obstacles with no prior knowledge or planning. The goal of this work is to develop a blind reactive locomotion controller that can operate in a variety of environments without the need for higher level planning or knowledge of the surroundings. By achieving this, the robot is not dependent on rigid planning and if the vision or mapping system fails, the robot can still reasonably stay upright and blindly continue its task.

6.3.2 Model Inaccuracies

The controller is also robust to model inaccuracies. As stated, the controller is passed in a model of the robot's kinematics as well as its approximate total mass and inertia which it uses to check kinematic limits, calculate heuristics, and check the dynamics constraint. However, the overarching theme for the controller design is to use simple models and minimum physical feasibility constraints to allow locomotion. This means that even without a perfect robot model, the reactive replanning of optimization forces and parameter adaptation laws will be able to robustly control the CoM states despite an incorrect model. Figure 6-5 depicts the ability to stabilize the CoM with deliberately incorrect robot models being provided to the RPC. In Figure 6-5a, the robot model given is twice as wide and long as the actual robot and so the heuristic $\boldsymbol{\mathcal{H}_r}({}^{\mathcal{I}}\boldsymbol{r}_h)$ puts the control model robot hips outwards from the actual robot body

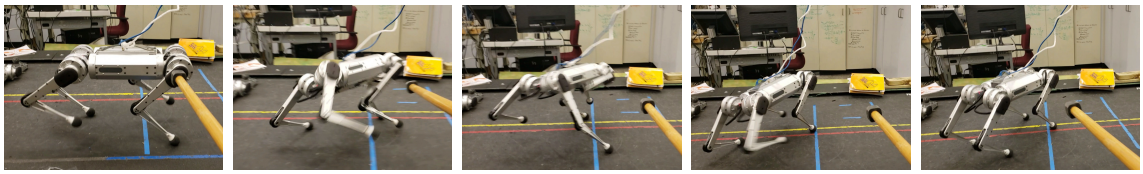


Figure 6-6: **Push Recovery.** The controller is robust to unexpected impulse pushes. Recovery to steady state locomotion is achieved within a small number of gait cycles.

which makes the nominal regularized steps wider than the actual design. The second Figure 6-5b shows the opposite, where the robot model provided to the controller is 10% as wide and long as the actual robot so the nominal stepping position under the model's hip is much closer together than it would be on the actual system. However, despite these large inaccuracies in the model, the robot is able to stabilize itself and take steps while turning and transitionally trotting.

6.3.3 Push Recovery

While many factory robots currently operate in controlled environments with minimal or no unexpected interaction with surroundings, legged robots are inherently intended to regularly work alongside humans and in situations that may require manipulating its environment or managing outside forces. For instance, a robot working in an earthquake disaster response may have a object fall on it or debris hit it as it navigates the scene. In addition, even if the robot is equipped with environment sensors, perception systems often fail due to low visibility, current short comings in the technology, or simply occlusions. If the robot loses vision or LiDAR sensing, it may collide with objects that it cannot avoid. This is where the basic mid-level balance and locomotion controller must be robust enough to blindly overcome the hit and continue balancing until sensors are able to provide meaningful information again.

There are an infinite number of possibilities for ways to push a robot in 3D with a constantly progressing gait. One such example is pictured in Figure 6-6 where the front right hip is given a sudden, rough impulsive push. The push occurs while the front right leg is in stance, but about to take off and causes the robot to slip and roll its body. The footstep selection from RPC is far in the direction of the push

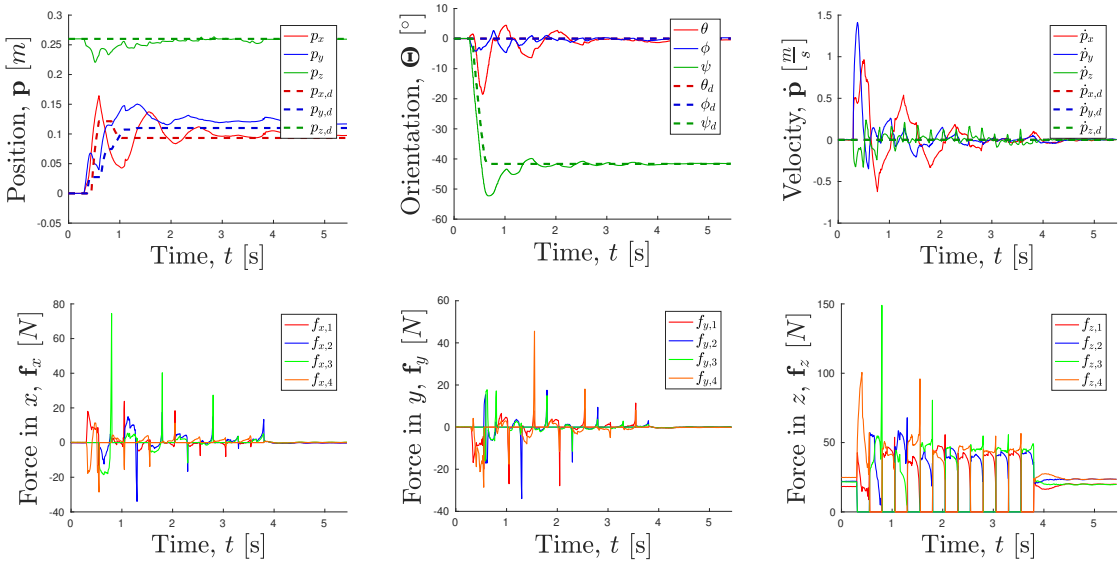


Figure 6-7: **Push Recovery States and Forces.** The impulsive push disturbs the robot and causes it to begin trotting to place the feet in favorable positions for returning to its nominal standing state. The robot manages its forces and footsteps to regain control within a few gait cycles.

causing the front feet to cross, while the back right foot is placed far in the opposite direction to create a larger moment arm for stability. as the body's angular velocity is brought back under control, the opposite feet place themselves favorably to resume standard operation. Finally, within a couple gait cycles, the Mini Cheetah is back under control and stably returns to standing with the feet placed under the hips.

In Figure 6-7 we can track the robot's states and calculated forces during the disturbance. From the stable, standing rest position, the stick caused a large nearly impulsive push of about $13.5Ns$ resulting in a total robot velocity of $\sim 1.5\frac{m}{s}$. Lateral forces with magnitude of up to $86N$ were commanded the feet to counteract the robot's momentum. The robot was able to regain control within five gait cycles despite being pushed about $0.1m$ in both the x and the y directions, as well as forces to rotate over 40° in the yaw direction. Maximum disturbed roll reached 18.5°, but was returned to approximately flat within one gait cycle. The objective for the test was simply to remain upright and have no velocity so desired position and yaw orientation commands are modified to follow the CoM within a bounded radius of

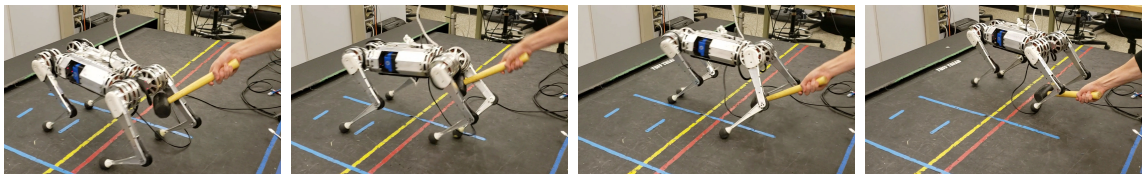


Figure 6-8: **Leg Push Disturbance.** Due to the fast replanning of the controller, it can handle disturbances to both the swing and stance legs. As the legs are moved around, the robot will plan forces using the new foot locations.

$0.1m$ and $10°$ respectively, rather than hold a global position and orientation.

It is worth noting that while a number of push disturbance tests were carried out successfully, the most common failure mode occurs when the legs are crossed and the knee joints lock together. Although the robot does not fall, it can cause the motors to overheat if they are exerting maximum torque continuously for a long period. This is not unexpected since the model provided to the RPC does not have constraints for leg collisions and assumes it can place the feet anywhere in its workspace and follows the nominal Bezier trajectory, regardless of the other leg configurations. A way to combat this is to add a negative quadratic cost at the other foot locations to repel optimal footsteps away from stepping too close to each other, or to design a more intelligent swing leg controller.

Pushes on the body are not the only form of disturbance that can happen. Possibly more catastrophically, the legs can be hit during their stance or swing phases. During stance, a disturbed leg may slip and change location with some velocity while consequently producing a force on the body. The unstable foothold may not be able to produce the required force needed and if it is in slip, it may further the problem by producing a lateral force outside the actual friction cone assumed by the constraint due to lower kinetic friction than static friction. In the case where the leg is hit during swing, the foot will be thrown off its planned trajectory and may touch down in a location that is not the intended step placement as well as exerting a force on the body. Again, since the gait patterns are generally $\geq 1Hz$ and the optimization runs at $> 80Hz$, the actual foot placement will be soon known to the controller and it can adjust the forces despite the feet being placed at sub optimal locations. Luckily, quadrupeds are not too sensitive to footstep placement and placing the feet a few cen-

timeters from the calculated optimal will be good enough for a temporary foothold as the leg will soon enter its swing phase and be able to readjust.

Figure 6-8 shows an example of both types of disturbances sequentially as the robot is commanded to trot in place. A rubber mallet is used to place an impulsive hit on the back right leg while it is in stance causing a slip. Then the mallet is used to quickly hit the back left leg while in swing, knocking it laterally off its intended trajectory. Within one gait cycle, the robot is able to recover to its nominal operating behavior and continues to trot in place.

6.3.4 Other Interesting Behaviors

The nominal RPC design is created to work for ideal conditions with flat ground and no disturbances. The assumptions made when formulating the optimization constraints and cost function hold this throughout locomotion as no external sensors provide information about the terrain or sense incoming objects that may disturb the state of the robot. However, due to the prediction horizon and a fast replan rate, the controller can robustly keep the robot from falling over under a variety of unexpected situations. The following behaviors were not considered when designing the controller or heuristics, but that may be encountered in use as a robotic first responder and still need to be handled by the robust controller.

Leg Orientation Flip

To simulate an occurrence where the robot has a leg unexpectedly constrained by a nearby object, a rope was tied around the back right leg of the Mini Cheetah and commanded to trot around a cluttered treadmill seen in Figure 6-9. The rope was pulled hard as the robot moved forward at $1\frac{m}{s}$. The leg was pulled inwards past the sagittal plane and through singularity. As it pulled the leg forward to take its next step, the orientation of the knee flipped and since the control model used by RPC has no concept of the physical leg, it continues trotting exactly the same as before. Despite the large disturbance, it can keep itself trotting, but unfortunately the leg

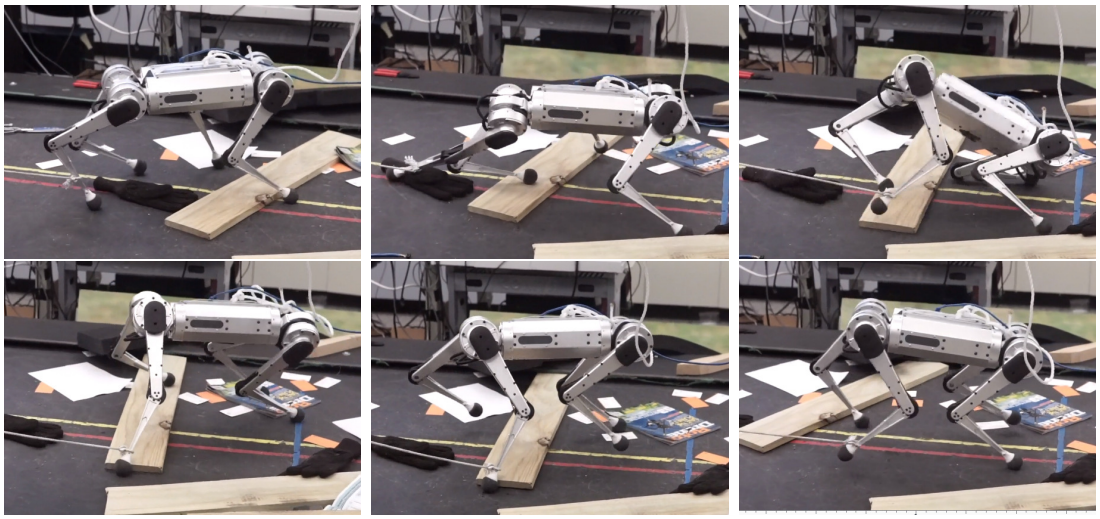


Figure 6-9: **Leg Orientation Flip.** As RPC does not care about the dynamics or kinematics of the legs, it works the same despite forcing the leg orientation to flip with a rope tied around the bottom link.

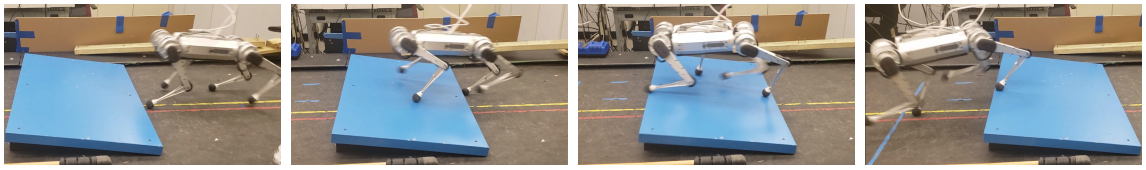


Figure 6-10: **Slippery Ramp.** While trotting at $1.5\frac{m}{s}$, the robot is able to easily walk over a small ramp with a slippery surface and climb off the back end.

manages to get itself pinned under the wooden plank. A couple steps are taken, with the leg constrained to the ground it is not able to reach the next foot location from the RPC solution and uses the actual location when in stance. When it finally frees its foot, the robot continues to operate as if normal conditions despite the flipped leg orientation.

Slippery Ramp

The current design does not feature any sensors on the foot for durability while walking around. Many commercial foot sensors are not reliable to handle the repeated impulses that occur in locomotion and although work is being done to design sensors that mitigate these effects while simultaneously measuring forces, surface normals, and slipping conditions [52], the current version does not have them. This means

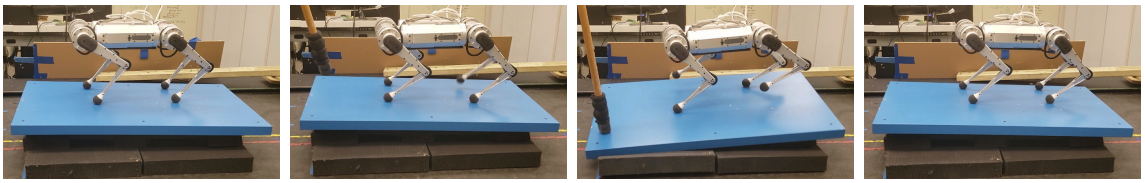


Figure 6-11: **Compliant Shaking Platform.** As the platform is rocked back and forth on soft foam, the robot takes steps to reposition itself and reduce the effects of the moving ground on the CoM.

that the value for the coefficient of friction with the surface has to be assumed, and is set for a value of $\mu = 0.5$. However, there is a high chance that a robot will encounter ground with s smaller coefficient of friction and with no adaptation to the coefficient online as in [39], the controller should not fail even with an incorrect surface characterization.

Mini Cheetah was commanded to trot forward at a constant velocity of $1.5\frac{m}{s}$ with a shallow slippery ramp as seen in Figure 6-10. The ramp has a surface with $\mu < 0.5$ and is resting on foam blocks that are compliant. The robot continues its path forward despite constant slipping on all of its legs as it climbs the ramp. The slipping causes disturbances on the body in the roll and yaw orientations as it walks over it. At the end of the ramp is a sudden, discrete drop of $0.12m$ which is roughly about half of the robot's nominal locomotion height. No major problems were faced when traversing the surface and the robot was successfully completed its commanded trajectory to the goal location.

Compliant Shaking Platform

If intended for locomotion in dangerous environments, it is likely that the ground it is walking or standing on may not be stable. During an earthquake, firm ground often shakes as if it were not fully solid. A small platform was placed on $0.2m$ of soft foam and repeatedly disturbed at various frequencies with both impulsive hits to the corners, as well as to the sides. Figure 6-11 depicts a sharp hit to the front left corner. The robot's CoM is disturbed, triggering the maximum standing velocity limit and it takes a step to reposition the CoM over the feet as the platform shakes.

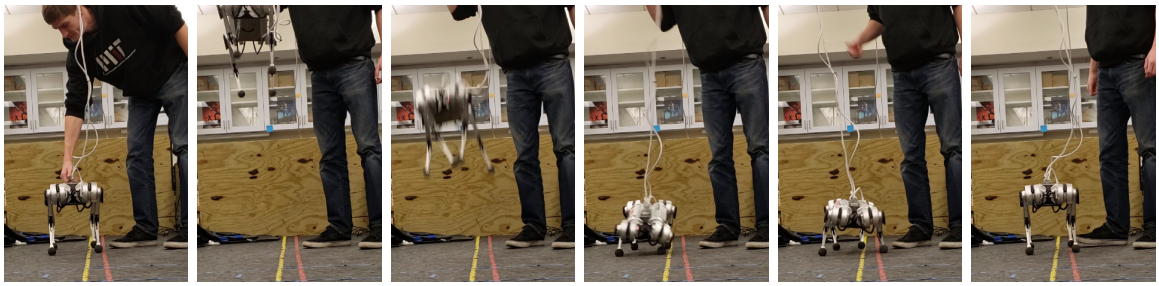


Figure 6-12: **High Drops.** Mini Cheetah is able to handle drops from over a meter in the air, which is more than 4 times the nominal body height of the robot.

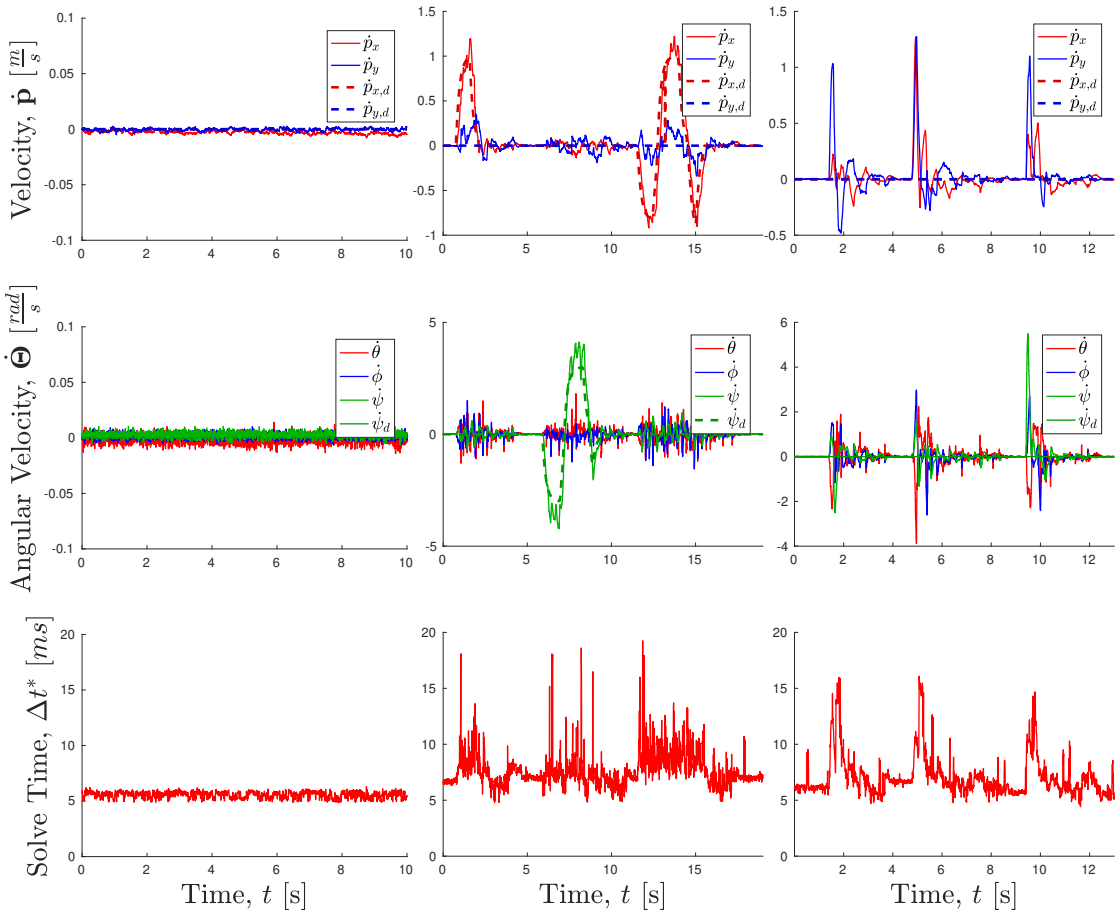
It quickly returns to standing once the shaking is finished.

High Drops

In a disaster scenario, it is possible that as the robot walks around, it may get to a place where it cannot simply walk down or where the unstable ground below it may collapse. The robot will have to endure a large drop and hopefully continue its mission. To simulate this, the robot is dropped from a CoM height slightly above $1m$ off the ground, meaning that at the point of impact, the CoM is traveling at $\sim 4.5\frac{m}{s}$. A maximum velocity threshold for the standing gait is triggered as the robot is dropped and it begins trotting to recover when it hits the ground. It is able to push on the ground enough to stop its downward momentum and within 2 steps, returns to a stable standing state as shown in the time lapse in Figure 6-12. The drop is over $4\times$ the nominal standing height of $0.25m$. Despite no work explicitly being done to design the RPC for drops or long, unexpected flight periods and with no external sensing of the environment to let it know where the ground is, the fast replanning frequency of the controller is able to deal with the situation as soon as the ground is reached.

6.4 Situationally-Varying Computation

Update rate and computation time are vital to the robustness of the algorithm. Figure 6-13 shows an example of three different situations where computation times vary



(a) Standing In Place (b) Commands (c) Disturbance

Figure 6-13: **Situational Solve Time.** The computation time of the RPC optimization varies depending on the state and desired commands. In more dynamic states where the robot is moving or facing a disturbance, the solver takes longer to converge, but is still running at a rate of over $100Hz$, well within the acceptable rate for stability.

with different characteristic properties. As a reference, the simple case of the robot standing in place with all four feet in contact and with no velocity commands or disturbances is used to create Figure 6-13a. The average solve time for the optimization over the short period where data was collected is found to be $5.5ms$, which corresponds to a controller solution frequency of about $180Hz$. However, when engaging in more dynamic behaviors when given a commanded velocity and turn rate during trotting as seen in Figure 6-13b, the computation times slow slightly. When beginning to move, the robot goes through a transient of acceleration resulting in spikes

where the optimization can take up to $20ms$ to solve for a short period. When the robot passes this transient and is trotting at speed, the RPC tends to run at $120Hz$, which is still $6\times$ faster than the minimum $20Hz$ rate that was found to be the limit of stability for the robot.

The most interesting cases are the ones where the robot faces large disturbances. Figure 6-13c is created from three distinct impulsive pushes to the robot. The first is a lateral push from the side, the second is a push from behind, and the third is a push on the hip motor causing a large rotation. In each case the robot is in a state that it has not previously seen and in which there is no clear heuristic found. Therefore it takes the solver longer to find a solution which stabilizes the robot back to standing as the solver is searching in a region of the cost space that is not well-shaped. At the worst case, the solver finds a solution in around $16ms$. However, unlike the transient spikes in the commanded velocity case, the slower computation times for disturbances are sustained longer until the state returns towards a comfortable region where heuristics help to bias the optimization towards a solution. Optimization convergence depends heavily on the initial guess and the desired trajectory, both of which are calculated from the heuristic functions. In situations where heuristics are well known, the solution frequency higher by anywhere from $1.5\times$ to $2\times$, which further highlights the importance of extracting good heuristics.

Chapter 7

Discussion and Implications

RPC was designed for general locomotion and to be robust under disturbances. It is clear that there is not one single method that is going to be the only solution for legged locomotion. A variety of other controllers have shown impressive capabilities in different situations. Some may be more applicable for different tasks. The controller was also formulated generally with the intention of being expandable and built upon. Besides the results presented, the possibilities for this regularized control philosophy have yet to be fully explored.

7.1 Extension to Other Robots

While the controller was developed for the MIT Cheetah 3 and Mini Cheetah robot platforms, it is easily generalizable to other systems. The philosophy of embedding analytical and data-driven heuristics as regularization is controller agnostic and can be added to any optimization-based controller using a similar framework to the one provided. RPC itself as presented here is designed specifically for optimizing footsteps and ground reaction forces for hybrid legged robots which makes the optimization nonlinear. However, it could easily be modified for continuous systems such as quadrotors with no need to optimize the location at which forces are applied.

The use of an external gait scheduler generalizes RPC in the current form to any number of legs provided that the gait schedule is feasible. A proof of concept

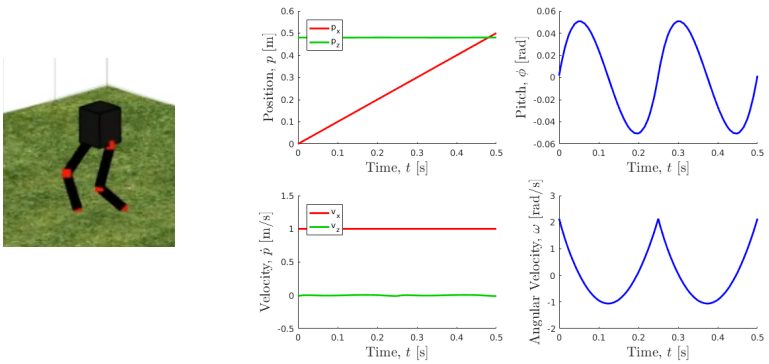


Figure 7-1: **Biped Walking.** State Trajectories for walking at 1m/s show stable forward locomotion while maintaining height and an periodic pitch oscillation.

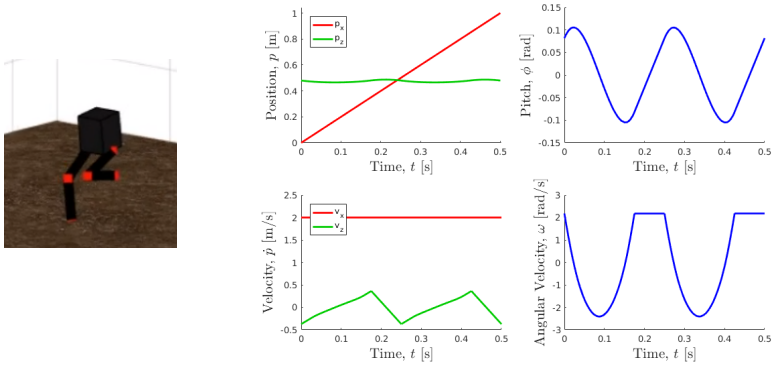


Figure 7-2: **Biped Running.** State Trajectories for running at 2m/s show stable forward locomotion with a short flight period between contact switches.

simulation was designed for a biped robot using RPC where the body is a single rigid body and the legs are massless as with the Cheetah platforms. A number of different gaits at different forward velocities were stabilized with no need to change the control framework other than the robot model passed in. The first gait considered was a simple walking gait. This features no flight phase and instantaneous contact state switching between opposing feet. It is the most common and least dynamic of the gaits and maintained a steady CoM height with constant forward speed. Abrupt changes in pitch rate seen in Figure 7-1 were due to the sudden force input during a new touchdown phase.

The next gait was the running gait with a slight flight period. During the flight phases between stances, no inputs from either feet can affect the motion of the CoM and therefore the system must compensate for this during the stance periods. Nat-

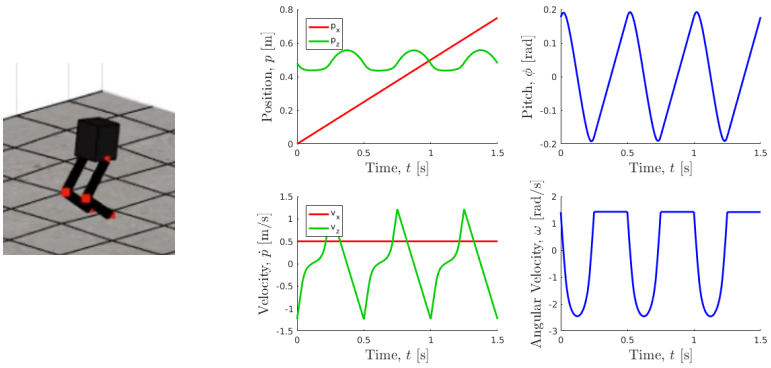


Figure 7-3: **Biped Hopping.** State Trajectories for hopping at 2m/s for a 6 phase prediction horizon show stable forward locomotion with a periodic fluctuation in height akin to what humans do when hopping.

urally, the prediction accounts for this. We also see in Figure 7-2 that there is a characteristic oscillatory height typical to running gaits with flight as the jumps between stance periods treat the CoM movement as a ballistic trajectory.

Finally, we observed a hopping gait in Figure 7-3. Similar to the running gait, there is a distinct flight phase between phases of double support from the feet. Since flight phases make up a larger part of the overall gait period, the CoM fluctuates much more. During the stance phases the robot dips down and generates forces to propel itself forward and up during the flight as humans do during jumping.

The controls problem for biped robots can be considered, in some ways, more difficult than that of quadrupeds due to the generally higher CoM, less legs available for contact, and the typically heavier limbs. Humanoids are an attractive solution to conduct tasks in a society built by humans as they would not need any special design to operate the same way as people, provided their hardware and control systems are capable of executing the motions. Despite a large amount of research, humanoid robots have generally not shown as many dynamic abilities as quadrupeds in recent years. While this proof of concept simulation was run on a reduced dynamics engine that uses a simplified robot and contact model, it serves as an encouraging result that RPC will work for a humanoid biped.

7.2 Implications

Many linear, convex controllers exist for legged robots that optimize ground reaction forces for tracking a desired trajectory and deal with foot placement in a separate controller. By optimizing the footsteps along with ground reaction forces, the robot is aware of the dynamic interaction between where the footsteps are located and the magnitude of the forces at those locations. Though it is more challenging to solve the resulting nonlinear optimization, the results are often better suited for robustness as it can exploit the location of footsteps to change the moment arm of the forces for more control over the torque applied to the CoM.

By including future states and footsteps, the controller knows that it will have a future contact mode and does not need to immediately correct for disturbances. For example, if the robot is pushed in a way where the current contact mode is unable to exert force to pitch the robot back to flat, but in a future contact mode it can place the feet in a favorable location to have control authority over that degree of freedom, it can sacrifice some cost at the current timestep in favor of a lower overall cost. This makes gaits with flight periods and longer periods of underactuation possible.

Furthermore, by embedding the previous optimal solutions and generalizing them to other cases through the heuristics, it was found that the performance of the robot improves without modification to the actual controller. A small set of previously unknown, simple heuristics was presented for common locomotion cases, but the framework makes it easy to discover more if they exist. Theoretically, if a set of heuristics, $\boldsymbol{\mathcal{H}}$, is found where $\boldsymbol{\chi}_R^* = \boldsymbol{\chi}_D^* = \boldsymbol{\chi}^*, \forall\{\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}} \in \boldsymbol{X}\}$ rather than approximate equation (3.1), then these heuristics are the solution to the optimization, $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{H}}^*$, for all of the viable operating conditions, $\boldsymbol{X}$. This then implies that there is no more need for heavy optimization as the optimal dynamics-based solution would always be equal to the heuristic solution. Finding $\boldsymbol{\mathcal{H}}^*$ is likely unfeasible, but the proposed method can be used to build the set $\boldsymbol{\mathcal{H}}$, and better approximate $\boldsymbol{\mathcal{H}}^*$ for a large number of cases. This will take the burden off of the optimization as the initial guess and regularized solution will be closer to the actual dynamics-based optimal solution.

While tailored for the task of legged locomotion and using RPC at the heart, the methods presented in Chapters 4 and 5 do not apply exclusively to the presented controller for locomotion. Using a similar methodical framework for extracting statistically correlated heuristics from data can be used to find variables between decision variables in any optimization. Similarly, the methods and algorithms used can improve the quality of any real-time nonlinear optimization solution. These are all general methods that were used together for the success of the RPC, but each individual part can be used in conjunction with other methods, that may not even be related to controls.

7.3 Future Work

The RPC framework developed in this work provides a powerful tool for control and planning of ground reaction forces and footsteps for legged robots during dynamic locomotion. However, it has several limitations. One of the more prominent is the lack of a higher level intelligent planner for navigating complex environments to accomplish tasks. The controller is intended as a generalized, mid level intelligent control planner for balance and locomotion. However, it was designed with the intent of integrating into a more complex hierarchical control architecture. For example, it takes in a commanded desired CoM state which can be input by any number of sources. This may be a prescribed trajectory, joystick commands, or an environment aware planning algorithm.

Similarly, the current gait scheduler uses a flexible time, phase-based gait pattern which results in semi-rigid stance and swing times. Although it is able to stabilize various regular and irregular gaits, the timing and adaptation of these must be designed by the operator. This is the current common method to define gait patterns. Recently, new results have shown promise in freeing these gait timings to create more natural gaits [11]. In the work presented by Boussema, a gait generator based on each leg's relative force generation capability shows naturally emergent gaits at different speeds. As RPC is designed to work with arbitrary gait patterns including flight and

variable leg contact combinations, these could be integrated into the controller with little modification.

Finally, as computation power increases for hardware systems and as heuristic models are built up, the control model used will be able to be extended for more accurate representations of the system. The controller behaves better with higher solve frequencies as it is able to replan footsteps and forces faster to reject disturbances. More powerful CPUs would also allow more complex dynamic models to be used. Adding leg masses, rotor inertias, and actuator friction may improve prediction fidelity and heuristic model accuracies resulting in more stable locomotion and better disturbance rejection. While it is shown that these are not necessary for good performance, better models should only serve to improve resulting solutions. In addition, preliminary results have shown that with orientation heuristics, $\boldsymbol{\mathcal{H}}_{\boldsymbol{\Theta}}(\boldsymbol{\chi})$, the full robot body rotation matrix, $\boldsymbol{R}(\boldsymbol{\Theta})$, can be used in place of the simplified yaw rotation matrix, $\boldsymbol{R}_z(\psi)$, in equation (3.6) for torque on the CoM as burden is taken off of the optimization.

7.4 Conclusion

Highly dynamic behaviors are demonstrated by the controller without the need to explicitly design or tune gains for each specific case. The same capabilities that many legged animals are able to flawlessly execute as discussed in Chapter 1 are shown alongside the analogous capabilities demonstrated by the RPC controlled robot hardware in Figure 7-4. While we have not quite achieved the same level of grace or magnitude of dynamic locomotion as animals in nature, the work presented represents an improvement in robot capabilities and a step towards creating robots that can robustly maneuver similarly to animals.

Results presented in this dissertation present a novel controller that exploits simple heuristic laws through regularization and the ability to use the framework for extracting new heuristic models. Results are shown in simulation as well as on the MIT Cheetah 3 and Mini Cheetah hardware. The work signifies an improvement on

(a) High Speed Turning (b) Rough Terrain (c) Disturbance Rejection

Figure 7-4: **Improved Robot Capabilities.** With the RPC framework, the robot is now able to take fast dynamic turns, blindly traverse rough terrain, and stabilize itself after large pushes. This represents a step closer to creating robots that can execute dynamic maneuvers as proficiently as legged animals.

the current capabilities of blind robotic controllers through a general control framework which robustly handles large unexpected disturbances. Translational and rotational locomotion, various parametrized gaits, sudden and sustained disturbances, and transferability between similar robots are all able to be performed without any changes to the control framework or gains. This generalizability and robustness demonstrates the power of the novel controller and the heuristic extraction framework for robust dynamic legged locomotion.

Appendix A

Nomenclature

Coordinate Frames

$\mathcal{I}$: Inertial World Frame

$\mathcal{B}$: Robot Body Frame

${}^{\mathcal{I}}(\cdot)$: Quantity $(\cdot)$ in the $\mathcal{I}$

${}^{\mathcal{B}}(\cdot)$: Quantity $(\cdot)$ in the $\mathcal{B}$

Robot Physical Parameters

m : Robot Mass

$\boldsymbol{I}$: Robot Inertia

${}^{\mathcal{B}}\boldsymbol{r}_h$: Hip Position

l_1 : Upper Leg Link

l_2 : Lower Leg Link

$\boldsymbol{q}$: Joint Angles

Robot States

θ : roll

ϕ : pitch

ψ : yaw

$\boldsymbol{p} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$: CoM Position

$\boldsymbol{\Theta} = \begin{bmatrix} \theta & \phi & \psi \end{bmatrix}^T$: RPY Orientation

$\mathbf{q}$: Quaternion

$\dot{\boldsymbol{p}}$: CoM velocity

$\dot{\boldsymbol{\Theta}}$: RPY Rates

${}^{\mathcal{I}}\boldsymbol{\omega}$: World Angular Rates

${}^{\mathcal{B}}\boldsymbol{\omega}$: Body Angular Rates

$\boldsymbol{p}_i$: Foot i Position

FR : Front Right Foot

FL : Front Left Foot

BR : Back Right Foot

BL : Back Left Foot

Gait Parameters

$\boldsymbol{\Phi}$: Scheduled Gait Phase

$\boldsymbol{s}_\Phi$: Scheduled Contact State

$\boldsymbol{\Phi}_{c \to \bar{c}}$: Switching Phase

T_P : Gait Cycle Period

T_S : Swing Period

ξ_Φ : Stance Fraction

RPC Variables

$\boldsymbol{f}_i$: Ground Reaction Forces at foot i

$\boldsymbol{r}_i$: Foot Position i Relative to CoM

$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^T & \boldsymbol{\Theta}^T & \dot{\boldsymbol{p}}^T & \dot{\boldsymbol{\Theta}}^T \end{bmatrix}^T$: Robot States

$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{r}_1^T & \boldsymbol{f}_1^T & \dots & \boldsymbol{r}_4^T & \boldsymbol{f}_4^T \end{bmatrix}^T$: Robot Inputs

$\boldsymbol{Q}$: State Weights

$\boldsymbol{R}$: Input Weights

$\boldsymbol{W}$: Decision Variable Weights

$\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{x}^T & \boldsymbol{u}^T \end{bmatrix}^T$: RPC Decision Variables

$\boldsymbol{x}_d = \begin{bmatrix} \dot{p}_x & \dot{p}_y & \dot{\psi} \end{bmatrix}^T$: Desired Robot States

N : Number of Timesteps

Δt^* : Optimization Timestep

$\bar{dt}$: Averaged Solve Time

$\boldsymbol{\zeta}$: Timestep Constraints

$\boldsymbol{\zeta}'$: Timestep Linking Constraints

$\mathbb{J}_{\boldsymbol{\zeta}}(\boldsymbol{\chi})$: Constraint Jacobian

$\mathbb{H}_{\mathcal{L}}(\boldsymbol{\chi}, \boldsymbol{\zeta}(\boldsymbol{\chi}))$: Hessian of the Lagrangian

n : Number of Decision Variables

m : Number of Constraints

Heuristic Extraction

$\boldsymbol{V}$: Candidate Variables

$\mathcal{H}_{v_i}(v_j)$: Heuristic Function for v_i Dependent on v_j

$a_n^{v_i}$: Extracted Polynomial Coefficient Degree d

$b_n^{v_i}$: Extracted Sine Amplitude n

$c_n^{v_i}$: Extracted Sine Frequency n

$d_n^{v_i}$: Extracted Sine Offset n

$\boldsymbol{\mathcal{H}_x}(\boldsymbol{\chi}, \Phi, \boldsymbol{x}_d)$: Heuristics for Robot States

$\boldsymbol{\mathcal{H}_r}(\boldsymbol{\chi}, \Phi, \boldsymbol{x}_d)$: Heuristics for Foot Placements

$\boldsymbol{\mathcal{H}_f}(\boldsymbol{\chi}, \Phi, \boldsymbol{x}_d)$: Heuristics for Ground Reaction Forces

$\boldsymbol{\mathcal{H}_\chi}(\boldsymbol{\chi}, \Phi, \boldsymbol{x}_d)$: Heuristics for All Decision Variables

Appendix B

Regularized Predictive Control

$$\min_{\boldsymbol{\chi}} \qquad J(\boldsymbol{\chi}) = \sum_{k=0}^{N-1} (\boldsymbol{\mathcal{H}}_{\boldsymbol{\chi}}(\boldsymbol{\chi}_k, \boldsymbol{\Phi}_k, \boldsymbol{x}_{d,k}) - \boldsymbol{\chi}_k)^T \boldsymbol{W}_k (\boldsymbol{\mathcal{H}}_{\boldsymbol{\chi}}(\boldsymbol{\chi}_k, \boldsymbol{\Phi}_k, \boldsymbol{x}_{d,k}) - \boldsymbol{\chi}_k)$$

subject to

Simplified Discrete Dynamics

$$\boldsymbol{x}_{k+1} - (\boldsymbol{A}(\Delta t_k)\boldsymbol{x}_k + \boldsymbol{B}(\Delta t_k) h(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\Phi}_k) + \boldsymbol{d}(\Delta t_k)) = 0$$

Foot Placed on Ground

$$\boldsymbol{s}_{\Phi,i,k} \left(\hat{z}_g(p_{x,i,k}, p_{y,i,k}) - p_{z,i,k}\right) = 0$$

Kinematic Leg Limits

$$\boldsymbol{s}_{\Phi,i,k} \left(\left\| \boldsymbol{r}_{i,k} - {}^{\mathcal{B}}\boldsymbol{r}_{h,i} \right\| - p_{z,d,k} \tan(\beta) \right) \leq 0$$

Foot Stationary During Stance

$$\boldsymbol{s}_{\Phi,i,k+1} \boldsymbol{s}_{\Phi,i,k} \left(\boldsymbol{p}_{i,k+1} - \boldsymbol{p}_{i,k}\right) = 0$$

Positive Ground Force Normal

$$-\boldsymbol{s}_{\Phi,i,k} \left(\boldsymbol{f}_{i,k} \cdot \nabla \hat{\mathbb{G}}(\boldsymbol{p}_{i,k}) \right) \leq 0$$

Lateral Force Friction Pyramids

$$-\mu \boldsymbol{f}_{z,i} \leq \boldsymbol{f}_{x,i} \leq \mu \boldsymbol{f}_{z,i}$$
$$-\mu \boldsymbol{f}_{z,i} \leq \boldsymbol{f}_{y,i} \leq \mu \boldsymbol{f}_{z,i}$$

Max and Min Bounds

$$\boldsymbol{\chi}_{min} \leq \boldsymbol{\chi} \leq \boldsymbol{\chi}_{max}.$$

Appendix C

All Locomotion Heuristics

Table C.1: Analytic Locomotion Heuristics

Heuristic	Heuristic Model	Dimensions
Hip Centered Stepping	$\boldsymbol{\mathcal{H}_r}(\boldsymbol{\Theta}) = \boldsymbol{P}^T\boldsymbol{P}(\boldsymbol{R}(\boldsymbol{\Theta})\boldsymbol{r}_h)$	$\mathbb{R}^{3\times1}$
Capture Point	$\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}}, \dot{\boldsymbol{p}}_d) = \boldsymbol{P}^T\boldsymbol{P}\left(\sqrt{\frac{p_z}{\|\boldsymbol{g}\|}}(\dot{\boldsymbol{p}} - \dot{\boldsymbol{p}}_d)\right)$	$\mathbb{R}^{3\times1}$
Impulse Scaling	$\boldsymbol{\mathcal{H}_f}(\boldsymbol{\Phi}) = \frac{m\boldsymbol{g}\xi_\Phi}{\sum_{i=0}^{N_f} \boldsymbol{s}_{\Phi,i}}$	$\mathbb{R}^{3\times1}$
Centripetal Acceleration	$\boldsymbol{\mathcal{H}_f}(\dot{\boldsymbol{\Theta}} \times \dot{\boldsymbol{p}}) = m\dot{\boldsymbol{\Theta}} \times \dot{\boldsymbol{p}}$	$\mathbb{R}^{3\times1}$

Table C.2: Extracted Locomotion Heuristics

Heuristic	Heuristic Model	Dimensions
Translational Stepping	$\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}}) = \boldsymbol{a}_1^{\boldsymbol{r}}\dot{\boldsymbol{p}} + \boldsymbol{a}_0^{\boldsymbol{r}}$	$\mathbb{R}^{3\times1}$
In-Place Turning	$\boldsymbol{\mathcal{H}_r}(\dot{\psi}) = \boldsymbol{a}_1^{\boldsymbol{r}}\dot{\psi} + \boldsymbol{a}_0^{\boldsymbol{r}}$	$\mathbb{R}^{3\times1}$
High Speed Turning	$\boldsymbol{\mathcal{H}_r}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}}) = \boldsymbol{a}_1^{\boldsymbol{r}}(\dot{\boldsymbol{p}} \times \dot{\boldsymbol{\Theta}}) + \boldsymbol{a}_0^{\boldsymbol{r}}$	$\mathbb{R}^{3\times1}$
Orientation Compensation	$\boldsymbol{\mathcal{H}_\Theta}(\dot{\boldsymbol{p}}) = \boldsymbol{a}_1^{\boldsymbol{\Theta}}\dot{\boldsymbol{p}} + \boldsymbol{a}_0^{\boldsymbol{\Theta}}$	$\mathbb{R}^{2\times1}$
Periodic Orientation	$\boldsymbol{\mathcal{H}_\Theta}(\boldsymbol{\Phi}) = \boldsymbol{b}_1^{\boldsymbol{\Theta}}\sin(\boldsymbol{c}_1^{\boldsymbol{\Theta}}\boldsymbol{\Phi} + \boldsymbol{d}_1^{\boldsymbol{\Theta}})$	$\mathbb{R}^{2\times1}$
Height Compensation	$\mathcal{H}_z(\dot{p}_x) = a_2^z\dot{p}_x^2 + a_1^z\dot{p}_x + a_0^z$	$\mathbb{R}^{1\times1}$

Appendix D

Key Related Publications

A list of several key papers by the MIT Bimimetic Robotics Lab that provide additional results and disscussion about sections of this dissertation:

[3] Gerardo Bledt. Policy regularized model predictive control for robust legged locomotion. Master's thesis, Massachusetts Institute of Technology, February 2018.

[4] Gerardo Bledt and Sangbae Kim. Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, China, Nov. 2019.

[5] Gerardo Bledt and Sangbae Kim. Extracting legged locomotion heuristics with regularized predidictive control. In 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, June. 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 Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, Oct. 2018.

[7] Gerardo Bledt, Patrick M. Wensing, Sam Ingersoll, and Sangbae Kim. Contact model fusion for event-based locomotion in unstructured terrains. In 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, May 2018. **Finalist Best Student Paper & Best Overall Conference Paper**.

[8] Gerardo Bledt, Patrick M. Wensing, and Sangbae Kim. Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, Canada, Sept. 2017.

[12] 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 Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, Oct. 2018.

[41] B. Katz, J. D. Carlo, and S. 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, May 2019.

[42] Benjamin G. Katz. A low cost modular actuator for dynamic robots. Master's thesis, Massachusetts Institute of Technology, June 2018.

[11] C. Boussema, M. J. Powell, G. Bledt, A. J. Ijspeert, P. M. Wensing, and S. Kim. Online gait transitions and disturbance recovery for legged robots via the feasible impulse set. IEEE Robotics and Automation Letters, 4(2):1611âĂŞ1618, April 2019.

[54] MIT Cheetah Software. https://github.com/mit-biomimetics/cheetah-software.

Bibliography

[1] Taylor Apgar, Patrick Clary, Kevin Green, Alan Fern, and Jonathan W. Hurst. Fast online trajectory optimization for the bipedal robot cassie. In *Robotics: Science and Systems*, 2018.

[2] Peter Bhlmann and Sara van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[3] Gerardo Bledt. Policy regularized model predictive control for robust legged locomotion. Master's thesis, Massachusetts Institute of Technology, February 2018.

[4] Gerardo Bledt and Sangbae Kim. Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Macau, China, Nov. 2019.

[5] Gerardo Bledt and Sangbae Kim. Extracting legged locomotion heuristics with regularized predidictive control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, June. 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 *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, Oct. 2018.

[7] Gerardo Bledt, Patrick M. Wensing, Sam Ingersoll, and Sangbae Kim. Contact model fusion for event-based locomotion in unstructured terrains. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 2018.

[8] Gerardo Bledt, Patrick M. Wensing, and Sangbae Kim. Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, Canada, Sept. 2017.

[9] R. Blickhan. The spring-mass model for running and hopping. *Journal of Biomechanics*, 22(11):1217 – 1227, 1989.

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

[11] C. Boussema, M. J. Powell, G. Bledt, A. J. Ijspeert, P. M. Wensing, and S. Kim. Online gait transitions and disturbance recovery for legged robots via the feasible impulse set. *IEEE Robotics and Automation Letters*, 4(2):1611–1618, April 2019.

[12] 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 *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, Oct. 2018.

[13] The Qt Company. Cross-platform software development for embedded and desktop.

[14] H. Dai and R. Tedrake. Optimizing robust limit cycles for legged locomotion on unknown terrain. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1207–1213, Dec 2012.

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

[16] Y. Ding, A. Pandala, and H. 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, May 2019.

[17] J. Englsberger, P. Kozlowski, and C. Ott. Biologically inspired deadbeat control for running on 3d stepping stones. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 1067–1074, Nov 2015.

[18] Haegwang Eom, Byungkuk Choi, and Junyong Noh. Data-driven reconstruction of human locomotion using a single smartphone. *Comput. Graph. Forum*, 33(7):11âĂŞ19, October 2014.

[19] M. F. Fallon, M. Antone, N. Roy, and S. Teller. Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 112–119, Nov 2014.

[20] N. Fazeli, M. Oller, J. Wu, Z. Wu, J. B. Tenenbaum, and A. Rodriguez. See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion. *Science Robotics*, 4(26), 2019.

[21] R. Featherstone. A beginner's guide to 6-d vectors (part 1). *IEEE Robotics Automation Magazine*, 17(3):83–94, Sep. 2010.

[22] R. Featherstone. A beginner's guide to 6-d vectors (part 2) [tutorial]. *IEEE Robotics Automation Magazine*, 17(4):88–99, Dec 2010.

[23] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2007.

[24] Roy Featherstone and David E. Orin. *Dynamics*, pages 35–65. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[25] Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.

[26] Michele Focchi, Andrea del Prete, Ioannis Havoutis, Roy Featherstone, Darwin G. Caldwell, and Claudio Semini. High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots*, 41(1):259–272, Jan 2017.

[27] Michele Focchi, Romeo Orsolino, Marco Camurri, Victor Barasuol, Carlos Mastalli, Darwin G. Caldwell, and Claudio Semini. Heuristic planning for rough terrain locomotion in presence of external disturbances and variable perception quality. *CoRR*, abs/1805.10238, 2018.

[28] R. J. Full and D. E. Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *J. Exp. Biol.*, 202(23):3325–3332, 1999.

[29] Patrick GelÃ§, Stefan Klus, Jens Eisert, and Christof SchÃijtte. Multidimensional approximation of nonlinear dynamical systems. *Journal of Computational and Nonlinear Dynamics*, 14(6), Apr 2019.

[30] Mingyang Geng, Xing Zhou, Bo Ding, Huaimin Wang, and Lei Zhang. Learning to cooperate in decentralized multi-robot exploration of dynamic environments. In *International Conference on Neural Information Processing*, pages 40–51. Springer, 2018.

[31] Eric Douglas Grant. *Constraint-Based Design by Cost Function Optimization*. PhD thesis, USA, 1992. UMI Order No. GAX91-35262.

[32] R. J. Griffin and A. Leonessa. Model predictive control for dynamic footstep adjustment using the divergent component of motion. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1763–1768, May 2016.

[33] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[34] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017.

[35] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Bloesch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer. Anymal - toward legged robots for harsh environments. *Advanced Robotics*, 31(17):918–931, 2017.

[36] J. Hwangbo, J. Lee, and M. Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, April 2018.

[37] 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.

[38] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519*, 2017.

[39] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter. Dynamic locomotion on slippery ground. *IEEE Robotics and Automation Letters*, 4(4):4170–4176, Oct 2019.

[40] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart. Fast nonlinear model predictive control for multicopter attitude tracking on so(3). In *2015 IEEE Conference on Control Applications (CCA)*, pages 1160–1166, Sep. 2015.

[41] B. Katz, J. D. Carlo, and S. 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, May 2019.

[42] Benjamin G. Katz. A low cost modular actuator for dynamic robots. Master's thesis, Massachusetts Institute of Technology, June 2018.

[43] Mitsuo Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718 – 727, 1999.

[44] Donghyun Kim, Jaemin Lee, Orion Campbell, Hochul Hwang, and Luis Sentis. Computationally-robust and efficient prioritized whole-body controller with contact constraints, 2018.

[45] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard. Whole-body model-predictive control applied to the hrp-2 humanoid. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 3346–3351, Sept 2015.

[46] Oliver Kroemer, Scott Niekum, and George Dimitri Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *CoRR*, abs/1907.03146, 2019.

[47] Scott Kuindersma, Frank Permenter, and Russ Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. *CoRR*, abs/1311.1839, 2013.

[48] J. Lee, H. Dallali, M. Jin, D. Caldwell, and N. Tsagarakis. Robust and adaptive whole-body controller for humanoids with multiple tasks under uncertain disturbances. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5683–5689, May 2016.

[49] Mingmin Liu, Daokui Qu, Fang Xu, Fengshan Zou, Pei Di, and Chong Tang. Quadrupedal robots whole-body motion control based on centroidal momentum dynamics. *Applied Sciences*, 9(7):1335, Mar 2019.

[50] Robert Mahony, Tarek Hamel, and Jean Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, 2008.

[51] Carlos Mastalli, Ioannis Havoutis, Michele G Focchi, Darwin G Caldwell, and Claudio G Semini. Motion planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control. working paper or preprint, October 2018.

[52] Donghyun Kim Juan Romero Meng Yee (Michael) Chuah, Lindsay Epstein and Sangbae Kim. Bi-modal hemispherical sensor: A unifying solution for three axis force and contact angle measurement. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Macau, China, Nov. 2019.

[53] Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *CoRR*, abs/1707.02201, 2017.

[54] MIT Biomimetics Lab. Cheetah software. https://github.com/mit-biomimetics/cheetah-software.

[55] ConcepciÃşn A. Monje, Santiago Martinez, Paolo Pierro, and Carlos Balaguer. Whole-body balance control of a humanoid robot in real time based on zmp stability regions approach. *Cybernetics and Systems*, 49(7-8):521–537, 2018.

[56] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli. Fast nonlinear model predictive control for unified trajectory optimization and tracking. In *IEEE Int. Conf. on Rob. and Automation*, pages 1398–1404, May 2016.

[57] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli. Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds. *ArXiv e-prints*, July 2016.

[58] Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine Dario Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *CoRR*, abs/1712.02889, 2017.

[59] H. W. Park, Sangin Park, and S. 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, May 2015.

[60] Hae-Won Park, Patrick Wensing, and Sangbae Kim. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[61] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. High-speed bounding with the MIT Cheetah 2: Control design and experiments. *International Journal of Robotics Research*, 36(2):167–192, 2017.

[62] Delyle T. Polet and John E. A. Bertram. A simple model of a quadruped discovers single-foot walking and trotting as energy optimal strategies. *bioRxiv*, 2019.

[63] I. Poulakakis and J. W. Grizzle. The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. *IEEE Transactions on Automatic Control*, 54(8):1779–1793, Aug 2009.

[64] M. J. Powell, E. A. Cousineau, and A. D. Ames. Model predictive control of underactuated bipedal robotic walking. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 5121–5126, May 2015.

[65] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 200–207, Genova, Italy, Dec. 2006.

[66] Mehdi Rahimi, Spencer Gibb, Yantao Shen, and Hung Manh La. A comparison of various approaches to reinforcement learning algorithms for multi-robot box pushing. In *International Conference on Engineering Research and Applications*, pages 16–30. Springer, 2018.

[67] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, Rob Playter, and the BigDog Team. Bigdog, the rough-terrain quaduped robot. In *Proceedings of the 17th World Congress*, pages 10822–10825, 2008.

[68] Marc H. Raibert. *Legged robots that balance*. MIT Press, Cambridge, MA, USA, 1986.

[69] J. R. Rebula, S. Schaal, J. Finley, and L. Righetti. A robustness analysis of inverse optimal control of bipedal walking. *IEEE Robotics and Automation Letters*, 4(4):4531–4538, Oct 2019.

[70] Shai Revzen and Matthew Kvalheim. *Data driven models of legged locomotion*, volume 9467 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 94671V. 2015.

[71] Robert Ringrose. Automatically tuning control systems for simulated legged robots. In *AAAI 1994*, 1994.

[72] M. Rutschmann, B. Satzinger, M. Byl, and K. Byl. Nonlinear model predictive control for rough-terrain robot hopping. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1859–1864, Oct. 2012.

[73] C. Semini, V. Barasuol, J. Goldsmith, M. Frigerio, M. Focchi, Y. Gao, and D. G. Caldwell. Design of the hydraulically actuated, torque-controlled quadruped robot hyq2max. *IEEE/ASME Transactions on Mechatronics*, 22(2):635–646, April 2017.

[74] L. Sentis and O. 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, May 2006.

[75] B. J. Stephens and C. G. Atkeson. Push recovery by stepping for humanoid robots with force controlled joints. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 52–59, Dec 2010.

[76] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[77] 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. *CoRR*, abs/1804.10332, 2018.

[78] María Tomás-Rodríguez and Stephen P. Banks. *Linear Approximations to Nonlinear Dynamical Systems*, pages 11–28. Springer London, London, 2010.

[79] Vassilios Tsounis, Mitja Alge, Joonho Lee, Farbod Farshidian, and Marco Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning, 2019.

[80] Ismail Uyanik. Identification of legged locomotion via model-based and data-driven approaches, 2017.

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

[82] Kenneth Waldron and James Schmiedeler. Chapter 1: Kinematics. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*. Springer, New York, 2008.

[83] Yang Wang and S. Boyd. Fast model predictive control using online optimization. *Control Systems Technology, IEEE Transactions on*, 18(2):267–278, March 2010.

[84] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. 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, June 2017.

[85] Patrick M. Wensing and David E. Orin. High-speed humanoid running through control with a 3D-SLIP model. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Rob. and Sys.*, pages 5134–5140, Tokyo, Japan, Nov. 2013.

[86] Alexander W. Winkler, Farbod Farshidian, Michael Neunert, Diego Pardo, and Jonas Buchli. Online walking motion and foothold optimization for quadruped locomotion. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5308–5313, 2017.

[87] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots, 2019.

[88] Kyongmin Yeo. Data-driven reconstruction of nonlinear dynamics from sparse observation. *Journal of Computational Physics*, 395:671âĂŞ689, Oct 2019.