Title
Control Implementation of Dynamic Locomotion on Compliant, Underactuated, Force-Controlled Legged Robots with Non-Anthropomorphic Design

Author
Yu, Jeffrey Chen

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Control Implementation of Dynamic Locomotion on Compliant, Underactuated, Force-Controlled Legged Robots with Non-Anthropomorphic Design

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mechanical Engineering

by

Jeffrey Chen Yu

2020

ABSTRACT OF THE DISSERTATION

Control Implementation of Dynamic Locomotion on Compliant, Underactuated, Force-Controlled Legged Robots with Non-Anthropomorphic Design

by

Jeffrey Chen Yu

Doctor of Philosophy in Mechanical Engineering

University of California, Los Angeles, 2020

Professor Dennis Hong, Chair

The control of locomotion on legged robots traditionally involves a robot that takes a standard legged form, such as the anthropomorphic humanoid, the dog-like quadruped, or the bird-like biped. Additionally, these systems will often be actuated with position-controlled servos or series-elastic actuators that are connected through rigid links. This work investigates the control implementation of dynamic, force-controlled locomotion on a family of legged systems that significantly deviate from these classic paradigms by incorporating modern, state-of-the-art proprioceptive actuators on uniquely configured compliant legs that do not closely resemble those found in nature. The results of this work can be used to better inform how to implement controllers on legged systems without stiff, position-controlled actuators, and also provide insight on how intelligently designed mechanical features can potentially simplify the control of complex, nonlinear dynamical systems like legged robots. To this end, this work presents the approach to control for a family of non-anthropomorphic

bipedal robotic systems which are developed both in simulation and with physical hardware. The first is the Non-Anthropomorphic Biped, Version 1 (NABi-1) that features position-controlled joints along with a compliant foot element on a minimally actuated leg, and is controlled using simple open-loop trajectories based on the Zero Moment Point. The second system is the second version of the non-anthropomorphic biped (NABi-2) which utilizes the proprioceptive Back-drivable Electromagnetic Actuator for Robotics (BEAR) modules for actuation and fully realizes feedback-based force controlled locomotion. These systems are used to highlight both the strengths and weaknesses of utilizing proprioceptive actuation in systems, and suggest the tradeoffs that are made when using force control for dynamic locomotion. These systems also present case studies for different approaches to system design when it comes to bipedal legged robots.

The dissertation of Jeffrey Chen Yu is approved.

Veronica Santos

Tsu-Chin Tsao

Ankur Mehta

Dennis Hong, Committee Chair

University of California, Los Angeles

2020

To my friends, family, and robots.

Table of Contents

List of Figures

Acknowledgments

Thank you to:

Dr. Dennis Hong, for providing guidance, support, and inspiration in my time at RoMeLa at UCLA, and for helping me become a better roboticist, robotics communicator, and just all round better person in general.

My labmates and co-authors: Joshua Hooks, Tym Zhu, Minsung Ahn, Xiaoguang Zhang, Sepehr Ghassemi, Hosik Chae, and everyone else in RoMeLa for building, repairing, letting me program, and repairing again all the sweet robots in the lab! I learned so much from everyone in the lab and could not have succeeded without everyone's support; I'm thankful to be able to call you all my friends.

Matt Haberland, for showing me the ways of trajectory optimization and giving me some perspective on academia and life.

Dr. Veronica Santos, for letting me TA her class when I was strapped for cash, and for her general support and advice on robots, fundraising, and management.

My PhD Committee, for taking the time to listen to my journey and guide me along the way.

All my professors and mentors at UCLA and Johns Hopkins for teaching me the ins and outs of engineering, academia, and life!

The UCLA bboy squad, for always being there for me and giving me a spot to chill out and not think about robots if I didn't want to.

Formula Drone at UCLA, for pushing me to compete in the AlphaPilot Challenge which turned out to be an amazing learning experience.

The AlphaPilot squad: Ablai A., Alim S., Luigi P., and everyone else for sticking it out with me on the wild ride that was the 2019 AIRR season.

And of course, my friends (west coast to east coast, NorCal to the SoCal) and family, especially mom, dad, and Vicky for always being there for me.

Curriculum Vitae

2010–2014	B.S. in Mechanical Engineering, Minor in Robotics, Johns Hopkins University.
2014-2015	M.S. in Mechanical Engineering, University of California, Los Angeles.

Publications

Yu, J., Hooks, J., Ghassemi, S., Pogue, A., & Hong, D. (2016, August). Investigation of a non-anthropomorphic bipedal robot with stability, agility, and simplicity. InUbiquitous Robots and Ambient Intelligence (URAI), 2016 13th International Conference on(pp. 11-15). IEEE.

Ghassemi, S., Yu, J., Hooks, J., & Hong, D. (2016, November). Feasibility study of a novel biped NABiRoS: Non anthropomorphic bipedal robotic system. InHumanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on(pp. 145-145). IEEE.

Yu, J., Hooks, J., Ghassemi, S., & Hong, D. (2017, August). Exploration of Turning Strategies for an Unconventional Non-Anthropomorphic Bipedal Robot. InASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference(pp. V05BT08A021-V05BT08A021). American Society of Mechanical

Engineers.

Yu, J., Hong, D., & Haberland, M. (2018, October). Energetic efficiency of a compositional controller on a monoped with an articulated leg and SLIP dynamics. In Intelligent Robots and Systems (IROS), 2018, International Conference on. IEEE.

Yu, J., Hooks, J., Zhang, X., Ahn, M., & Hong, D. (2018 November). A proprioceptive, force-controlled, non-anthropomorphic biped for dynamic locomotion. In Humanoid Robots (Humanoids), 2018 IEEE-RAS 19th International Conference on. IEEE.

CHAPTER 1

Introduction

1.1 Motivation

Legged robots and automata that mimic the form and function of humans and animals have captivated the imaginations of humanity for centuries. Importantly, these systems have the ability to perform work in a variety of environments when wheeled platforms, aerial systems, or biological workers are not the best solution. Today, legged robots are more capable than ever before, with videos of humanoid robots doing gymnastics and quadruped robots dancing to the latest viral craze becoming more and more common. However, these videos typically do not highlight the fact that legged locomotion remains an area of active research, and the videos that are presented to the public may or may not be representative of the typical performance of the system. In most practical measures, many of these machines still lack the agility, efficiency, and robustness of their biological counterparts. In reality, legged locomotion is a deceptively difficult problem to solve because, even though most humans and animals perform legged locomotion every day, the act of simply walking already involves a series of highly coordinated feedback-controlled motions with periodic impact events and switching contact conditions.

Many of the current deficiencies in locomotion are a byproduct of traditional robot design, both mechanically and in control approach. That is, traditionally, legged systems were realized with stiff, fully-actuated limbs, which are amenable to the classical position-based control approaches utilized on robot manipulator arms. In essence, the earliest realization of a legged robot was a pair of manipulator arms bolted together at the base. However, examining natural occurrences of legged locomotion and interaction with unstructured external environments suggest this rigid approach to design and control (figuratively and literally) may not be the most ideal. It is the ability to apply force to and comply with ones surroundings rather than the ability to precisely position an end-effector that is more desirable for behaviors like walking over uneven terrain. Technology such as series elastic actuators (SEA) and the recent proprioceptive actuators are some prominent tools for roboticists trying to achieve such behaviors, but also introduce a new set of challenges around controlling these devices to act as desired.

The control of a legged system with some sort of compliant actuation is, somewhat surprisingly, one of the earliest problems in legged robotics, and there is a considerable amount of prior work done to generate locomotion on such platforms. Still, there is not yet a one size fits all solution to the control problem, as typically the control solution is tightly coupled to the mechanical design of the robotic system. The most popular legged robot designs today are the anthropomorphic, two-legged *humanoid*, and the dog-like, four-legged *quadruped*. However, there is still significant variance in the design and form factor of these systems, and furthermore there is not yet a standardized legged platform design that incorporates compliance. Compliance typically appears in the design of legged robots in the

form of mechanical springs, implemented at the actuator level or on the legs of the system themselves. These springy elements inevitably complicate the (already nonlinear) dynamics of the systems, leading to issues like underactuation and a potentially large number of states in the system model that invite the curse of dimensionality and lead to computationally intensive controllers that are difficult to actualize, even on modern silicon.

Fortunately, not all hope is lost, as legged robots have been around since even before Moore's Law had really started taking effect. Accordingly, A classic approach to the control of legged locomotion is to formulate a simplified dynamic model ('template') and then design (often heuristically) an easily tune-able controller around it. This has worked amazingly well for a large number of legged robots, but is again dependent on the template model dynamics at least somewhat resembling the real system dynamics. Today, with modern advances in computing power and efficiency, more computationally intensive optimization based methods (that iteratively solve for solutions) for control have started to become more and more popular for running on real hardware, given the optimization problem remains tractable. Alternatively, optimizations can be run offline to generate trajectories for locomotion when the number of states and constraints makes it infeasible to run the optimizations online. Optimization based methods (both linear and non-linear) offer the possibility of generating 'optimal' motions which minimize the energy consumed by the actuators or the time taken to accomplish a task. This approach to designing locomotion is more generalizable to a broader range of systems, but often lacks the robustness that the older, more heuristic controllers can provide.

This work investigates these different approaches to implementing the control of dy-

namic locomotion in legged robots of varying design that all incorporate compliance or force controlled actuation, in order to provide insight on their utility in physical hardware implementation. The results of this work can help guide roboticists when deciding on a particular control strategy or choice of actuation when implementing legged locomotion. Providing solutions to this problem of having a machine get around with legs instead of with wheels or thrusters would not only expand the possible applications of modern automation technology, but also suggest potential methods for solving other complex, hybrid dynamical systems. Furthermore, this topic contributes to the subject of force control using various forms of compliant actuation, a concept that has application in a vast range of robotics areas such as legged locomotion, manipulation through contact, and human-robot interaction, to name a few. Finally, the use of optimization for this work can potentially provide some insight into the phenomenon that is legged locomotion.

1.2 Background

1.2.1 A Brief History of Legged Robotics

Arguably, the field of legged robotics has been around for as long as mankind has attempted to create a machine in their own form. The earliest 'walking machines' date back as early as the mid 1800's with Chebyshev's "The Plantigrade Machine" and Rygg's mechanical horse [107], where meticulously designed linkages coordinated limb movements to make the machines walk. However, these machines are more kinetic sculpture than robot, as their legged motion consisted of an open-loop, fixed gait. Eventually, walking machine aficionados

began adding more degrees of freedom (DoF) to the legs of their machines, with 3 DoF being the minimum to position the foot in 3D space and 6 DoF the minimum to also define an orientation. One of the first instances of a legged quadruped design with 3 DoF per leg was the 1968 GE walking truck, a pickup truck with four legs that were manually controlled by an operator in the truck cabin. [77].

At a similar time, Ichiro Kato began his pioneering work on humanoids and bipedal machines, with the artificial lower limb model WL-1 in 1967.This work later led to WAP-1, 2 and 3, a series of bipeds with a gradually increasing number of DoF which were eventually able to walk in 3D. In 1973, he introduced the first full-scale humanoid, WABOT-1 [69], followed by a number of bipedal robots with quasi-static gaits in the 80s. This work pioneered the field of anthropomorphic bipedal robots, leading to what we know of today as the *fully actuated humanoid*; that is, humanoid robots that have a minimum of 6 DoF per limb, in order for each limb to be able to achieve arbitrary configurations in 3D space. One of the most iconic humanoid robots of the early 21st century is Hondas ASIMO (Advanced Step in Innovative MObility) [108], a successor to Hondas P2 system first debuted in 1996.

Before the realization of the fully-actuated humanoid, there was a completely separate phenomenon occurring that was being pioneered by Marc Raibert. When Marc Raibert approach the problem of legged locomotion in 1981, he embraced the idea that a leg should be dynamic and springy, as muscles and tendons on biological organisms are, and began research on a series of groundbreaking computer controlled pogo-sticks [97]. The first so called *Raibert Hopper* was a single pneumatic shank actuator leg attached to a horizontal body that hopped around in a plane by being attached to a boom arm. After Raibert

released the work done on his hoppers, several others began developing similar systems that featured minor changes like electric actuation instead of hydraulic, or metal springs instead of pneumatic [2, 20, 61]. This *Raibert Revolution* changed the way many people thought about legged locomotion, and its effects can still be seen in legged robots to this day that feature springy legs [58, 28, 51]. Raibert went on to establish Boston Dynamics, one of the first legged robotics companies, making iconic hydraulically actuated systems such as Big Dog [96] and ATLAS [34], and later electronically actuated systems like Spot [35].

Another branch of legged locomotion started shortly following Raibert's work. In 1990, Tad McGeer published his work on developing a walking machine that could convincingly walk down a shallow incline without any actuation [81]. He coined the phrase *passive dynamic walking* to describe this form of locomotion, and today the idea of passive dynamics has been extended to not only these unactuated walking machines but also to machines with minimal actuation that incorporate many of the concepts of the original passive dynamic walker. Perhaps the most compelling result of McGeers approach was how natural and humanlike the walkers moved, especially when compared to the locomotion approach of the fully-actuated humanoids mentioned previously. Today, the most prominent result of research on passive dynamic walking with minimal actuation is the Cornell Ranger [6], a robot that is capable of walking over 40 miles on a single battery charge using optimal control techniques to minimize power draw. Importantly, the Ranger shows how incorporating the passive dynamics of a system can greatly improve its efficiency.

Until recently, these different approaches to legged locomotion were independent schools of thought, with the results of one not really being that impactful on the development of

another. One of the reasons for this schism is the fact that every approach was tightly coupled to the design of the physical robot it was applied to. Traditional humanoids were typically designed like industrial manipulator arms, Raibert-style robots were designed with springy legs that are amenable to the Raibert Controller, and passive dynamic walkers were carefully tuned kinematic machines. Today, researchers are finally beginning to bridge these gaps in legged robotics, with robots like Jonathan Hurst's ATRIAS [58] and Cassie [103] making use of the passive dynamics of a compliant leg to implement Raibert-style running.

Another thrust that is helping close the gap between these approaches is the development of the high torque transparency, low gearing *proprioceptive* actuators, a modern twist on the classic, position-controlled servo. Today, the legged robots that most successfully utilize this sort of actuator technology are quadrupeds. The MIT Cheetah line of robots [113, 114, 9] all utilize this approach to actuation in their custom-designed actuators and are able to perform some amazing acrobatics and locomotion that are comparable to what can be achieved using hydraulics, but much more efficient [86, 31]. These platforms can all run dynamically with aerial phases, which points to the main benefit of this form of actuation: a controllable compliance. The low gear ratio enables the rapid motions associated with running while also being able to detect and mitigate ground impacts following an aerial phase without the use of external sensors.

The varying approaches to implementing legged systems also makes it difficult to define a current state-of-the-art in legged robotics. While it may be difficult to argue against Boston Dynamics ATLAS humanoid [1] being the best approach to legged robots available, there are cases to be made for MITs highly efficient and electrically actuated Cheetah robot, ANY-

botics' versatile series-elastic ANYmal [59], or even Agility Robotics' digitigrade bipedal robot, Digit [104]. Indeed, it is the co-optimization of a robot's design, along with the integration of different sensors, various control algorithms, and more that produces an effective end result. Perhaps the state-of-the-art just depends on the question being addressed; ATLAS comes the closest to being a human analogue, MIT Cheetah is the fastest and most dynamic quadruped around, ANYmal is one of the best robotic systems in terms of overall system integration, and Digit is a demonstration of how passive dynamics can be utilized as a foundation for the design of legged robot control.

One final phenomenon that has just begun in the past couple of years is the commercial legged robotics industry. With the growing ubiquity of cheap, high-performance electronics and manufacturing, dynamic quadrupeds have begun to join the the ranks of mobile service robots and multirotor aircraft (drones) that can be used to perform autonomous inspection and patrol. Interestingly, it is almost exclusively quadrupeds that are being considered for this task, as they are the most robust for the cost. Another common problem the commercial legged robotics industry hopes to solve is the *last mile* problem of package delivery, where the final few steps from a delivery vehicle to the front door of a house transition from the nicely paved road systems of modern civilization to soft grassy lawns, uneven brick paths, and stairs which are not easily traversed by wheeled robots. Another intriguing proposed use of legged robots is for entertainment. These machines already garner a large online following and commonly fill auditoriums at tech shows and expositions. Perhaps humanity, being biological organisms at our core, have a unique and specific affinity for machines designed in the form of us, or our legged, biological companions. It will be interesting to see the impact

of legged robots in modern society in the coming years, and hopefully this will inspire further development in the field of legged robots.

1.2.2 Legged Robot Design and Actuation

Because the implementation of legged locomotion is so strongly coupled to the physical realization of the robot system, it is informative to discuss the impact that design and choice of actuation has had on legged robots. The earliest legged robots were essentially manipulator arms bolted together at their base with a large plate mounted at the end of the wrist as a foot analog. Compared to the past, design of a legged robot's most defining feature, its legs, has received much more research attention because there is now a better understanding of the essence that allow legged robots to robustly walk, run, and perform other dynamic movements.

1.2.2.1 Leg Design

A legged robot's leg needs sufficient strength to support a significant fraction of the robot weight (if not the entire robot weight), while still being robust to fatigue from constant, repetitive impacts. This is because unlike other continuous-motion transmission interfaces such as rotating wheels or propellers, legs make repeated, discrete contacts with their surroundings which expose them to ground reaction forces (GRF) that can be multiple times the robot's bodyweight (e.g. three for humans [17]). Early on, legged robotics researchers would (somewhat naively) design larger, stronger legs to ensure their robustness, and then utilize large surface area feet to ensure an adequate convex hull of support, or *support polygon*, is

created by the feet during stance. However, the increased inertia of the legs had a profound effect on the dynamics of the system, which at the time was not necessarily appreciated because the actuators moved so slowly that the leg dynamics could be considered negligible.

As actuator technology matured, legged robots became capable of more dynamic motions, which meant the dynamics of heavy legs could no longer be ignored. With the constant asymmetric motions of a leg moving in the swing phase of a gait, it became possible to excite resonances and potentially even destabilize a robot gait that would have otherwise been stable with lighter legs. Even slower, less dynamic robots still need to be able to move their legs quickly to reposition their feet and maintain balance. The humanoid community approached the dynamics problem with control solutions, but other groups took the design approach. Therefore, the current trend on platforms that demonstrate robust locomotion is lightweight legs with small feet [72]. This approach to leg design reduces the leg's overall inertia, which provides two main advantages: 1) It allows the legs to accelerate at high rates while minimally influencing the dynamics of the overall robot, and 2) Many template models assume massless legs, so lighter legs bring the actual design of the robot closer to the template.

To achieve this lightweight-leg paradigm in practice, the most prevalent option has been to keep the leg actuators close to the robot's body, as opposed to collocating them at the joints. Removing the relatively heavy steel, copper, and rare-earth metal components of an electric actuator from the leg allows the leg to mainly comprise of light and strong materials such as carbon fiber, aluminum, and titanium. However, moving the actuators require the introduction of a power transmission mechanism, such as a belt/chain and pulley, or

a linkage mechanism. These two transmissions trade off range of motion for rigidity and reduced backlash. Another approach to leg weight reduction is to design underactuated legs with passive joints to reduce the number of actuators required to begin with. Additionally, modern additive manufacturing techniques are actively used with modern finite element optimization and analysis tools to create robust, low inertia leg link structures.

1.2.2.2 Actuation

The actuators on virtually any robot are one of the most crucial aspects of the system, and a machine without them could hardly be considered to be a robot. Legged robots are no exception, and recent research in legged robot actuation often focuses on new types of soft actuators that mimic biological muscles [70, 3]. While these actuators show great potential, the robustness, maturity and ubiquity of modern electric and hydraulic actuators with a variety of transmissions make them the actuator of choice for most legged systems.

Electric motors that transform electrical energy to mechanical energy via magnetic fields are used extensively in robotics applications, including legged robots. However, traditional legged robots used brushless direct-current (BLDC) motors with a large gear reduction to supply the high torques at low speeds (the classic *servo*) necessary to actuate a leg. This configuration is a relic of classical manipulator arms, and are not amenable to the tasks required of modern legged robots: mainly, force control and impact resistance. One option to achieve force control with a classic servo is to add an external force torque (F/T) sensor, but these fragile sensors often break when exposed to the repetitive strain induced by legged locomotion.

Currently, the main actuation approaches to achieve the force control and impact resistance necessary for legged locomotion are series elastic actuators (SEA) and proprioceptive actuators. SEA typically comprise of a classic servo in series with a spring element whose displacement can be measured to calculate the output force [94]. SEA retain the high torque output and efficiency of the classic servo while gaining impact resistance and force control capabilities from the elastic element. However, they suffer from potentially limited actuation speed and operating bandwidth as determined by the stiffness of the elastic element. Proprioceptive Actuators [115] and Direct Drive actuation [71] have been growing in usage in the legged robotics community over the last decade, in part due to the increasing ubiquity of high-performance BLDC motors created for small unmanned multirotor aircraft. These actuators comprise of a high torque BLDC motors with little to no gear reduction, where the large motor radius compensates for the reduced torque amplification from the gearbox.

Topologically, a proprioceptive actuator is the same as a classic servo; it just involves the clever selection of motor, gear ratio, and low level controller that provide its main benefits. The main advantage of these actuators is that they have highly *transparent* transmission systems which allows for reasonably accurate force control by feeding back the current in the motor [112, 71]. Though this method is not as accurate as using an F/T sensor, it has been shown to be sufficiently accurate for several legged robots. Furthermore, the actuator's stiffness can be changed online by the motor controller without loss of bandwidth. The main drawback of proprioceptive actuators is their high heat production caused by running the large BLDC motors at low speeds, meaning additional design considerations must be made for systems requiring high continuous torque output.

Proprioceptive actuators typically use a control architecture for high-speed locomotion on the legged robots that use them, and are becoming pervasive on any modern robot that requires a variable compliance. Because they are designed for high speeds, they are once again placed closer to the body to reduce the leg inertia, and the feet at the end of the legs they drive are often left as a point contact. The proprioceptive actuator is almost universal in modern dynamic quadruped robots, and its potential uses has not yet been fully explored.

As a final note, a discussion on legged robot actuation would be incomplete without considering the hydraulic actuators popularized by Boston Dynamics. These types of actuators have a much larger torque density than electric actuators, but require a pump to pressurize hydraulic fluid for actuation. Traditionally, the only suitable hydraulic actuators for robots came from the aerospace industry, making them over-designed and prohibitively expensive for legged robotics. In recent decades, Boston Dynamics and a few other groups around the world have made great strides to make them better suited for legged robots [111]. Still, in spite of these successes, only a few groups other than Boston Dynamics have been successful in implementing hydraulics on legged robots due to their complexity and sensitivity to being physically damaged. As hydraulic actuation for the purpose of legged robots matures, it will be interesting to see how the actuation landscape for legged robots changes.

1.2.3 Modeling, Estimation, and Control of Dynamic Legged Locomotion

The earliest attempts at legged locomotion on a robot were very simple and relied heavily on heuristics and intuition, as a legged robot was often no more than a pair of manipulator arms that were bolted together at the base. As automation technology developed, legged

robotics researchers were able to leverage the advanced approaches for controlling robot arms for controlling their legged systems. Methods that involved a certain level of nonlinear dynamics cancellation (inverse dynamics [25]/computed torque [88]/feedback linearization [118]) became popular in robotic manipulators as they enabled the use of the wealth of results from linear control theory, but the biggest problem in controlling walking was the issue of underactuation [122]. Simply put, the inability to produce arbitrary acceleration in the system due to lack of control authority or simply from having more degrees of freedom than degrees of actuation.

In the legged community today, there are now many different approaches taken to control a legged robot, and the choice of control is often linked to the design of the system. For many humanoids, for example, the dynamics cancellation approach is still a popular choice, as their legs are often constructed like manipulators. For many robots with series-elastic actuation, there is also a push towards an emphasis on utilizing the passive dynamics of the legged robot, rather than simply attempting to force the system to behave like a linear system.

1.2.3.1 Pendula, ZMP, and the Limits of Linear Control Theory

Modern linear control theory has produced some incredible results and accomplished some amazing feats in past decades, but it has certain deficiencies that become apparent when working with legged robots. To start, legged robots are typically conceptualized and modeled as a set of rigid bodies that are linked together at actuated, revolute joints. This configuration often leads to a sort of *rigid body tree* structure, in which the base or trunk of the tree is the

central body link, and the various limbs of the robot represent the branches. For locomotion, one must additionally consider the branches of these rigid body trees constantly impacting the ground, instantaneously reducing the foot velocity to zero and adding a new, unactuated joint between the foot and ground. These features of legged locomotion mean that models of it are usually 1) nonlinear, as the second order dynamics always contain velocity products and trigonometric functions, 2) nonholonomic, so the models cannot be solved at arbitrary points in time, meaning trajectories need to be numerically integrated, 3) hybrid-dynamical, ie. the dynamics of the robot in one phase of a locomotion gait (stance) are different than the next phase (flight), and 4) underactuated, which most commonly occurs at the interface between the robot foot and the ground.

These analytical complexities make analytically-derived control approaches to legged locomotion difficult, but it is possible with some simplifying assumptions. One canonical nonlinear system that is an inherent part of nearly all legged locomotion is the inverted pendulum [66]. A legged robot supporting itself on one leg without a rigid connection between the foot and the ground (ie. when taking a step) resembles an inverted pendulum. It was this concept that drove one of bipedal locomotions first big steps forward: the Linear Inverted Pendulum Model (LIPM) used with the Zero Moment Point (ZMP) [128]. The LIPM is simply the inverted pendulum model linearized about its upright position, and the ZMP (equivalent to the Center of Pressure, CoP) is the point on the ground at which the sum of all moments on the robot equals zero. If this point is within the support polygon of the robot, then no rotational motion will be induced in the robot, and the robot will not tip over. In the classic approach [65], a CoM trajectory is generated with the LIPM and

the ZMP is used to produce a set of foot placements that would allow the robot to walk. This pendulum trajectory is then executed on the physical hardware through the dynamics cancellation approach mentioned before (with the foot as the base of the pendulum and the hip or center of mass (CoM) being the lump mass at the end of the pendulum), or simply through position control via inverse kinematics (if the leg motion is conservative enough). This approach was quite pivotal in humanoid robotics, and is often still the first thing to try on a humanoid robot today.

It is interesting to look at how this classical control approach dealt with the constraints imposed by legged locomotion. It removed the nonlinear and nonholonomic problems through using a much simplified model, dealt with the hybrid modes by avoiding impacts altogether [1], and handled underactuation by simply creating a scenario that ensured the system would never be underactuated. That is to say, as long as the the ZMP of the robot is within the support polygon, then the leg that is supporting the robot is effectively a fixed based manipulator that is amenable to all of the classical control approaches for manipulator arms. This method was later extended to utilize the concept of Centroidal Momentum [85], which helps encompass the motion of the entire robot to allow for more precise tracking of these CoM trajectories. While the ZMP preview control method is effective, it produces somewhat unnatural walking, in which the knees of the robot are always bent to keep the hip at a constant height. It has also been shown that this approach is rather inefficient when compared to a more natural human gait [102]. One of the reasons why this approach still

[1]The linear inverted pendulum keeps a constant height, so with enough control authority one can assume that the foot of the leg in the air will touchdown with zero net velocity, and thus no energy lost during impact.

remains relevant is because of the design of humanoid robots. That is, classic humanoids adhered to the previously mentioned paradigm of several manipulator arms bolted together at a torso, meaning there is little compliance in the system, and the gaits relied heavily on accurately timed position-based trajectories.

1.2.3.2 The Raibert Renaissance and Passive Dynamics

An alternative approach to legged locomotion was achieved in the 1980's by Marc Raibert, the founder of Boston Dynamics. His approach to legged locomotion was a drastic departure from the other approaches being taken at the time, and his seminal work in developing the "Raibert Hoppers" and "Raibert Controller" based on intuitive models and heuristics are commonly cited in legged robotics literature to this day. As an aside, one may find it interesting that Raibert's most impactful contributions come in the form of a couple of actuated pogo-sticks and a control algorithm based on heuristics that can be written in 3 (short) lines of code. However, Raibert's machines spawned a plethora of Raibert-style hoppers (not to mention Boston Dynamics), and the simple fact was that these machines worked *really* well. The author posits that Raibert's impact on the field shows that, at least in legged robotics, there is still room for novel contributions to be made simply from an intuitive understanding of locomotion. Raibert's approach is still being used to create dynamic running and jumping locomotion on a variety of legged systems with springy legs, and can often be fairly intuitive in their construction [51, 58]. However, they are typically very empirical in nature, making their development more of an art than a science.

More generally, Raibert's hoppers showed the important of two somewhat revolutionary

concepts: the importance of compliance in legged locomotion, and the utility of passive dynamics. Raibert's original hoppers utilized a pneumatic cylinder with controllable pressure as a leg, effectively creating a spring which could be injected with energy. Later iterations of Raibert style hoppers began implementing the spring in other ways, from using a coil spring to spring plates to bows [20]. The Raibert style hopper is still experimented with today, but this concept of having a spring in addition to the typical actuation started a new trend which is also seen today: the series elastic actuator (SEA) [94]. The original SEA concept was developed by Jerry Pratt, and in contrast to the way Raibert used springs as a component of a leg, the SEA used springs directly at the output of the actuator. With this configuration, a robot utilizing only SEA has compliance at each joint. Importantly, this allows torque control to be used on the robot without a separate F/T sensor, as the torque output by the SEA can be determined by measuring the deflection of a spring. However, adding this level of compliance at each joint simply adds additional components to be modeled for control, as now it is necessary to consider the dynamics of the motor, spring, and rigid body links. Controlling SEA, in contrast to controlling classic Raibert hoppers, is often an exercise in nonlinear dynamics.

The other impact of Raibert's hoppers was that they began breaking the traditional control paradigm set by roboticists at the time, which was to design a simple, rigid-linked system, and then cancel the nonlinear dynamics to achieve a nice, linear system amenable to linear control theory. The key insight introduced by Raibert and expanded on by others later was that locomotion is inherently a compliant behavior, and that incorporating both compliance and the passive dynamics of the system are key in developing effective locomo-

tion. In 1990, Tad McGeer developed a machine that could walk down a shallow incline on gravity alone thanks to its passive dynamics, and made a point to show how 'human-like' these gaits were. Later, passive dynamic walkers that utilized actuation and could walk across flat ground were developed. The passive dynamic walking phenomena has become somewhat less prevalent in recent years, but importantly it showed that a legged system, if modeled to walk well, can achieve walking given the physical machine closely matches the model. Furthermore, perhaps more importantly, the walking model does not even need to be incredibly complex, and it is often simpler for control design to just use a simple model that encompasses the main dynamical modes of the physical robot.

1.2.3.3 Simple Template Models for Control

Modeling the world around us is a difficult task, especially when it comes to modeling biological creatures. The symphony of carefully timed chemical reactions and electrical impulses that make even the simplest of animals function is an incredible feat of nature. The act of just attempting to model such a system would be difficult, let alone attempting to use that model for control. Fortunately, Bob Full and Dan Koditschek showed in [38] that perhaps this complex modeling is not necessary, and that biological systems can often be modeled very simply using so called *templates and anchors*. Simply put, a template is a simple model that can be used to represent a much more complex system, the anchor. By developing control around these simple models, more complex robots which encompassed the key dynamics of the simple models could be made and controlled as desired.

To illustrate this point, Raiberts classic hoppers have now been associated with the

Spring Loaded Inverted Pendulum (SLIP) or spring-mass model. This model is comprised of a point mass that interacts with the ground through a single massless spring leg that can be positioned arbitrarily during flight. This simple and elegant design template facilitated the use of the classical Raibert Controller, an empirical controller that composites three controllers that each control one aspect of the hoppers motion: the vertical jumping height, the horizontal velocity of the locomotion using foot placement, and the pitch of the body. This sort of controller has been formalized as the composition of several templates that each control a particular degree of freedom. These templates can then be anchored in a physical system and composed together such that it creates a stable running/hopping locomotion [97, 28].

To extend the idea of templates even further, the LIPM mentioned previously can also be considered as a simple template model, as it is a drastically reduced order model which is used as a basis for controlling the much more complex humanoid robot. Granted, the LIPM lacks the biological motivation that the SLIP model does, but is effective nonetheless. The model used by many passive dynamic walking research was the Compass Gait Walker [45], which resembles a double pendulum that has been folded over such that is constantly falling onto its 'front' leg. Another simple model that is utilized today is the *hovercraft* model, in which the robot is modeled as a single rigid body floating in 3D space, and its legs are thrusters that intermittently provide forces and moments on the rigid body [86, 122]. This simple model makes some assumptions, such as the leg dynamics not affecting those of the main rigid body (massless legs), but often the controllers made around this model can be robust enough to handle the disturbances introduced by non-massless legs. The impact of

simple models for legged locomotion should not be understated, as it has produced some amazing results. However, it is important to mention that making use of simple models does assume that the physical hardware somewhat fits the model. If the hardware diverges too much from the simple model, even the most robust of controllers will fail. In this case, it is often necessary to look towards other approaches of control.

1.2.3.4 Underactuated Control

As mentioned previously, underactuation occurs in a system when it is unable to produce arbitrary accelerations [122]. In the case of legged robots, underactuation typically manifests in two ways. The first is when the legged robot does not have a flat, actuated foot on the ground, such as on a robot with *point feet*, or on a robot with a flat foot that is now tipping over the edge of the foot. The other case of underactuation is when the robot is unable to provide sufficient torques at the joints to produce a desired motion. In both of these cases, the dynamics cancellation approach to control tends to fail. Additionally, if the robot was not designed in such a way to resemble a simpler dynamic model, then the approach of templates and anchors will also fail. In these instances, it is necessary to look towards underactuated control.

The canonical examples of underactuated systems are the 'Cart-Pole' and the 'Acrobot' [119]. The Cart-Pole is a pendulum pinned to a cart which can move side to side, and the Acrobot is a double pendulum which is actuated only at the joint connecting the two links (the 'elbow' joint). The classic control problem posed for these systems is the *swing-up* problem, in which the pendular components start hanging down in gravity and need to be inverted so

that they are balanced upright[2]. The erratic and non-intuitive motion that Acrobot takes to solve the swing-up problem (which is not unlike the motion a gymnast on a hanging bar takes to solve the same problem) is a clear illustration of the challenges of underactuation. In [118], Spong developed the technique of *partial feedback linearization*, in which systems that could not be fully feedback-linearized (ie. have all their nonlinear dynamics cancelled) could be partially feedback-linearized (ie. certain states are linearized) given certain conditions on a *strongly coupled* system. Another approach to underactuated control is transverse linearization, in which an arbitrary trajectory can be effectively stabilized by making a local linearization of the *transverse dynamics* and stabilizing those dynamics [79, 4]. These methods tend to have limited regions of accuracy, and again are attempting to eliminate nonlinearities, rather than use them.

Another widely accepted approach to control of nonlinear systems is Hybrid Zero Dynamics (HZD), a form of nonlinear control that was utilized in [132] as a means of producing legged locomotion on a bipedal robot. In this scheme, the many degrees of freedom of the robot are virtually constrained to follow a specific set of zero dynamics that result in a stable limit cycle for the system. The hybrid aspect of HZD refers to the extension of the zero dynamic concept to hybrid systems, such as ones involving the rigid body impacts. Designing HZD controllers relies on the careful selection of virtual constraints to produce desirable behavior, involving controlling the pose of the robot based off the support leg angle. A legged robot walking using virtual constraints is effectively walking like a passive dynamic walker, meaning the stability properties would need to be inherent in the design of the zero

[2]Once upright, linear controls can be used to stabilize the system close to the linearization point.

dynamics. This approach has seen success on many different systems such as [22, 120], as well as a fully actuated humanoid-style biped with additional ankle compliance [102].

Two additional methods for underactuated control are motion primitives [93] and central pattern generators [62]. Motion primitives refers to an approach where several discrete motions are combined in a state machine to produce a reasonable transition from one motion to another. This method is attractive because it provides a solution to the difficult problem of handling and adapting to a broad range of environments and possible disturbances which could take place during legged locomotion. Motion primitives are also commonly used in animation, where dynamics are less important than the smooth transitions across a variety of terrain. The actual discrete motions from motion primitives can come from any source; sometimes they are based on the approaches described previously, but for the animation case they are often done with motion capture on an analog subject. Central pattern generators (CPG) have shown to be the foundation of many biological actions, including legged locomotion. CPG have the benefit of being clocked, periodic motions that don't require higher level monitoring or feedback, and have been shown to be really effective for controlling swimming on robotic fish [23]. However, legged locomotion often has discrete jumps in the state due to the foot impacting on the ground, and CPG can have difficulty accounting for these sorts of disturbances.

1.2.3.5 Optimal Control

Optimal control here broadly refers to the class of control approaches that involve the minimization or maximization of a certain *objective* or *cost* function. In the field of legged

robotics, optimal control typically comes in the form of minimizing the energy used by the system, either by looking at the absolute power used by the actuators, or by looking at other metrics such as the cost of transport. In linear controls, a classical example of an optimal controller is the linear quadratic regulator (LQR), which guarantees an optimal control input based on a cost function defined by looking at the weighted square of the inputs and states. Interestingly enough, even in the face of nonlinearities, underactuation, and more, LQR is still relevant in many different legged control frameworks, thanks to techniques like partial feedback linearization and transverse linearization. In many cases, LQR can be used to locally stabilize a trajectory or a linearized Poincare map. Still, the LQR is still just regulating, not necessary discovering new trajectories or gaits. In these instances, a different approach is taken.

To generate these longer, more involved trajectories, a common, modern approach is to use trajectory optimization [5]. Trajectory optimization has its roots in generating optimal trajectories for spacecraft, where fuel economy is critical [52]. In legged robotics, trajectory optimization is used predominantly for generating trajectories (typically periodic) which result in the robot efficiently walking, running, or otherwise locomoting. There are two broad forms of trajectory optimization: shooting and collocation. *Shooting methods* involve optimizing the control and states on a discrete time grid and integrating the dynamics of the system in between time points [75]. *Collocation methods* forego integration between the time grid and simply model the dynamics as polynomials in the states and simply attempt to collocate (ie. match at every point in time) the value of the polynomial and its derivatives with the value of the state and its derivatives [127]. These formulations have advantages and

disadvantages, but both have uses with respect to legged locomotion.

As with many forms of optimization, the art of producing good results lies in the formulation of the problem itself. Often times, simply attempting a free optimization with a minimal number of constraints produces quite sensitive gaits which are practically impossible to stabilize on a physical system, so additional constraints are often applied. Furthermore, these methods can have difficulty handling an unknown sequence of contact conditions, as changing contact conditions typically signal a switch in dynamics. When a contact sequence cannot be predefined, the two common solutions are to approximate the contact as a continuous (but stiff) force model that never needs to switch [125], or to reformulate contact into a linear complimentary constraint that are more suitable for the algorithms to handle [91]. Still, the trajectories found through trajectory optimization are not regulating controllers in the sense that they will stabilize some gait; with the exception of the virtual constraint approach in HZD, the optimal trajectories have no guarantees of stability.

Trajectory optimization offers an enticing solution to the problem of generating locomotion gaits on legged robots, but don't offer much in terms of stability or robustness. In the response to this, a recently popular method for dynamic legged locomotion is the use of finite horizon model predictive control (MPC). This form of control has been around for quite some time and is recognized across several fields. In the context of legged robotics, MPC can refer to either its linear or nonlinear variants, and can be used to both plan a motion and also provide feedback to ensure the motion is followed [32, 57]. Specifically, MPC utilizes a model of the system to be controlled, and at each time step determines over a finite horizon a sequence of control inputs and corresponding state trajectory to achieve some desired high

level command or motion plan. MPC when applied to legged robots also typically requires the use of optimization, be it convex or otherwise, to solve for the control inputs subject to various constraints. This approach has been used quite successfully on quadrupeds such as the MIT Cheetah 3 [31] and ANYmal [133], and has also shown some success on humanoid systems [36]. Optimal control, especially MPC, can offer practical and effective solutions to legged locomotion. However, when it comes to nonlinear optimization, there are always implementation details that must be considered.

1.2.3.6 State Estimation

State estimation has an interesting role in legged robotics. In most mobile robot applications, estimation of the robot state within a global reference frame and then planning a path through the space is the main problem being solved [124]. The assumption in this case is that the robot is able to then execute whatever motion plan it was given to reach its final goal. However, for legged robots, this assumption is not always true, as legged locomotion itself is a challenge that needs to be solved first. As such, estimation can often be somewhat decoupled from the locomotion control problem, which can be seen from the fact that a significant body of work has been done on developing visual-inertial estimation on other mobile platforms like multirotors [30].

State estimation on mobile platforms typically rely on some form of *odometry* that estimates the distance traveled, along with some means of correcting or updating the predicted state taken from the odometry. On wheeled platforms, the revolutions of the wheels can be used to measure the distance travelled, and cameras tracking visual features can be used

to update the prediction. On flying platforms, prediction is often done by integrating an accelerometer and gyroscope, and then updates are performed with vision. Legged robots are typically equipped with joint encoders which can measure the leg configuration, which offers a unique form of odometry that other mobile robot systems might lack.

Legged robots have the advantage of (typically) having at least one connection with the ground during normal locomotion, meaning a local estimate of position with can be tracked as long as the feet never slip, and the kinematic model of the robot is precise. However, sensor fusion with some other form of sensor is more robust in most applications, and is the approach taken on most legged platforms that require a global position estimate [84, 14, 13]. Interestingly, these results also suggest that even in the absence of a global position reference, the body-centric velocity is still observable [14]. This can be useful when implementing a legged robot as a mobile robot, where a higher-level path planner produces velocities that the legged robot tracks over time.

1.2.4 Legged Robots In Real Life

Legged locomotion offers a rich variety of intellectual challenges and motivations, both in a mathematical sense as well as a practical one. In theory, the complex, discontinuous dynamics of legged robots provide a canvas for attempting the latest approaches in nonlinear dynamics. In practice, legged robotic platforms are some of the most complex mechatronic platforms around, utilizing some of the highest numbers of sensors and actuators compared to other robotics sub-fields. When it comes to the physical implementation of control on legged robots, there are an immense number of considerations that must be taken, many

of which have kept them from being practical for commercial applications until recently. Despite all of these hurdles, the author believes that legged robots are here to stay, if not as commercial products, then as academic exercises that can teach and inspire.

1.2.4.1 The Reality Gap

Some of the biggest challenges in developing legged robots (and other types of robots) are the differences between the simulated world used for development and the real world. This disparity between simulation and reality is known as the *reality gap* [63], and it can be found in nearly any application in which a simulation attempts to model the physical world around us. For certain robotic systems such as multirotors, simulations can often capture the majority of the dynamics, thanks to the simple yet effective power transmission systems on modern multirotors, as well as the fact that they are floating in the air and do not have to consider contacts (typically). Legged robots by contrast are highly complex, power-limited systems that are modeled as complex rigid body trees with constantly changing contact conditions. In this case, it is necessary to model things such as friction and impacts that all can be a challenge to model accurately in physics simulators.

When it comes to dynamic behaviors, for a complex legged robot, a typical simulator such as Gazebo [74] or V-REP [105] can provide a proof-of-concept that the motion is possible, but it is typically necessary to perform meticulous testing on the physical robot afterwards in order to correctly tune the system to function as desired. As an aside, the reality gap is also one of the main challenges of having a robot execute a trajectory that has been found through trajectory optimization or machine learning; if you generate your trajectory

in simulation, the motions will likely not be reflected on the robot, but if you learn/optimize on the robot, you will need to spend a lot of time fixing broken hardware due to falls and crashes.

1.2.4.2 Hardware and Software Tractability

Another consideration that must be made when implementing algorithms on real hardware is their ability to be executed quickly and effectively. While certain systems can get away with a single ARM based processor running their estimation and control loop, many legged robots do not have this luxury. Most legged robots will utilize multi-core processors with speeds comparable to those in modern notebook computers, and it is also common to have an ARM based processor for low level control of each limb or joint as well. However, while this distributed processing approach improves the general execution speed of systems, it adds the challenge of communicating between the different processors. Typically, serial communication is used, though Ethercat is also a popular choice [64]. However, one downside of Ethercat is the fact that it requires the use of heavy controller modules and bulky Ethernet cables, which all add unnecessary weight to a legged robot. Assuming that communication is not a problem, the algorithms being run still need to execute fast enough for control. Currently, nonlinear optimization is difficult to do online with a full rigid body model, so it may be more tractable to run optimizations on a simpler (template) model. Perhaps the solutions to many of these problems will scale with Moore's law, but in the present they still pose significant barriers to effective legged robots.

1.2.4.3 A Note on Stability

Currently, there is no standardized definition of stability for legged locomotion. Often times, local stability can be defined for a particular legged system, and that legged system can be proved to be locally stable. However, these definitions often leave out one major form of instability that is somewhat unique to legged robots: falling over. Falling over is relatively intuitive for legged systems such as us humans to understand, but it is difficult to quantify on a legged robot. This can make finding solutions through optimization difficult, as they require specific mathematical definitions of constraints or tasks. Typically, stability of legged locomotion is handled with limit-cycle stability analysis, but often times the regions of attractions are too small to even handle something as minute as the backlash in the gearbox of a joint actuator. In the present, the most robust method for verifying stability on a legged robot is still just deploying an algorithm, and verifying that it performs the expected motions while not falling over.

1.3 Organization and Contributions

Today, legged robots are finally beginning to leave the lab and are becoming pieces of technology that can be utilized for commercial applications. However, there is still a broad range of questions in legged robotics that have yet to be asked, and that are unlikely to be addressed in industry. One such question is that of design morphology and configuration. Today, a large majority of legged robot designs fall broadly into just one of two archetypes: the fully-actuated (bipedal) *humanoid* and the dog-like *dynamic quadruped*. These types

of platforms have had a large amount of development resources invested into them, and therefore do not leave much to the imagination when it comes to their design and implementation. However, one need not look far to see that the space of possible legged robot designs is incredibly vast, and that robots need not take one of the two forms above to be successful. Legged robots inspired by the design of flightless birds, cockroaches, and more have had a profound impact on the field of legged robots. Even though these systems have not yet had the same limelight as humanoids and quadrupeds, they still contribute to the field as niche platforms for specific tasks, or as platforms to explore certain fundamental aspects and models of legged locomotion or control.

The non-anthropomorphic biped, or NABi, that is introduced here is one such exploratory platform, and is the main contribution of this work. The NABi system is neither a fully-actuated humanoid nor a dynamic quadruped, but rather combines elements of both in order to explore dynamic bipedal locomotion on a heavily underactuated bipedal design. Furthermore, NABi explores the use of force control and proprioceptive actuation for bipedal locomotion, something that is traditionally only done on dynamic quadrupeds. Locomotion algorithms are then developed for walking and jumping on NABi. While NABi's relatively specialized design does mean that the particular locomotion algorithms introduced here may not be directly generalizable to systems that do not resemble NABi, the concepts and techniques that are used to implement these algorithms are generalizable to different legged systems. Importantly, NABi has the quality of being a (pair of) physically realized robot(s), meaning the results are not limited to simulation or theory, and that the intuition and techniques developed from this work can be used to guide future roboticists working with

similar hardware.

The hardware of NABi itself also represents an important component to the novelty of this work. NABi utilizes the recently popularized *proprioceptive* actuation through the use of back-drivable actuators with low gear ratios. These actuators are typically incorporated into the legs of dynamic quadruped robots in such a way that the actuators are all placed near the hip to minimize the leg inertia. This differs from the design philosophy of most fully-actuated humanoids, where the actuators are collocated with the corresponding joint. The legs on the proprioceptive version of NABi take the quadrupedal approach of a lightweight 3DoF leg, sacrificing an actuated ankle in favor of a more lightweight, dynamically capable leg. The impact of taking this approach to leg design and actuation is a recurring motif in this work.

More generally, this dissertation contributes to the area of legged robotics and locomotion, with a focus on the use of force control for dynamic walking and jumping locomotion. It also emphasizes many of the more practical technicalities of physical implementation on hardware, and comments on various heuristics and other topics that may otherwise be skipped when considering a pure theoretical or pure simulation based system. As a general note for the reader, due to the variegated nature of the topics being covered in this work, there is not a single section that covers all the theoretical background necessary in its entirety. Instead, background topics are introduced as they are used in the work, allowing for each chapter to be read independently. That being said, the author believes that reading from beginning to end still provides better flow and improved consistency for the reader.

The remainder of this dissertation is organized to reflect the development process of the

NABi family of robots. Chapter 2 introduces the concept of the novel Non-Anthropomorphic Biped, Version 1 (NABi-1), a simple, minimal planar biped that has compliant feet elements and explores how intelligent mechanical design can simplify the process of developing dynamic locomotion. Some simple motion controllers on NABi-1 are developed, and their performance is discussed. Chapter 3 goes into the next iteration of NABi, NABi-2, which features a similar leg configuration as NABi-1 but now incorporates proprioceptive actuators, and a different approach to control. Chapter 4 takes an aside to introduce state estimation for legged robots, and how it was introduced to NABi-2 as a prerequisite to realize directional locomotion. Chapter 5 details how directional walking and jumping locomotion was implemented on NABi-2 once state estimation was available, and the difficulties of developing these algorithms for the unique NABi-2 design. Chapters 6 begins a discussion of the performance of these force control approaches to locomotion by looking at a simple three-link monoped design that is made to hop using a simple Raibert-style compositional controller and a trajectory optimized for energy efficiency. Many different three-link monoped designs are tested using a monte carlo method, and show the energetic cost of having a stable, robust controller. Chapter 7 summarizes and concludes the dissertation, reflecting on lessons learned in the research, as well as offering possible avenues of future research.

CHAPTER 2

NABi-1: The Non-Anthropomorphic Bipedal Robotic System

Abstract

This chapter presents a novel bipedal robot concept and prototype that takes inspiration from humanoids but features fundamental differences that drastically improve its agility and stability while reducing its complexity and cost. This Non-Anthropomorphic Bipedal Robotic System (NABiRoS/NABi-1) modifies the traditional bipedal form by aligning the legs in the sagittal plane and adding a compliance to the feet. The platform is comparable in height to a human, but weighs much less because of its lightweight architecture and novel leg configuration. The inclusion of the compliant element showed immense improvements in the stability and robustness of walking gaits on the prototype, allowing the robot to remain stable during locomotion without any inertial feedback control. NABiRoS was able to achieve walking speeds of up to 0.75km/h (0.21m/s) using a simple ZMP based gait and a positioning accuracy of +/- 0.04m with a hand-tuned quasi-static algorithm.

2.1 Introduction

For almost twenty years, humanoid robots, designed to mimic both the form and function of human beings, have been on the leading edge of robotics research. The end goal of the humanoid robot is to have generalist machine that could be used to perform tasks in humans environments, when the tasks are too dangerous, dull, or dirty for biological workers. Significant work has been done in the field of full sized humanoids using expensive high degree-of-freedom (DoF) platforms with complex control and rigid linkages [55]. These platforms have seen drastic advancements in recent years, but are still years away from being practical due to concerns over complexity, cost, reliability, and safety. In order to have a biped platform that could be applicable today, a shift in the humanoid paradigm is necessary.

Research on full sized humanoid robots tends to focus on improving their ability to work in environments designed for humans. This is often accomplished using rigidly controlled robots with two arms and two legs attached to a torso, and with 6 or more degrees of freedom in each limb [55, 65, 68]. These robots are quite versatile, but are generally too slow, unsafe, and expensive to be used practically. An alternative approach to a generalist robot platform has been to mount an anthropomorphic (humanoid) upper body on a robust mobile base which can have wheels, treads, or more than two legs [73, 82], sacrificing simplicity and agility for stability. Significant advances have also been made in the area of compliant mechanisms, which can be used to improve the performance and safety of humanoids [47, 94]. These endeavors utilize elastic elements in the joint actuators, allowing for improved safety and more fluid motions at the cost of control and mechanism complexity.

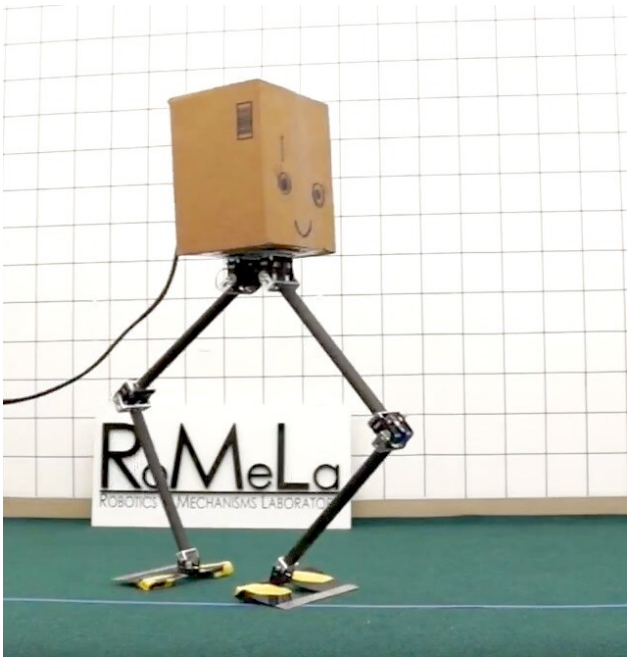


Figure 2.1: The Non-Anthropomorphic Bipedal Robotic System (NABiRoS) prototype, with cardboard box and face dictating the forward-backward direction.

Instead of trying to tackle the deficiencies of bipedal walking with a novel control algorithm on a high DoF humanoid, this work attempts to tackle the issues by rethinking the fundamental design of a bipedal robot and humanoid from the ground up. A biped does not necessarily need to have the exact same morphology or features of a human to perform human-centric tasks. The aim of this work is to develop a bipedal robot with a completely new form factor; a robot whose design can apply the control approaches common amongst bipedal robots today but with significantly enhanced agility and reliability. To this effect, the Non-Anthropomorphic Bipedal Robotic System (NABiRoS) was developed: a robot with a novel lower body configuration that resembles a normal humanoid robot walking sideways (Figure 2.1).

The remainder of the chapter is organized as follows: Section 2.2 introduces the non-

anthropomorphic biped concept and design of the system. Section 2.3 presents the approaches taken to control the system to walk. Section 2.4 discusses the performance of walking on NABiRoS, and how it could be potentially modified to achieve greater flexibility. Finally, 2.5 ends with concluding remarks.

2.2 Concept and Design

Bipedal walking is a complex, three-dimensional control and stability problem that humans often take for granted. One of the reasons why bipedal walking is difficult on classic humanoids is because of the offset in the hip joints in the frontal plane. In typical forward walking, this offset creates undesirable oscillatory moments that force the robot to lean in the direction orthogonal to the direction of motion. A small perturbation in this orthogonal direction can then easily destabilize the robot, and classical humanoid robots would struggle to recover from this sort of disturbance due to the restrictive workspace of the legs and torque-limited actuators. As such, these robots are forced to take small, calculated steps, as well as use a wide array of expensive force/torque and inertial sensors to perform simple walking or balancing tasks. However, these moments dont appear when taking steps side to side (as the hip offset is in line with the direction of motion), and if the main mode of transport is sideways walking, the leg can be simplified significantly. The forward-facing, anthropomorphic knees are no longer necessary, and can be rotated such that the legs are aligned in a plane. By aligning the legs with the sagittal plane, stable forward walking can be achieved using the sideways walking motion (Figure 2.2). This leg realignment also means the number of DoF in a leg can be significantly reduced, and the ankle can be removed and

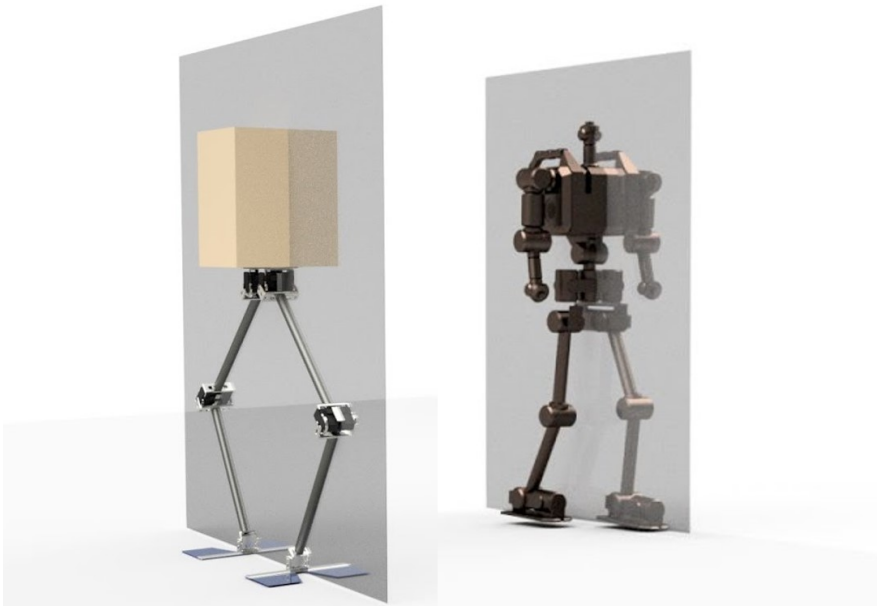


Figure 2.2: Comparison of the original NABiRoS (left) and a more traditional humanoid (right) that shows the sagittal plane of each.

replaced a much simpler foot element. The novel configuration of NABiRoS simplifies the control and stability problem to a two dimensional planar one, and offers a new approach for a simple, lightweight, and practical bipedal robot.

NABiRoS achieves bipedal walking along the front-back axis with only two degrees of freedom on each leg; one at the hip, and one at the knee. There are no degrees of freedom at the ankles, significantly reducing the weight and moment of inertia of each leg. The leg links are comprised of aluminum brackets and carbon fiber tubes. The 'body' of the robot rests above the hip joints and comprises of a box that covers the computing electronics. The materials used allow this robot to be extremely lightweight (3.97 kg, with external power).

Compliant feet are used on NABiRoS to account for small perturbations and uneven terrain, without the need for an ankle joint (Figure 2.3). Legged robots are traditionally

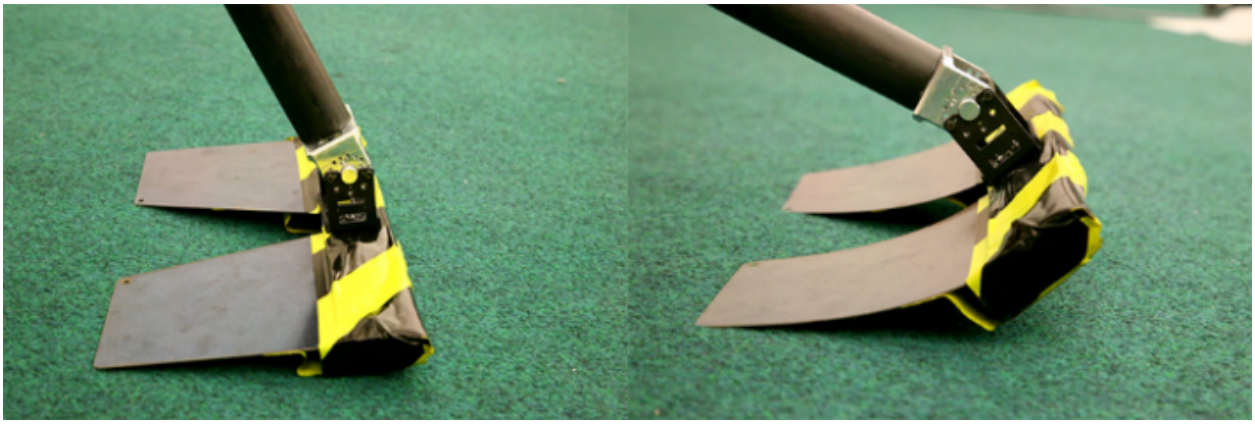


Figure 2.3: Prototype for compliant foot end effector using spring steel toes and foam damping heels.

designed to be mechanically rigid with high-gain, position controlled joints, making it imperative for foot trajectories to match terrain contours perfectly. This type of design works well in a laboratory setting where the environment is known, but often fails when introduced to uncertain real-world conditions [21]. A widely used solution to this problem is the series elastic actuator (SEA), which adds a compliant element in series with a traditional actuator. Adding this elastic element can lead to improved energy efficiency and impact resistance, but often complicates the robot dynamics and adds latency to the control [94]. NABiRoS uses compliant end-effectors rather than compliant actuators at each link, effectively creating a *series elastic leg*. Adding compliance in this way retains SEA benefits, but is simpler to implement mechanically. Moreover, the compliant feet are configured such that they only take action during specific motions. This serves to aid robot dynamics in targeted areas of the walking cycle, while allowing for fast response times when the spring system is not engaged. The deflection angle of the spring is also configurable, analogous to a variable stiffness SEA, enabling multiple types of locomotion.

2.3 Locomotion and Control Design

While the mechanical and system design of NABiRoS is highly simplified compared to that of a traditional humanoid, the addition of the spring steel foot does complicate the control and implementation of locomotion gaits. Traditional *fully actuated* bipeds have enough DoF in each leg to allow the foot to achieve an arbitrary spacial configuration within the leg's workspace. NABiRoS legs are restricted to a plane, meaning a minimum of three actuated DoF is necessary to be fully actuated. The replacement of the ankle joint for the compliant foot makes the system *underactuated*, in the sense that there are fewer actuators than available spacial degrees of freedom.

Underactuation drastically complicates the control of robots in general, making the classical dynamics cancellation approaches used widely in manipulator arms challenging. Furthermore, the NABiRoS prototype is only equipped with position controlled actuators at each degree of freedom, and is not equipped with an inertial measurement unit (IMU) or force torque sensors. In order to command a walking motion with such limited sensory feedback and position controlled actuators, it is necessary to generate joint trajectories offline, and then 'play them back' on the robot. That is, generate a time-parameterized end-effector trajectory in Cartesian space, convert it to joint space using inverse kinematics, then execute it on the robot by looping at a constant rate and interpolating the joint positions based on execution time. This is one of the simplest approaches to control on a legged robot, yet can still be effective if the robot has some sort of inherent stability. Furthermore, while it may be naive to have developed a system with such limited feedback, it presents an interesting a novel challenge to develop some open-loop (though still using joint encoder feedback)

trajectory that incorporates the passive compliance of the foot element.

2.3.1 Quasi-Static Walking

As an initial attempt at locomotion, a simple, quasi-static walking gait is developed. Before the development of algorithms that took advantage of the dynamics of a robot, humanoid robots used statically stable gaits where the center of mass was slowly moved within the robots support polygon [121]. A similar approach is taken initially with NABiRoS to assess its stability and performance. However, the analogy is not perfect, as most static gaits assume a fully actuated leg, and a relatively large surface area foot to have as large a support polygon as possible.

When supported on one foot, the support polygon of NABiRoS can be described as the area covered by the whole foot including the compliant element; however, resting on the compliant foot cannot be described as completely static because the compliant element stores energy when the robot leans onto it. Because of this, instead of slowly transitioning the CoM of the robot from one foot to another, the CoM is rapidly swung over one foot, charging the compliant element. Then, before the compliant foot releases its stored energy, the opposite foot is extended out, taking a step. Finally, to complete the step, the CoM is moved over the foot that was just placed, and the above process is repeated with the other foot (This motion is depicted in Figure 2.4). By utilizing the compliance in the feet during what can be described as a quasi-statically stable gait, the Quasi-Static walking gait is achieved.

On the physical platform, the Quasi-Static gait was implemented by commanding the

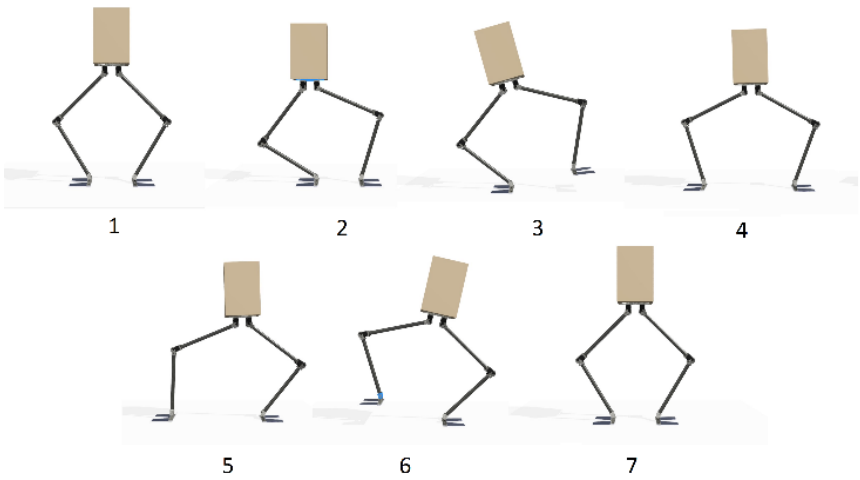


Figure 2.4: Frame depiction of the quasi-static walking cycle. In Steps 1-2, the robot leans over one foot to charge the compliant foot. Steps 3-4 show the robot lifting its leg and taking a step. Steps 5-7 show the robot repeating this process for the opposite leg and returning to the first position.

body to the left and right with respect to the feet, and lifting a foot once the body had shifted over sufficiently. This approach is very heuristic, and was empirically tuned to work, but offered some intuition on how locomotion on this platform could be approached.

2.3.2 ZMP-Based Walking

The quasi-static walking gait utilizes heuristically tuned trajectories that were designed using intuition and experimentation. However, the quasi-static gait does not really utilize any sort of dynamic model and thus produces a very slow, conservative walking motion. In order to have a more dynamic walk, a model-based approach was necessary. In this case, one of the classical ZMP-based methods was used [65]. The key insight of this approach is the use of

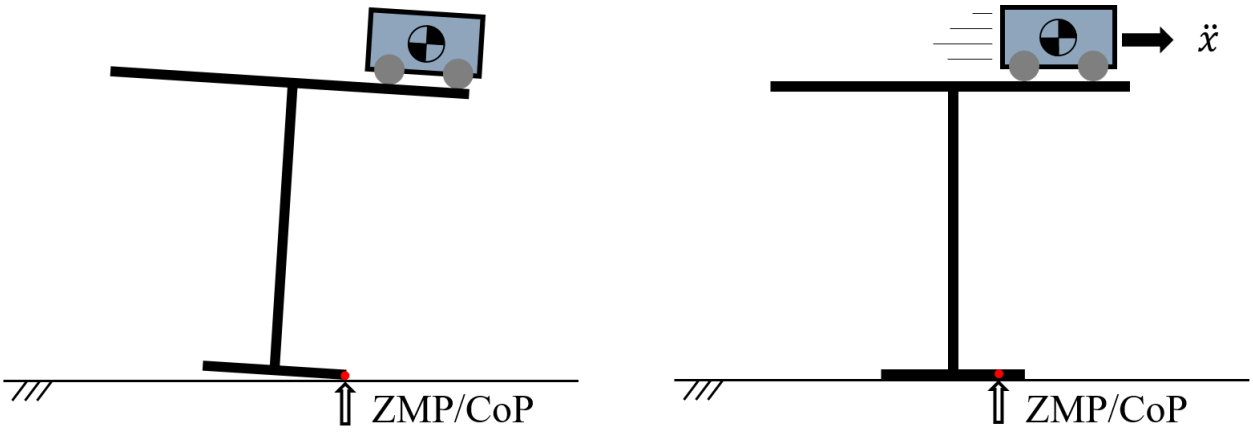


Figure 2.5: Intuition on the cart table model: if the cart is over the edge of the table base and not moving (left), the ZMP is at the edge of the table base and the table will start tipping over. If the cart is accelerating at the edge of the table (right), the reaction force on the table produces a moment that keeps the ZMP within the table base, preventing tipping.

the simplified *cart-table* model, and then applying a *preview controller* to track a prescribed ZMP trajectory.

The cart-table model can be described by a cart that is sliding on the surface of a massless table, with the flat base of the table representing the support polygon generated by the stance foot (feet). The model describes how a cart moving on the surface of the table effects the Zero Moment Point (ZMP) at the table base, as shown in Figure 2.5. This model is illustrated in more detail in Figure 2.6, which also show how the model is embedded into the NABiRoS system. The dynamics of this model can now be represented written out concisely as:

$$p = x - \frac{h}{g}\ddot{x} \tag{2.1}$$

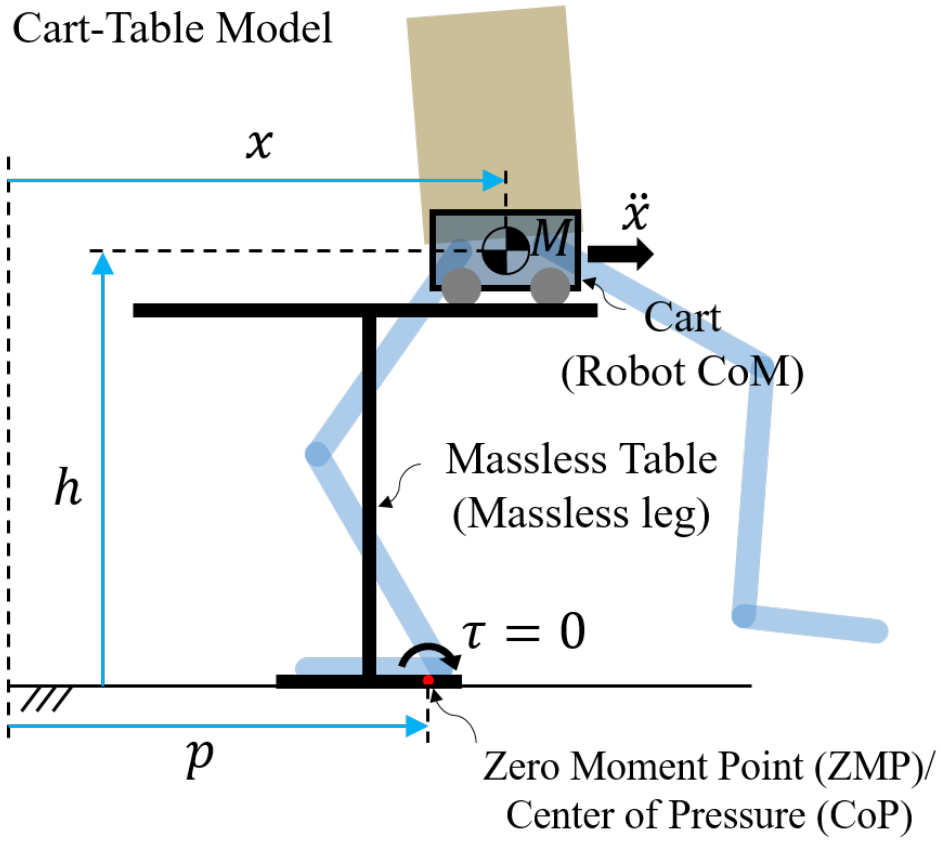


Figure 2.6: Simplified cart-table model used for ZMP based locomotion planning on NABiRoS. The cart represents the center of mass of the robot, and the table base is the stance foot of the robot.

With a corresponding representation in state space:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{2.2}$$

$$p = \begin{bmatrix} 1 & 0 & -\frac{h}{g} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \tag{2.3}$$

Where p is the ZMP or Center of Pressure (CoP), x describes the horizontal component of the Center of Mass (CoM) of the robot (or the cart), h is the height of the CoM which remains constant, and g is the gravitational constant. Note that in the state space representation the system is augmented so that the system input u is the jerk of the CoM.

Now, this model can be used with a preview controller that utilizes future information of the ZMP reference trajectory in order to yield a smooth CoM trajectory. The system in discrete form with sample time Δt can be written as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{b}u_k \tag{2.4}$$

$$p_k = \boldsymbol{c}\boldsymbol{x}_k \tag{2.5}$$

Where

$$\boldsymbol{x}_k = \begin{bmatrix} x(k\Delta t) & \dot{x}(k\Delta t) & \ddot{x}(k\Delta t) \end{bmatrix}^T \tag{2.6}$$

$$u_k = u(k\Delta t) \tag{2.7}$$

$$p_k = p(k\Delta t) \tag{2.8}$$

$$\boldsymbol{A} = \begin{bmatrix} 1 & \Delta t & \Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} \frac{\Delta t^3}{6} \\ \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \tag{2.9}$$

$$\boldsymbol{c} = \begin{bmatrix} 1 & 0 & -\frac{h}{g} \end{bmatrix} \tag{2.10}$$

$$\tag{2.11}$$

Now, to track a reference trajectory, we minimize the following quadratic cost:

$$J = \sum_{j=1}^{\infty} \{Q(p_j^{ref} - p_j)^2 + Ru_j^2\} \tag{2.12}$$

Where Q and R are positive gains on the error in state and input, respectively. For a preview controller, cost J is minimized by the following input:

$$u_k = -\boldsymbol{K}\boldsymbol{x}_k + \begin{bmatrix} f_1 & f_2 & \dots & f_N \end{bmatrix} \begin{bmatrix} p_{k+1}^{ref} \\ p_{k+2}^{ref} \\ \vdots \\ p_{k+N}^{ref} \end{bmatrix} \tag{2.13}$$

Where

$$\boldsymbol{K} = (R + \boldsymbol{b}^T \boldsymbol{P} \boldsymbol{b})^{-1} \boldsymbol{b}^T \boldsymbol{P} \boldsymbol{A} \tag{2.14}$$

$$f_i = (R + \boldsymbol{b}^T \boldsymbol{P} \boldsymbol{b})^{-1} [(\boldsymbol{A} - \boldsymbol{b}\boldsymbol{K})^{(i-1)}]^T \boldsymbol{c}^T Q \tag{2.15}$$

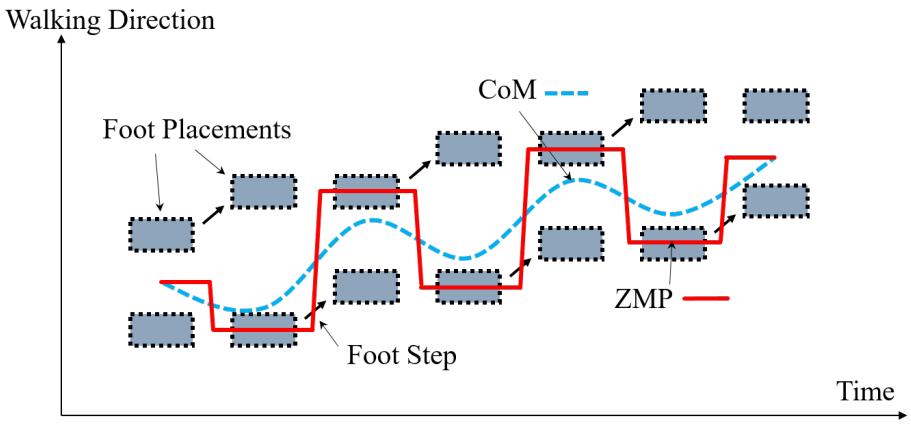


Figure 2.7: Diagram showing foot placements, ZMP trajectory and CoM trajectory over time for NABiRoS walking with the ZMP-based gait.

And $\boldsymbol{P}$ is the solution to the discrete Riccati Equation:

$$\boldsymbol{P} = \boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A} + \boldsymbol{c}^T Q\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{P}\boldsymbol{b}(R + \boldsymbol{b}^T\boldsymbol{P}\boldsymbol{b})^{-1}\boldsymbol{b}^T\boldsymbol{P}\boldsymbol{A} \tag{2.16}$$

The control law described by (2.13) comprises of the normal optimal state feedback law plus an additional feed-forward term which is the inner product of N future reference points gained by weights $[f_1, f_2, ...f_N]$.

To actually implement this result on NABi-1, the following steps are taken:

1. Prescribe foot placements $(P_0, P_1, ..., P_n)$ which the robot will take. Theoretically, the steps can come from a high level planner and can be made as large as the leg kinematics would allow for while maintaining a constant hip height, but for NABiRoS they were manually assigned and made small in order to reduce the effects of the leg dynamics. These are shown with dashed rectangles in Figure 2.7.

2. Define a ZMP trajectory $p(t)$ through the support polygon that is produced by the prescribed foot placements. This ZMP trajectory need not be smooth, but should try

and stay close to the center of the support polygon in order to provide some conservative margins when tracking it. For NABiRoS, the ZMP trajectory was generated by linearly interpolating between the centers of the support polygon at every step. This is shown with a red line in Figure 2.7.

3. Utilize the ZMP preview controller described above to track the desired ZMP trajectory, outputting a CoM trajectory in the process. This is shown with a blue dashed line in Figure 2.7.

4. With the location of the feet and CoM known for a specified number of steps (and time), the robot legs can be commanded to track the trajectory via inverse kinematics (for joint space) or inverse dynamics (in end effector space).

5. An additional motion is necessary to be implemented for the swing leg for taking steps. For NABiRoS, the swing foot is commanded to follow a cycloid function which can help reduce the impact when contacting the ground.

Figure 2.7 provides an example of what the various elements of the gait can look like for a forward walking motion on NABiRoS. In an ideal situation, the trajectory that is executed on the robot should perform the walking motion perfectly, but in reality this is never the case. Typically, there is an additional stabilizing feedback controller that is used to ensure adequate tracking of the foot, ZMP, and CoM. These are enabled by the use of additional sensors such as an IMU that can output center of mass accelerations or F/T sensors at the ankles to estimate the actual robot ZMP. However, NABiRoS does not have any of these forms of feedback, so the trajectories produced from this approach are directly executed,

and the ZMP trajectory, foot placements, and step timing were tuned to produce a motion that was stable over flat (and even slightly uneven) ground. The main tuning parameters for this locomotion method on NABiRoS are 1) the height of the table which determines how much forward-back swing occurs and 2) the duration of the single support and double support phases which are tuned to accommodate the natural oscillations of the spring feet.

2.4 Discussion

2.4.1 Performance of Planar Locomotion

The quasi-static and dynamic walking strategies were implemented on the NABiRoS prototype using a Raspberry Pi 3 single board computer. This was made possible thanks to the reduced number of sensors and also the fact that the actuators were positioned controlled, meaning the high level control loop only needs to run at 200Hz or so. Though these methods are quite lightweight, they worked quite well on the robot, enabling planar locomotion at up to 0.21m/s in the case of the ZMP-based gait. The speed of the robot is affected by many different factors. One of the intuitive reasons is that the robot needs to accelerate both forward and backwards in a step, and to do so more quickly would require modifying both the torque capabilities of the actuators and the stiffness of the foot spring. Another less obvious reason is the relative inertia of the legs to the body, which affects how quickly a step can be taken. Early on, the robot was simply two legs tethered to a power supply, and it was quite difficult to get this system to even take a step due to the fact that the body inertia was so low that attempting to lift a leg would create a moment that tilted the

body down, effectively cancelling out any height generated by lifting the leg. The addition of the cardboard box head improved walking significantly as now larger steps could be taken without worrying about the body dipping down excessively.

The dynamic walking implicitly makes use of the compliant foot elements when the robot is in single support (ie. when the robot is supported on a single leg), even though the compliant element is not explicitly modeled in the dynamics. This is seen by the fact that in preliminary tests, no compliant foot elements were used, and it was much more difficult to create a ZMP trajectory that would lead to successful walking. The foot elements also add significant robustness and stability to the open loop trajectories being played back, which intuitively makes sense due to the fact that they are essentially providing some reactive recovery torque at the ankle based on how far the robot has tilted past a certain angle. The walking is robust to steps up to a few centimeters, and the robot can be slightly perturbed in the direction of locomotion and still recover. However, the design of the system precludes the ability to produce motion out of plane, so perturbations in the direction perpendicular to locomotion cannot be recovered from.

2.4.2 Turning and Out of Plane Motion

In order to be able to change the direction of travel on NABiRoS, an investigation was done on how the system could be redesigned to add minimal complexity but still offer the ability to turn [134]. To this end, several different approaches to turning were tested on the NABiRoS platform. Many of these methods included the addition of a pair of 3 DoF arms that were used to either provide a pivot around which the robot would turn, or use as a

reaction mass to induce a turning moment. These approaches all resulted in very interesting and unique locomotion, and eventually led to the multi-modal locomotion robot ALPHRED [56]. However, these additional limbs made the robot begin to resemble a quadruped, which leaves the scope of the original NABiRoS concept. As such, NABiRoS was made to turn by simply adding an additional actuator at the hip that allows the leg to yaw with respect to the main body. With these hip yaw actuators, the robot can take steps based on the quasi-static locomotion gait, and simply change the direction the leg is facing while it is in the air.

2.5 Conclusions

The NABiRoS concept is a unique one that represents the potential of utilizing unique leg configurations that are not necessarily seen in nature. The original NABiRoS prototype is a simple and inexpensive bipedal robot that is both agile and stable thanks to the use of cleverly designed compliant foot elements, and can walk at speeds up to 0.21m/s using a ZMP based locomotion gait. The author posits that this result shows that the bipedal robots need not be of the humanoid form to be robust or agile, and that the judicious addition of compliant elements can allow for simple control.

At its core, the NABiRoS prototype was a proof of concept to show the capabilities of its novel design. Extensive modeling and analysis of dynamics and stability was not done on this platform due to the fact that the system was simple and intuitive enough to just have heuristic controllers that performed adequately. This is in the same vein as Raibert when he was developing his hoppers originally, and points to the fact that intelligently designed, intuitive

systems do not necessarily need a complex control algorithm to be successful. However, while NABiRoS was able to walk quite well, it was less successful at other tasks such as running or jumping. For these types of motions, it was necessary to modify both the design and the control approach.

CHAPTER 3

NABi-2: Platform for Dynamic Locomotion with Proprioceptive Force Control

Abstract

The performance of traditional humanoid robots is often limited by their design, with high DoF limbs and stiff actuation complicating their dynamics and impeding their ability to operate in unsupervised environments. In response to these deficiencies, this chapter introduces the next iteration in the NABiRoS family of systems: the Non-Anthropomorphic Biped: Version 2 (NABi-2). NABi-2 is a bipedal robot that is a departure from the conventional humanoid paradigm in not only its morphology but also its actuation method. That is, NABi-2 is a platform with a unique leg configuration that is designed around high torque back-drivable electric actuators that provide proprioception and force control capabilities. This paper details the concept and design of this system, and presents a simple yet robust compositional controller for performing in place, two-legged pronking, a form of continuous jumping locomotion that is typically realized with series elastic or hydraulic actuation.

3.1 Introduction

Humanoid robots that emulate the form and function of human beings have been a prevalent area of robotics research for several decades. Traditionally, these systems are realized with stiff, fully-actuated limbs, which are amenable to classical position-based control approaches. These systems see the most success in fairly structured environments, where the robot's movements and interactions are either dynamically conservative or determined in advance. However, the human-fashioned spaces these robots are expected to occupy in the future are often quite unstructured and would necessitate the use of more active behaviors, meaning this type of approach may not be the most effective.

Recently, more capable legged systems have been developed that can operate over unregulated terrain through the use of dynamic running, hopping, trotting, and bounding behaviors enabled by hydraulics [76, 96, 111] or series elastic actuation [98, 59]. These methods of actuation are customarily used for such gaits due to the fact that dynamic locomotion such as running and jumping requires high torques, and electric motor actuators typically achieve this through severe gearing that is easily damaged when confronted by the large impulses experienced during the ground impact following an aerial phase. These robotic systems with such added mechanical compliance usually have complex dynamics, making them difficult to control at best and restricted in their workspace and capabilities at worse. However, recent advances in electric motor technology have shown that electric actuators can be made to provide sufficient power and torque for dynamic locomotion without the need for high gearing. This facilitates mechanical designs with electric motors but no mechanical compliance that can achieve high fidelity force control, proprioceptive sensing, and impact resistance

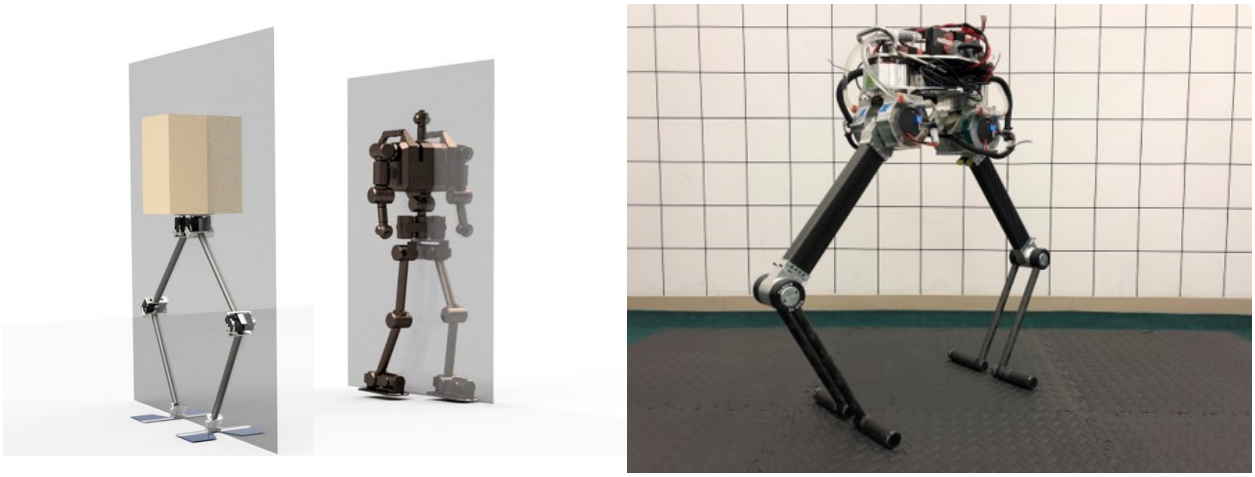


Figure 3.1: Comparison of the original NABi-1 with a more traditional humanoid (left), as well as the Non-Anthropomorphic Biped Version 2 (NABi-2) lower body which utilizes high torque, back-drivable actuators that provide high fidelity force control capabilities (right).

thanks to back-drivability.

This back-drivable, high-torque actuator technology is predominantly applied to legged robotics in the form of dynamic quadrupeds on a large scale [86] and quadrupeds and bipeds on a smaller scale [71, 99], but is not often used on a large scale to power humanoids or bipeds in general. To this end, we aimed to incorporate the potential of this actuation method into the Non-Anthropomorphic Biped (NABi-1), a bipedal robot that attempts to tackle some of the difficulties of biped locomotion and control by rethinking the fundamental design of a bipedal robot. The result of this development is the Non-Anthropomorphic Biped: Version 2 (NABi-2) shown in Figure 3.1, a biped with a unique leg morphology that can perform dynamic behaviors such as two-legged jumping thanks to the use of proprioception and force control. The main contributions of this work are to present the NABi-2 platform and examine how dynamic, event-based controllers can be implemented on a biped with

back-drivable electric actuation.

The remainder of the chapter is organized as follows: Section 3.2 explains the non-anthropomorphic biped concept and discusses the adaptations made from the original NABi-1. Section 3.3 presents an overview of the NABi-2 system. Section 3.4 details how an in-place jumping locomotion can be achieved using a force-control based approach. Section 3.5 discusses some interesting observations regarding the system as a whole, and 3.6 ends with concluding remarks.

3.2 Non-Anthropomorphic Biped Concept

Conventional humanoid systems are designed to be highly versatile in function, but in implementation are often prohibitively slow, unsafe, or expensive due to the approach taken to perform bipedal locomotion. Traditional forward-walking bipedal locomotion on a typical humanoid with 6DoF legs is a heavily underactuated problem that typically leverages some form of dynamics compensation algorithm with inertial and force feedback (to enforce a particular CoM/ZMP trajectory, for example) [65]. This form of walking is inherently difficult due to the fact that there is an offset in the hip joints that is perpendicular to the walking direction, creating undesirable oscillatory moments that can only be accounted for through accurate system modeling combined with sophisticated closed-loop feedback algorithms and high fidelity sensors, or by taking smaller steps. However, these moments do not appear when taking steps side to side, because the hip offset is in the same plane as the direction of travel. If the main mode of locomotion is sideways walking, the forward-facing knees are not being used, and can be rotated by ninety degrees so that the legs are aligned in a plane. By

aligning the legs in the sagittal plane, forward walking can be achieved using the sideways walking motion. With the legs aligned in a plane, the ankle can be removed and replaced with a much simpler foot element.

This simple, 2DoF leg configuration was first featured on NABi-1, a prototype biped with traditional, high-gear ratio servo actuators at the joints and a mechanical spring foot element that can walk using a classical cart-table ZMP approach, and perform two-legged jumping thanks to its series-compliant leg and foot [135]. NABi-1, illustrated in Figure 3.1, demonstrated that bipedal robots do not need to share the morphology of a humanoid to be able perform simple locomotion, but the original platform was limited to locomotion in a plane. Further investigation into how the NABi-1 platform could be adapted to achieve turning was done in [134], with the results showing that adding a third, yaw DoF at the hip allow for the simplest and most effective turning strategy.

While NABi-2 shares the morphology of the original NABi-1 with the additional hip DoF, it differs in how it is actuated. NABi-2 uses Back-drivable Electromagnetic Actuator for Robotics (BEAR) modules at each of its leg joints, while NABi-1 uses position controlled servos. Furthermore, NABi-2 no longer needs a compliant foot element in series with the rest of the leg to perform more dynamic motions because compliance can be achieved through software in the BEAR modules.

3.3 System Overview

3.3.1 Design

NABi-2 shares the same morphological characteristics as the original NABi-1 to continue to take advantage of the benefits that are inherent in the non-anthropomorphic design. However, it has a third yaw DoF at the hip, and its leg joints are all driven by back-drivable actuation modules that can provide significantly improved dynamic performance over most traditional position controlled servos. Additionally, NABi-2 is planned to have a pair of 3-DoF arms with modular end-effectors that can mount assorted tools to enable NABi-2 to perform various inspection and manipulation tasks. The arms can also be potentially used for locomotion and fall recovery, as was explored in [134]. The design and structure of NABi-2 is shown in Figure 3.2.

To maximize dynamic performance of NABi-2, its legs are designed with minimal mass and inertia. A low leg mass and inertia are useful for dynamic locomotion as they can be accelerated more quickly than a heavier leg with the same actuation force. Furthermore, designing the system with lightweight legs allow for some simplifying assumptions in modeling, such as the massless leg assumption. As such, the femur and tibia links are comprised of lightweight carbon fiber tubes epoxied to aluminum joints and comprise around 20% of the total robot mass. All leg actuators are located at the hip to minimize the inertia of the leg, so pair of 1:1 ratio pulleys with a timing belt are employed, as show in Figure 3.3.

The carbon fiber tube on the femur links has a rectangular profile that covers the timing belt and shields the transmission from external contaminants and other external forces. A

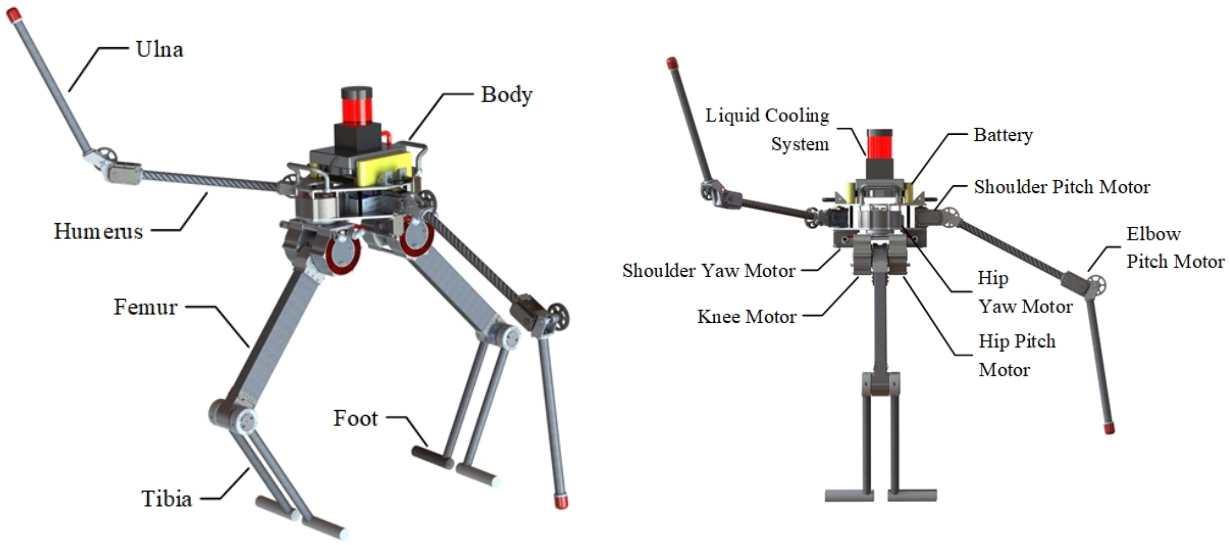


Figure 3.2: NABi-2 isometric view (left) showing key aspects of the non-anthropomorphic design, and a front view (right) detailing the layout of the actuation modules and associated subsystems.

Figure 3.3: Section view of the femur link with the pulley transmission mechanism inside enabling continuous knee rotation.

pair of tensioners located at each end of the femur link keep the belt under tension and prevent backlash that is typical in conventional gearboxes. NABi-2 also takes advantage of the belt-pulley transmission by adopting a double-shin design that allows the knee joint to rotate continuously, enabling some creative methods for locomotion across certain obstacles as investigated in [42]. The lightweight carbon fiber and aluminum structure of the legs are also applied to the arms, but the arm actuators are standard geared servos located directly at the joints for simplicity. Typically, the arms are not attached to the NABi-2 to simplify and expedite the development of locomotion control.

NABi-2 carries two 3250mah 4S LiPo batteries that power all of its subsystems: an Intel NUC computer, a liquid cooling system comprised of a reservoir-pump assembly and radiator-fan assembly (typically used in PC liquid cooling), a LORD MicroStrain 3DM-GX4-25 IMU, and its actuators. Nominal specifications of NABi-2 are listed in table 3.1.

Weight (kg)	11
Height (m)	0.85
Max. Payload (kg)	1.0
Max. Runtime (mins)	30

Table 3.1: Specifications of NABi-2 platform.

3.3.2 Proprioceptive Actuation

To original NABi-1 was designed with high gear ratio position controlled servos, but was able to perform an energy efficient walking gait as well as jumping thanks to the addition

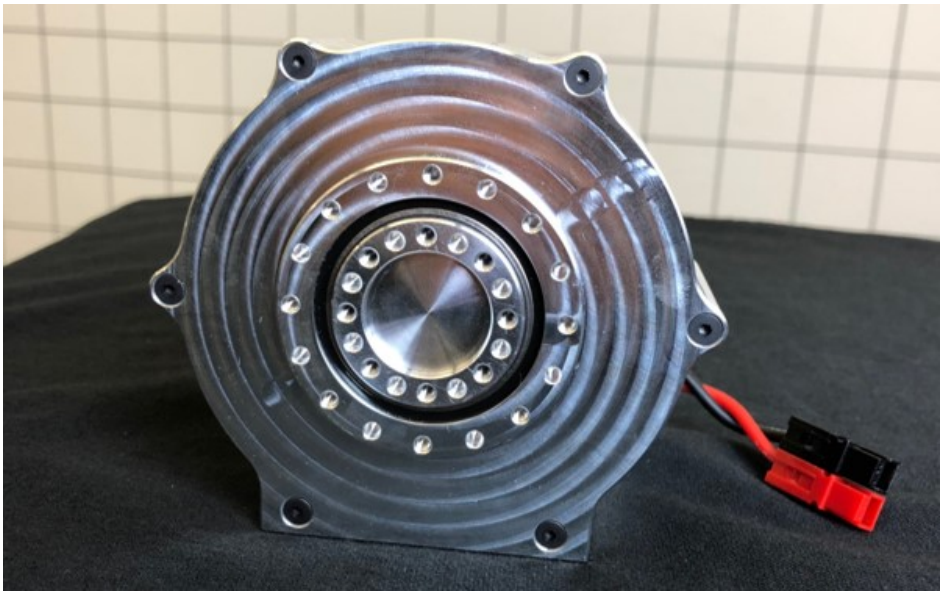


Figure 3.4: One of the Back-drivable Electromagnetic Actuator for Robotics (BEAR) modules that power NABi-2.

of compliant feet that could store energy. However, these compliant feet were not equipped with any form of sensory feedback, so the system was unable to actively control the compliant behaviors of the system or detect certain discrete events like foot touchdown, meaning this series elasticity was forced to be used in a more passive capacity. Furthermore, the bandwidth of the servos was too poor to take full advantage of the elastic element, meaning the elasticity could only be utilized in a very specific set of motions. These are issues that NABi-2 is able to avoid thanks to the use of back-drivable/proprioceptive actuation.

NABi-2 utilizes six Back-drivable Electromagnetic Actuator for Robotics (BEAR) modules to power its leg joints. The BEAR module, pictured in Figure 3.4, was designed with legged robotics in mind, providing the speed, torque, and transmission transparency necessary for proprioceptive force control and impact mitigation [137]. To achieve this, the BEAR

module is built with a low gear ratio single phase planetary gearbox and a large motor with superior torque density characteristics. The power and control electronics are also custom made and packaged within the actuation module itself, improving modularity. Additionally, the BEAR incorporates field-oriented control (vector control) to minimize the current that does not provide torque in the motor, meaning the torque can be controlled relatively accurately using just the current to the motor and no external sensors. Additionally, the actuation modules have the capability to be liquid-cooled for improved actuator heat dissipation to maintain a high torque output. The specifications of the BEAR module are shown in Table 3.2

Weight (g)	670
Gear Reduction	10:1
Voltage (V)	30
Max Current (A)	60
Peak Torque (Nm)	32
Cont. Torque	10
Max Velocity (rpm)	300

Table 3.2: BEAR Specifications

3.3.3 Software Architecture

The original NABi-1 software architecture is structured with simplicity and modularity as its focus to promote rapid development. However, to achieve stable proprioceptive force

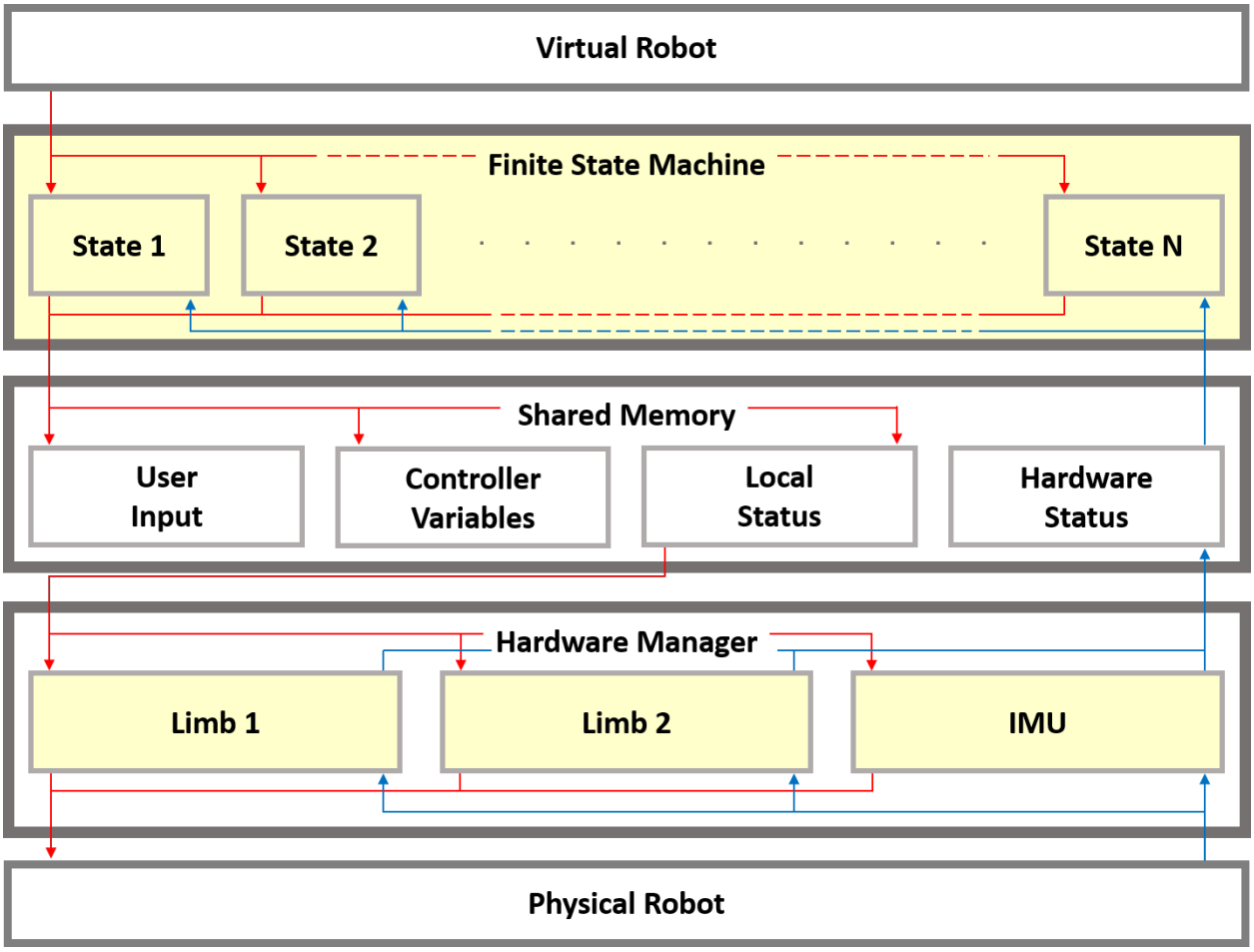


Figure 3.5: NABi-2 software architecture layout, with concurrently running processes highlighted in yellow.

control while retaining the aforementioned characteristics, the architecture is modified to maximize speed by restructuring the architecture layout such that it supports concurrency. The different modules are shown in Figure 3.5, with concurrently run processes highlighted.

The computer used to run NABi-2 is equipped with an Intel Core i5-7260U @ 2.2 GHz with 8 GB of RAM. NABi-2's software is written in Python 2.7 under Ubuntu 16.04, utilizing open-source libraries and optimized in-house built modules. Unlike other setups that may require a significant time from the user to prepare the machine, NABi-2 can be readily set up under a Python virtual environment. This allows the code base to remain simple for many people to quickly get involved with the development.

Modularity of the architecture also invites multiple people to work on the development with minimal merge conflicts. By abstracting the details of each controller in one or more states in a trampoline based finite state machine (FSM), multiple people can simultaneously work on multiple controllers. Operations in each state is standardized by passing between states a 'virtual robot' object that resides on the stack. Then, different controllers can independently calculate their respective inputs and command the virtual robot using standardized method calls. The robot object then spins once to update its attributes and execute necessary methods, which includes updating the shared memory segment which is imperative for a fast control loop.

Concurrently, a hardware manager that is a dedicated process for each chain of limbs is run. The manager indefinitely runs a while loop sequence of communication with the POSIX shared memory segments that: 1) Read the state of the hardware and, with the semaphore, update the shared memory block that the hardware manager is writing to, 2) If necessary,

with the semaphore, read from the shared memory block that the virtual robot writes to and write to the hardware. POSIX shared memory and semaphores were chosen to stay safe for potential multithreading and to keep semaphore overhead low. The hardware manager communicates with the hardware at maximum speed, and processes can be assigned to dedicated cores to further increase the communication frequency as seen in Table 3.3, which shows an average frequency over 10,000 communications between the hardware manager and the two chains of limbs. Through this approach, we are able to achieve stable proprioceptive force control despite using a dynamically typed language with unpredictable delays.

Shared Core [Hz]		Dedicated Core [Hz]	
2026.99	2169.82	2289.47	2348.09

Table 3.3: Communication Frequency Comparison

3.3.4 Control Implementation

The main control loop used for the robot's locomotion is run at around 700Hz, but the actual control of current for position on the actuator is run at several kHz on the actuator microcontroller. That is to say, it is the job of the main Intel x86 CPU to perform tasks such as calculating inverse kinematics, differential kinematics (Jacobians), and dynamics. Once the algorithm input signals are calculated, they are sent to the ARM-based MCU's on the BEAR modules as reference signals for the actuator-level current/position loops to track. The dynamically typed Python may seem like an interesting choice for a task that is typically accomplished using a lower-level systems language, but importantly Python

allows for modularity and fast software prototyping. Once a particular algorithm has been tested and verified, it is often transferred to C++ and optimized for performance. Another consideration is the communication between the various processors on the system. For NABi-2, control signals are transmitted and received through a standard USB port and read by a microcontroller which converts the signal to RS-485 protocol sent to the actuators.

3.4 Force-Controlled Jumping

The overall design of NABi-2 makes it an ideal platform for pursuing force/torque based control approaches, though position control can be used if desired. For NABi-2, force control simply involves defining some desired force to be exerted by the end effector, then converting these forces to joint torques by first rotating the forces into the robot frame and then multiplying by a Jacobian transposed:

$$\boldsymbol{\tau} = \boldsymbol{J}_c^T \boldsymbol{R}_{BN} \boldsymbol{f} \tag{3.1}$$

Where $\boldsymbol{\tau}$ is a vector of joint torques, $\boldsymbol{J}_c$ is the contact Jacobian relating joint rates to end effector (foot) velocities, $\boldsymbol{R}_{BN}$ is the rotation matrix that rotates the inertial frame into the robot body frame, and $\boldsymbol{f}$ is the force exerted by the end effector. This relationship yields a fairly simple model of NABi-2 as a floating rigid body with massless legs that transfer forces from the ground, as shown in Figure 3.6. The lightweight leg design of NABi-2 facilitates this massless leg model, which is computationally simple and does not require very accurate measurements of link mass and inertia. The systems presented in [86, 95] also use the Jacobian relationship for similar reasons. The following section details a simple controller

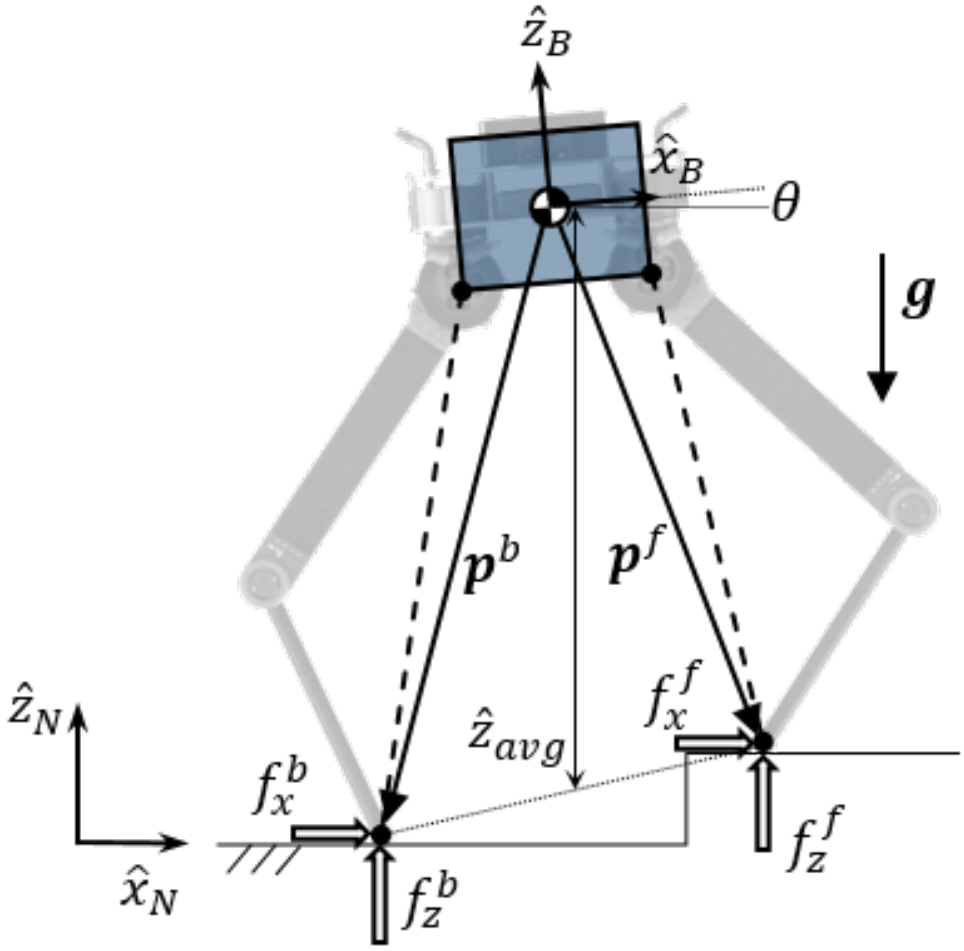


Figure 3.6: NABi-2 modeled as a floating rigid body with massless legs that transfer ground reaction forces to the body.

that effectively make use of the BEAR modules to achieve two-legged, in-place pronking on NABi-2.

3.4.1 Double Support

A key aspect of controlling legged robots is leveraging ground reaction forces at the feet to govern the system's overall posture and position. This is apparent when NABi-2 is standing

in double support, when it only has two point-foot contacts with the ground (when viewed orthogonal to the plane created by the legs). For this reason, an aggregated task-space PD controller was developed to control the ground reaction forces created by NABi-2s feet during double support. The first portion of the controller applies a normal force that counteracts the weight of the robot and performs task-space PD control around a nominal foot setpoint to account for disturbances and modeling inaccuracies.

$$\boldsymbol{f}^f_{grav} = \frac{p^b_x}{p^f_x + p^b_x} m\boldsymbol{g} \tag{3.2}$$

$$\boldsymbol{f}^b_{grav} = \frac{p^f_x}{p^f_x + p^b_x} m\boldsymbol{g} \tag{3.3}$$

$$\boldsymbol{f}^f_{ee} = \boldsymbol{K}_{p,ee}(\boldsymbol{p}^f_d - \boldsymbol{R}_{BN}\boldsymbol{p}^f) + \boldsymbol{K}_{d,ee}(\dot{\boldsymbol{p}}^f_d - \boldsymbol{R}_{BN}\dot{\boldsymbol{p}}^f) \tag{3.4}$$

$$\boldsymbol{f}^b_{ee} = \boldsymbol{K}_{p,ee}(\boldsymbol{p}^b_d - \boldsymbol{R}_{BN}\boldsymbol{p}^b) + \boldsymbol{K}_{d,ee}(\dot{\boldsymbol{p}}^b_d - \boldsymbol{R}_{BN}\dot{\boldsymbol{p}}^b) \tag{3.5}$$

Where the superscripts $i = \{f, b\}$ denote the front or back foot, $\boldsymbol{K}_{p,ee}$ and $\boldsymbol{K}_{d,ee}$ are diagonal gain matrices, $\boldsymbol{f}^i_{grav}$ and $\boldsymbol{f}^i_{ee}$ are the force contributions from gravity and end effector position feedback respectively, $\boldsymbol{p}^i_d = [p^i_{d,x}, p^i_{d,y}, p^i_{d,z}]^T$ and $\dot{\boldsymbol{p}}^i_d = [\dot{p}^i_{d,x}, \dot{p}^i_{d,y}, \dot{p}^i_{d,z}]^T$ are the desired position and velocity of the i^{th} end effector in the inertial frame, and $\boldsymbol{p}^i = [p^i_x, p^i_y, p^i_z]^T$ and $\dot{\boldsymbol{p}}^i = [\dot{p}^i_x, \dot{p}^i_y, \dot{p}^i_z]^T$ are the position and velocity vectors for the i^{th} end effector in the body frame. Note that the gravitational force is scaled by the normalized x components of the end effector positions, ensuring that no moment is created when accounting for the gravitational force. Now, double support over known terrain can be achieved by summing the gravity ($grav$) and end effector feedback (ee) contributions for each leg.

To achieve a balanced double support over unknown terrain, pitch (θ) and roll (ϕ) con-

tributions are added to the force being generated by the legs:

$$f_{pitch} = K_{p,\theta}(\theta_d - \theta) + K_{d,\theta}(\dot{\theta}_d - \dot{\theta}) \tag{3.6}$$

$$f_{roll} = K_{p,\phi}(\phi_d - \phi) + K_{d,\phi}(\dot{\phi}_d - \dot{\phi}) \tag{3.7}$$

For the pitch contribution, because the feet position cannot be predefined over unknown terrain, individual foot position feedback often resists the effort of the pitch controller. The solution to this was to take an average z position of the feet, z_{avg}, and replace the current z position of both foot with this average value, so $p_z^f = p_z^b = z_{avg}$ when the pitch controller is active over unknown terrain. This effectively creates a pivot point at the robot body that allows the legs to have different lengths. An example of the robot rejecting a sharp change in ground angle using the pitch controller is shown in Figure 3.7a.

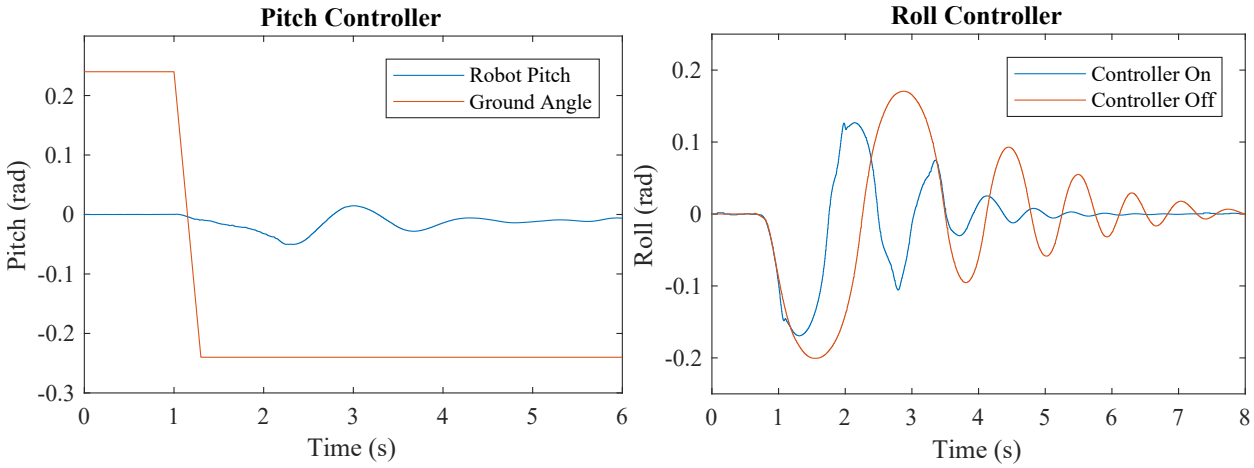
One consideration when implementing the roll controller is the fact that NABi-2 does not have an ankle joint, so any motion from the hip yaw joint will create an undesirable twisting motion of the foot on the ground. This resultant twisting combined with noise in the system can cause instabilities when the roll controller is active. However, once the robot has tipped significantly, the roll controller can help prevent a fall and/or recover from disturbances much faster as can be seen in Figure 3.7b. For this reason, the roll controller is only activated once a roll of 2 degrees is exceeded.

The total force $\boldsymbol{f}^i$ exerted by each leg can now be determined by simply summing the component controllers for each leg:

$$\boldsymbol{f}^f = \boldsymbol{f}^f_{grav} + \boldsymbol{f}^f_{ee} + [0, f_{roll}, f_{pitch}]^T \tag{3.8}$$

$$\boldsymbol{f}^b = \boldsymbol{f}^b_{grav} + \boldsymbol{f}^b_{ee} + [0, f_{roll}, -f_{pitch}]^T \tag{3.9}$$

The pitch contribution is added in equal and opposite magnitudes in the z component of the leg force to create a restorative moment about the y-axis, while the roll contribution is added to the y component of force to 'push' the robot back when it is tipping.



(a) (b)

Figure 3.7: The pitch controller (left) is able to reject a step of 0.5rad when the ground angle is suddenly shifted. The roll controller (right) is able to reduce the severity of an impulsive lateral kick; when the controller is applied, the system stabilizes more quickly and can also prevent the system from tipping over.

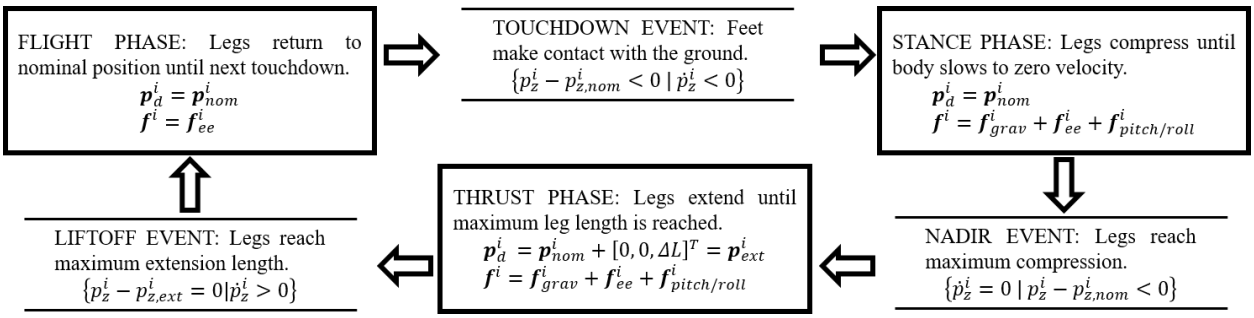


Figure 3.8: Diagram showing the continuous time phases and discrete events that are used to perform the two-legged pronking. In each phase, the desired foot setpoint $\boldsymbol{p}_d^i$ is set to either the nominal (nom) configuration or the extended (ext) configuration. In flight, only (3.4) and (3.5) need to be used, but in stance and thrust the full (3.8) and (3.9) are necessary.

3.4.2 Compositional Pronking Controller

Continuous jumping on a legged robots has been an active field of research since Raibert instituted his event-based heuristic controller on a prismatic pneumatic actuator monoped [97]. Many systems have since been implemented to follow Raiberts paradigm, and the so called Raibert Controller is now commonly implemented to demonstrate the efficacy of various compliant and springy leg designs [51]. A similar approach is taken here to illustrate how force control on the legs of NABi-2 can be utilized to mimic a mechanical compliance.

The classical Raibert controller was an empirical controller that was comprised of three control modules that independently regulated different aspects of Raiberts hopping robots: the vertical jumping height, the horizontal velocity, and the orientation of the body. This sort of modular controller has been formalized as the composition of several templates in [27], which also discusses the assumptions necessary for stability.

The jumping controller utilized on NABi-2 takes inspiration from Raiberts seminal work,

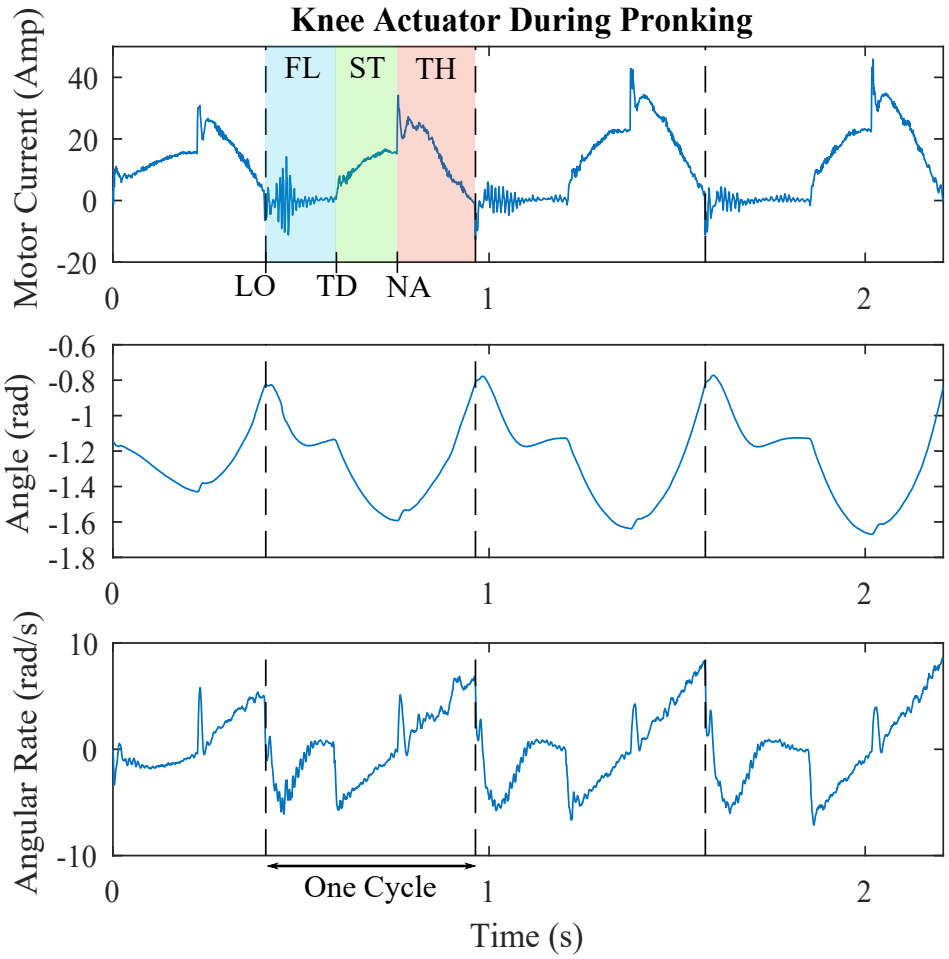


Figure 3.9: Plots showing the motor current, angle, and rate of one knee actuator while pronking. Discrete events are typically triggered by abrupt changes in at least one of these proprioceptive attributes. The discrete events are labeled LO, TD, and NA for liftoff, touchdown, and nadir, respectively; the continuous phases are FL, ST, and TH for flight, stance, and thrust, respectively. Note the sharp changes in value in one of the proprioceptive attributes during a discrete phase change.

but is modified slightly to accomplish in-place pronking; a form of jumping locomotion seen in nature where all feet are simultaneously lifted off and placed on the ground. For this form of jumping, NABi-2 can be modeled as a floating rigid body with massless spring-damper legs attached at the hip. These massless leg springs are easily implemented on the robot by providing a foot setpoint, $\boldsymbol{p}_j^i$, that matches the length of the leg-spring for leg i in phase j, and then tuning the PD gains of the associated PD controller to mimic desired spring-damper qualities. The use of massless, springy legs in simple running and jumping models has been well documented for several decades, seeing recognition in both the robotics and biology communities [11].

For the proposed controller, separate modules are used to control the thrust applied at each leg and the orientation of the body. Like the Raibert Controller, vertical height is controlled by applying a constant thrust in the stance phase. This thrust is achieved by increasing the nominal rest length of the leg-spring by ΔL, thus injecting energy into the system in the form of leg-spring potential energy. However, unlike the original Raibert Controller, the thrust does not occur as soon as the foot contacts the ground. Instead, thrust is only applied after the leg-spring reaches its full compression. At this point, the body has reached its lowest point, or nadir, in the jumping cycle, and the thrust is applied. Thrust begins at the nadir rather than at touchdown because this provides a longer thrust stroke and a protracted stance phase, producing a higher apex and providing more time for the the body orientation to stabilize. The body orientation is controlled by the PD controller described previously and is activated as soon as touchdown occurs. Touchdown occurs after both feet have contacted the ground and is determined by measuring sharp jumps in displacement and

velocity of the foot during flight.

The controller can be visualized using the diagram in Figure 3.8. Proprioceptive data (motor current, angle, and rate) from a knee actuator are shown in Figure 3.9 and demonstrate how the data is used to trigger events. When implementing the controller, the PD gains of the force controller as well as the thrust length of the jump were tuned empirically. The robot is able to stably jump continuously at a height of approximately 15cm as illustrated in Figure 3.10.

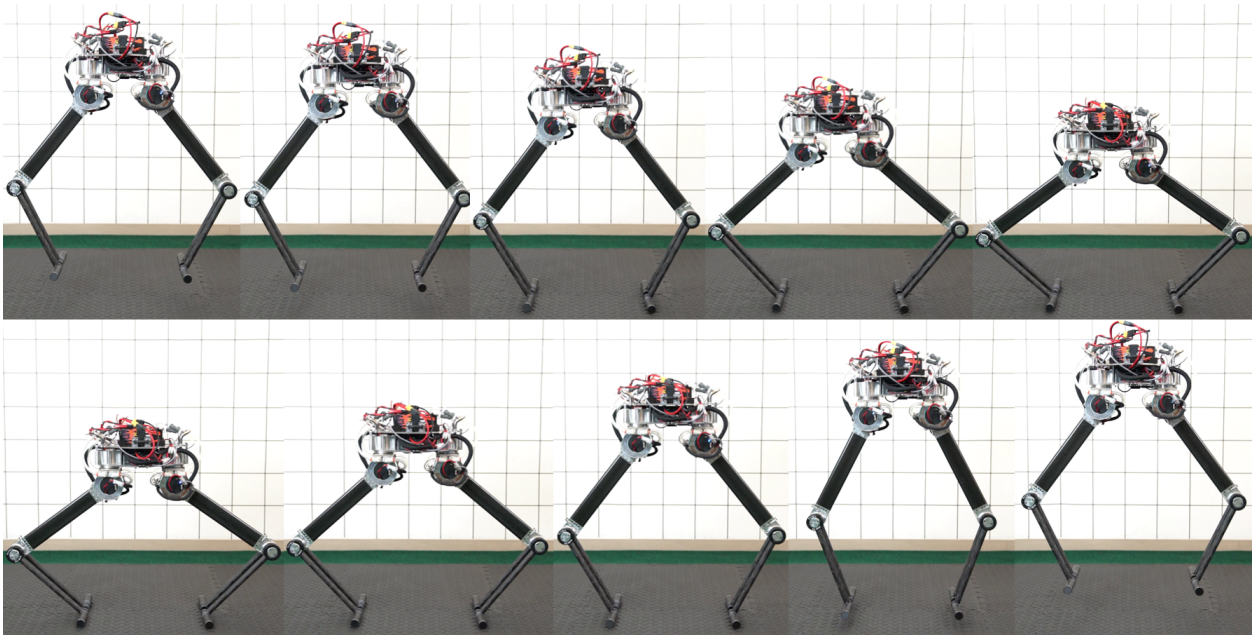


Figure 3.10: NABi-2 pronking sequence.

3.5 Discussion

3.5.1 The Energetic Cost of Compliance

While NABi-2 demonstrates the effectiveness of high torque, low gear ratio actuation at performing dynamic motions that require compliant behavior, it is worth noting that this approach does sacrifice some of the benefits of using heavily geared electric actuators. For example, the nominal energy consumption of the system standing at idle is significantly higher than that of the similarly sized NABi-1 system, which uses conventional servos. This is because the only way for NABi-2 to counteract gravity is for the BEAR modules to continuously provide torque (and thus, current) at the joints, while the traditional servos on NABi-1 can utilize the significant friction from the gearboxes in addition to the application of torque. However, when performing dynamic motions like pronking, it is possible for the BEAR module to actually regenerate significant amounts of current, something that is generally negligible on traditional servos. This means that NABi-2 has the potential to be more efficient when in motion, while NABi-1 is more efficient when stationary. In any case, the benefits of the BEAR modules still outweigh the drawbacks, as NABi-1 was never able to fully realize proprioceptive force control as NABi-2 is, despite it's series elastic leg design.

3.5.2 Walking and Directional Locomotion

On the original NABiRoS, the ZMP tracking approach using a cart-table model was used to generate time-dependent trajectories that would be 'played back' using high gain PD control on the servos of the robot. However, applying this joint-space approach on NABi-2 proved to

be wildly inconsistent, resulting in the robot taking anywhere between one and three steps before ultimately becoming 'out of sync' with the controller. The cause of this is likely the inherent compliance in the system reducing the fidelity of the position control capabilities in the form of unpredictable transients. Usually, these issues can be mitigated by having high damping which comes naturally with the high friction gearboxes of traditional servos, but is something that must be done through software on the BEAR modules, and is subject to factors such as sensor noise and control rate.

In the future, it would be possible to further optimize the robot control architecture and BEAR module firmware to allow for improved position control fidelity, but it may be more interesting and useful to instead embrace the compliant nature of the system and develop directional locomotion around it. Pronking already provides an interesting avenue of work that is reminiscent of Raibert's classical hoppers, and may be amenable to the foot touchdown controller that was used on his monopods to control horizontal speed. This idea could be further extended to both walking and potentially running, in which the system would somewhat resemble a quadruped bounding as seen from the side. However, in order for these sorts of motions to be implemented on the physical robot, it would be necessary to have some sort of state estimation that is able to supply the velocity and position of the system with respect to some inertial coordinate frame.

3.6 Conclusion

NABi-2 is a unique bipedal platform that deviates from traditional humanoid design in an attempt to provide a simple and robust platform for dynamic locomotion using propriocep-

tion and force control. The system's morphology, actuation, and software architecture are all designed around this premise, resulting in the adoption of things like non-anthropomorphic leg design, BEAR modules, and concurrently run software processes. The outcome of this particular combination of features yielded a versatile legged platform that is capable of performing pronking using a Raibert-style compositional controller, a feat typically reserved for systems with mechanical springs or other forms of physical compliance. However, it was also found that the inherent compliance of NABi-2 made traditional time and position dependent walking difficult, suggesting the advantages of using event-based controllers with the BEAR modules. Still, NABi-2 represents a step towards the development of fully realized humanoids that have the potential to be used in a variety of unstructured environments and scenarios thanks to advances in actuation technology.

CHAPTER 4

State Estimation for Legged Robots

Abstract

The control of robotic systems is largely dependent on feedback from joint position/velocity/torque sensors that provide information about the system kinematic and dynamic state. However, for mobile robots which are not fixed to a base in the world or inertial coordinate frame, it is necessary to estimate additional information regarding the state of the floating base in the inertial frame. When it comes to legged robots, the local linear and angular velocities of the body are of particular importance because they can be used to counteract external disturbances to the system and help prevent the robot from falling over. To accomplish this, legged robots typically make use of kinematic information provided by the leg encoders when in contact with the ground as well as accelerometer and gyroscope readings provided by an Inertial Measurement Unit (IMU). This chapter presents a brief overview of common approaches to state estimation for legged robots and then describes the implementation of an Extended Kalman Filter that fuses IMU readings with leg kinematic information in order to produce estimate of the egomotion of the body of NABi-2.

4.1 Introduction

Feedback control is reliant on a system's state, which are variables in the system model which describe the evolving behavior of the dynamical system. For some systems, the state is provided directly with sensors, but for many others the state needs to be estimated by using a sparse set of sensors and some knowledge of the dynamical model [124]. For robotic manipulator arms which are bolted to a fixed base, the state is typically described by the relative joint angles and velocities between each arm link, and are commonly measured directly using encoders. However, for a mobile system such as a legged robot or human, it is often desirable to additionally track the system's holistic motion with respect to some fixed origin. This so called *egomotion*, which was originally used to describe the 3D motion of a camera through an environment [26], is especially important for dynamic legged robots which need to constantly counteract the force of gravity in addition to external disturbances in order to maintain balance.

Looking at the broader robotics and autonomous systems community, the state estimation problem is tackled in a variety of ways. In mobile (wheeled) robots and self-driving cars which are in constant contact with the ground, it is common to use wheel revolutions as a simple measurement of distance, and then fuse additional information from accelerometers, cameras, range sensors (LiDAR), and GPS to estimate egomotion [18]. For flying micro aerial vehicles which are not in contact with the ground, an IMU is typically used in conjunction with any number of camera, range, and optical sensors for estimation [30]. Legged robots differ from wheeled and flying robots in the fact that they experience intermittent contact with the ground, meaning a combination of these two approaches may be applicable. As an aside,

an interesting comparison can be made between state estimation on legged robots and on their biological counterparts. Humans and animals make use of proprioceptive joint position and stress information from muscles, linear and angular acceleration information from the vestibular system (inner ear), and visual information from the eyes. Each of these systems has a direct counterpart on a legged robot, which are joint encoders and F/T sensors, an IMU, and cameras, respectively. Of course, the exact way biological systems make use of all these signals is still a topic of much research, but it at least suggests an approach to state estimation for legged systems.

This chapter presents a broad overview of some common approaches to legged state estimation that are generalizable to legged robots with arbitrary number of legs. The remainder of the chapter is organized as follows: Section 4.2 begins with a note on different sensing paradigms, as well as some theoretical prerequisites for the remainder of the chapter. Section 4.3 then describes how legged robots can be modeled for estimation. Section 4.4 then discusses the sensor options available, and Section 4.5 describes how they can be fused together for estimation. Section 4.6 talks about the implementation of one of these approaches on the physical NABi-2 platform. Section 4.7 closes with some concluding thoughts.

4.2 Prerequisites

4.2.1 Proprioceptive vs Exteroceptive

When discussing state estimation for mobile robots, there are a variety of approaches that have shown to be very successful. A very popular approach is the localization front end

of Simultaneous Localization and Mapping (SLAM), which is typically performed with a camera [26]. Today, with the ubiquity of small, low-cost cameras, it is becoming increasingly popular to perform estimation and localization with the help of Visual Odometry (VO), which tracks 3D landmarks in view of the camera and localizes the camera based on these features. In this context, the information obtained through the camera can be considered *exteroceptive*, as it relates to the external world around the robot. Camera data is often then fused with additional information coming from a LiDAR or IMU in order to accurately estimate the 3D pose of the system (and map the environment as well, in many cases). However, the downside of utilizing cameras are their relatively low update rates (though with recent advances in event camera based estimation this no longer is necessarily true [101]), and there is a tradeoff between computation and accuracy.

Legged robots are fortunate to have access to *proprioceptive* joint encoders, which are also low cost, exhibit very low biases compared to an IMU, and are typically always installed on legged robots due to the role they play in the low-level control of the joint actuators. Depending on how the kinematic leg encoder data is used, it can be very impactful on estimation without greatly increasing the computational load required. Ideally, visual, inertial, and kinematic data could all be fused, but the additional resources required would likely incur a penalty elsewhere, such as in the form of increased latency of the estimator or increased mass of the robot. The sensor fusion approach that is presented in the following mainly makes use of proprioceptive data from the encoders and an IMU, but this approach can be extended to make use of additional data through sensor fusion. Importantly, this approach yields velocity data at a high rate, which is crucial in being able to stabilize dynamic

motions on a robot.

4.2.2 Notation and Conventions

Throughout this chapter, scalars will be denoted by non-bold lowercase letters, e.g. s, vectors will be denoted by bold lowercase letters, e.g. $\boldsymbol{r}$, and matrices will be denoted by bold capital letters, e.g. $\boldsymbol{M}$. Coordinate frames will be represented with non-bold capital letters, e.g. W, and this notation will also be overloaded to represent a physical point in space; this is done so that a single coordinate frame and its associated origin point can be represented with a single letter.

To represent a vector (in Euclidian space) from point B to point C expressed in frame A, the notation ${}_A\boldsymbol{r}_{BC}$ is used, where the left subscript is the coordinate system the vector is represented in, and the right subscript describes the start and end points of the vector. For velocities, the vector $\boldsymbol{v}_B$ denotes the absolute (w.r.t. some fixed world/inertial frame) velocity of frame/point B, and the vector $\boldsymbol{a}_B$ for accelerations.

Rotations in this context will be represented by *Hamilton* convention quaternions or rotation matrices, but this approach is generalizable to other representations of $SO(3)$. The unit quaternion $\boldsymbol{q}_{BA}$ denotes the relative orientation of frame B with respect to frame A; it also can be thought of as the mapping: ${}_B\boldsymbol{r}_{BC} = (\boldsymbol{q}_{BA}){}_A\boldsymbol{r}_{BC}$, which conveniently places frames that will 'cancel' out adjacent to each other (ie. frame A in $\boldsymbol{q}_{BA}$ 'cancels' with frame A in ${}_A\boldsymbol{r}_{BC}$, resulting in $\boldsymbol{r}_{BC}$ being with respect to the remaining frame, B). The rotation matrix that performs the equivalent mapping as the quaternion $\boldsymbol{q}_{BA}$ is written as $\boldsymbol{C}(\boldsymbol{q}_{BA})$ The superscript $^\times$ is used to denote the skew symmetric matrix ${}_A\boldsymbol{v}^\times$ of a vector

${}_A\boldsymbol{v}$. Rotation arithmetic is represented with unique operators to differentiate them from normal Euclidian operations: addition $\boxplus$; subtraction $\boxminus$; multiplication $\otimes$. Rotational rates use $\boldsymbol{\omega}_{AB}$ which is the rotational rate of frame B w.r.t. frame A. Note that rotations in general are not commutative ($\boldsymbol{C}_{AB}\boldsymbol{C}_{BC} \neq \boldsymbol{C}_{BC}\boldsymbol{C}_{AB}$), and the rules for addition, subtraction, and differentiation are different. An informative introduction on the calculus of $SO(3)$ can be found in [15], and a thorough treatment of quaternions and their use for estimation is explained in [117].

When discussing filtering and sensor fusion, typically the tilde ($\tilde{}$) will be used to describe measurements, and the carat/hat ($\hat{}$) will be used to describe an estimated value. Most of the filters employ zero-mean Gaussian noise as the main stochastic model, so a multivariate random vector will be written as $\boldsymbol{n}_i$, where there index i will label the noise to a particular state. The associated covariance matrix is denoted as $\boldsymbol{R}_i$, so a full description of the normal distribution can be written as $\boldsymbol{n}_i \sim \mathcal{N}(0, \boldsymbol{R}_i)$

4.3 Modeling Legged Systems

The equations of motion for a general legged system with n DoF, m contact forces, and l joint torques/forces can be written as:

$$\boldsymbol{M}(\boldsymbol{\theta})\dot{\boldsymbol{u}} + \boldsymbol{b}(\boldsymbol{\theta}, \boldsymbol{u}) + \boldsymbol{g}(\boldsymbol{\theta}) + \boldsymbol{J}_c^T(\boldsymbol{\theta})\boldsymbol{F}_c = \boldsymbol{S}_a^T\boldsymbol{\tau} \tag{4.1}$$

Where $\boldsymbol{\theta}$ represents the generalized coordinates of the system, $\boldsymbol{F}_c \in \mathbb{R}^m$ are the (external) contact forces acting on the system, and $\boldsymbol{\tau} \in \mathbb{R}^l$ are the input joint torques/forces. Note that $\boldsymbol{\theta}$ may be over-parameterized due to the use of quaternions or rotation matrices, but

$\boldsymbol{u} \in \mathbb{R}^n$, $\dot{\boldsymbol{u}} \in \mathbb{R}^n$ the generalized velocities and accelerations of the system respectively, will be in the n dimensional space. The standard terms from the classical equations of motion for manipulators are the inertia matrix $\boldsymbol{M}(\boldsymbol{\theta}) \in \mathbb{R}^{n \times n}$, the velocity products (coriolis and centrigufal terms) $\boldsymbol{b}(\boldsymbol{\theta}, \boldsymbol{u}) \in \mathbb{R}^n$, and the gravity term $\boldsymbol{g}(\boldsymbol{\theta}) \in \mathbb{R}^n$. The remaining terms are the contact Jacobian $\boldsymbol{J}_c(\boldsymbol{\theta}) \in \mathbb{R}^{m \times n}$ which maps generalized velocities to contact point velocities, and the actuator selection matrix $\boldsymbol{S}_a \in \mathbb{R}^{l \times n}$ which distinguishes between actuated and non-actuated DoF.

These equations of motion describe the dynamics for a legged robot with a *floating base* B which is free to move around in the fixed world frame W. The pose of the floating base is parameterized by its position ${}_W\boldsymbol{r}_{WB}$ and attitude $\boldsymbol{q}_{BW}$ in the world frame, meaning the generalized coordinates $\boldsymbol{\theta}$ can be decomposed decomposed into:

$$\boldsymbol{\theta} = ({}_W\boldsymbol{r}_{WB}, \boldsymbol{q}_{BW}, \boldsymbol{\alpha}) \tag{4.2}$$

Where $\boldsymbol{\alpha}$ are values describing the internal configuration of the robot such as joint angles for revolute joints and link lengths for prismatic joints.

Additional constraints can be added to the dynamics assuming that there is no slipping at the contact points:

$$\boldsymbol{J}_c(\boldsymbol{\theta})\boldsymbol{u} = \boldsymbol{0} \tag{4.3}$$

This condition requires the velocity of the contact point (the foot) to be zero, verifying the no-slip condition. To incorporate this constraint into the second order dynamics of the system, the time-differentiated form of this constraint is often used:

$$\dot{\boldsymbol{J}}_c(\boldsymbol{\theta})\boldsymbol{u} + \boldsymbol{J}_c(\boldsymbol{\theta})\dot{\boldsymbol{u}} = \boldsymbol{0} \tag{4.4}$$

Equations 4.1 and 4.4 can be combined to simultaneously solve for the generalized accelerations as well as the contact forces:

$$\begin{bmatrix} \boldsymbol{M}(\boldsymbol{\theta}) & \boldsymbol{J}_c^T(\boldsymbol{\theta}) \\ \boldsymbol{J}_c(\boldsymbol{\theta}) & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{u}} \\ \boldsymbol{F}_c \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}(\boldsymbol{\theta}, \boldsymbol{u}) + \boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{S}_a^T \boldsymbol{\tau} \\ \dot{\boldsymbol{J}}_c(\boldsymbol{\theta}) \boldsymbol{u} \end{bmatrix} = \boldsymbol{0} \tag{4.5}$$

Note that in this formulation of the dynamics the contact forces are stated explicitly because legged robots can have dynamic contact conditions (e.g. feet are constantly lifted off and placed back down). This formulation is useful for simulation where all quantities are given except for the generalized accelerations $\dot{\boldsymbol{u}}$ and contact forces $\boldsymbol{f}_c$, but is not necessarily as useful for estimation since the the given quantities depend on the available sensor information. In the case that the contact forces are not given, they can be 'cancelled out' by utilizing a nullspace projection. This is done by left-multiplying the the first row of 4.5 by the right-nullspace matrix $\boldsymbol{N}_c(\boldsymbol{\theta})$ of the contact Jacobian which satisfies $\boldsymbol{J}_c(\boldsymbol{\theta})\boldsymbol{N}_c(\boldsymbol{\theta}) = \boldsymbol{0}$.

$$\boldsymbol{N}_c^T(\boldsymbol{\theta})\boldsymbol{M}(\boldsymbol{\theta})\dot{\boldsymbol{u}} + \boldsymbol{N}_c^T(\boldsymbol{\theta})\boldsymbol{b}(\boldsymbol{\theta}, \boldsymbol{u}) + \boldsymbol{N}_c^T(\boldsymbol{\theta})\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{N}_c^T(\boldsymbol{\theta})\boldsymbol{S}_a^T\boldsymbol{\tau} \tag{4.6}$$

This effectively removes the contact forces from the state and reduces the dimensions of the first row by the number of constraints m. The generalized accelerations can now be solved for with:

$$\dot{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{N}_c^T(\boldsymbol{\theta})\boldsymbol{M}(\boldsymbol{\theta}) \\ \boldsymbol{J}_c(\boldsymbol{\theta}) \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{N}_c^T(\boldsymbol{\theta})\boldsymbol{S}_a^T\boldsymbol{\tau} - \boldsymbol{N}_c^T(\boldsymbol{\theta})\boldsymbol{b}(\boldsymbol{\theta}, \boldsymbol{u}) - \boldsymbol{N}_c^T(\boldsymbol{\theta})\boldsymbol{g}(\boldsymbol{\theta}) \\ -\dot{\boldsymbol{J}}_c(\boldsymbol{\theta})\boldsymbol{u} \end{bmatrix} \tag{4.7}$$

4.3.1 Contacts

A joint between two rigid bodies can be viewed as a set of constraints on the relative motion between the two bodies. Similarly, contacts can be conceptualized as additional joints in

the system that constrain certain bodies in the system to the surrounding environment. These constraints can be written explicitly using using a set of (scalar) contact constraint functions concatenated into a vector, $\boldsymbol{c}(\boldsymbol{\theta}) = \mathbf{0}$. For legged robots, a contact constraint could describe the difference between the position of foot and the contact point on the ground. The contact Jacobian describes how one of these contact constraints changes with respect to the generalized coordinates, and is derived from the gradients of the contact constraint functions:

$$\boldsymbol{J}_c(\boldsymbol{\theta}) = \frac{\partial \boldsymbol{c}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{4.8}$$

This shows that the contact forces occur along the directions associated with gradients of the contact constraint functions. It is worth noting that this constraint-based view of contacts implies they are unilateral (non-penetrating) and infinite friction (non-slipping), which is reasonable when doing estimation because the underlying filter algorithms often intrinsically can handle slight deviations from these assumptions due to their stochastic nature. However, other approaches can explicitly detect slippage and account for it, such as in [13, 41, 106].

To that end, the two most common contact models for legged robots are the *point foot* model and the *flat foot* model. A point foot implementation on a robot is simply a small nub which is covered in rubber. The model of contact that this type of foot makes simply constrains the position of the foot to the contact point (except for the in the positive normal direction to the ground surface), leaving the orientation free. The contact constraint function is thus an equality constraint on the location of the contact point P in the world frame W:

$$\boldsymbol{c}(\boldsymbol{\theta}) = {}_W\boldsymbol{r}_{WP} - [{}_W\boldsymbol{r}_{WB} + \boldsymbol{C}(\boldsymbol{q}_{BW})^T {}_B\boldsymbol{r}_{BP}(\boldsymbol{\alpha})] = \mathbf{0} \tag{4.9}$$

Where ${}_W\boldsymbol{r}_{WP}$ is the position of the contact point in the world frame and ${}_B\boldsymbol{r}_{BP}(\boldsymbol{\alpha})$ is the

position of the foot in the body frame as defined by the robot kinematics via the internal configuration parameters $\boldsymbol{\alpha}$. The contact Jacobian is now:

$$\boldsymbol{J}_c(\boldsymbol{\theta}) = \begin{bmatrix} \mathbb{I}^3 & \boldsymbol{C}(\boldsymbol{q}_{BW})^T({}_B\boldsymbol{r}_{BP}(\boldsymbol{\alpha}))^\times & \boldsymbol{C}(\boldsymbol{q}_{BW})^T \frac{\partial}{\partial \boldsymbol{\alpha}}({}_B\boldsymbol{r}_{BP}(\boldsymbol{\alpha})) \end{bmatrix} \tag{4.10}$$

Where the first identity term comes from the body position, the second term from body orientation, and the third term from the internal configuration.

The flat foot implementation on a robot looks like a flat plate with a rubber sole (often found with humanoid robots). This model constrains both the position and the rotation of the foot, meaning the contact constraint function is an equality constraint on both the location and orientation of the contact point:

$$\boldsymbol{c}(\boldsymbol{\theta}) = \begin{bmatrix} {}_W\boldsymbol{r}_{WP} \\ \boldsymbol{q}_{PW} \end{bmatrix} - \begin{bmatrix} {}_W\boldsymbol{r}_{WB} + \boldsymbol{C}(\boldsymbol{q}_{BW})^T {}_B\boldsymbol{r}_{BP}(\boldsymbol{\alpha}) \\ \boldsymbol{q}_{PB}(\boldsymbol{\alpha}) \otimes \boldsymbol{q}_{BW} \end{bmatrix} \tag{4.11}$$

Where the quaternion $\boldsymbol{q}_{PB}(\boldsymbol{\alpha})$ describes the rotation between the body and the foot and is parameterized by the internal configuration through the kinematics, and $\boldsymbol{q}_{PW}$ is the orientation of the foot when it is laid flat on the contact surface. The contact Jacobian is then:

$$\boldsymbol{J}_c(\boldsymbol{\theta}) = \begin{bmatrix} \mathbb{I}^3 & \boldsymbol{C}(\boldsymbol{q}_{BW})^T({}_B\boldsymbol{r}_{BP}(\boldsymbol{\alpha}))^\times & \boldsymbol{C}(\boldsymbol{q}_{BW})^T \frac{\partial}{\partial \boldsymbol{\alpha}}({}_B\boldsymbol{r}_{BP}(\boldsymbol{\alpha})) \\ \boldsymbol{0} & \boldsymbol{C}(\boldsymbol{q}_{PB}(\boldsymbol{\alpha})) & \frac{\partial}{\partial \boldsymbol{\alpha}}\boldsymbol{q}_{PB}(\boldsymbol{\alpha}) \end{bmatrix} \tag{4.12}$$

4.4 Sensors

Many different kinds of sensors are utilized in legged robots, and for different purposes. Critically, in the real world, all of these sensors experience some form of noise, be it from electrical noise or discretization errors, and some also have constant offsets in their measure-

ments which are known as biases. One could naively just take the measurements from the sensors for use in their control algorithms, but this would often lead to issues in practice. As an illustrative example, consider controlling the position of an electric motor using an encoder. Theoretically, a simple PID controller should suffice to maintain a certain position, and the gains can be tuned to achieve a desired response. However, if the encoder being used to measure the position has high variance noise or very low resolution (high discretization), there may be large spikes in velocity when performing discrete differentiation, which would then cause large jumps in the contribution of the derivative term, degrading performance at best and sending too much current into the motor and melting the winding insulation at worst. These types of issues are why filtering and estimation are necessary for most systems being controlled, and why stochastic models of the sensors are needed to capture their behavior.

4.4.1 Encoders (Kinematic Sensors)

Perhaps the most common type of sensor that can be found on a legged robot is the kinematic sensor, which typically come in the form of joint encoders. Encoders can be revolute or prismatic, and can utilize optical, magnetic, and electric phenomena to encode a position or change in position. Encoders which maintain a constant origin through power cycles are absolute encoders, and ones that do not are incremental/relative encoders. Modern Hall-Effect sensors are a compelling option for absolute encoding of position, but can be more difficult to mount mechanically as it requires the careful placement of a rotating magnet. Optical encoders are often limited by resolution at the scale needed for legged robots, but

can be useful for measuring velocity. Potentiometers can be used for encoders, but are not ideal for legged robots as they are quickly worn out from the repetitive, abrupt motions experienced at the joints.

With respect to modeling, kinematic sensors are mostly used to measure the internal configuration of a robot and often are direct measurements of the corresponding generalized coordinates, $\boldsymbol{\alpha}$, representing revolute joint angles or prismatic joint lengths. A kinematic measurement $\tilde{\boldsymbol{z}}_{kin}$ can thus be written as:

$$\tilde{\boldsymbol{z}}_{kin} = \boldsymbol{f}_{kin}(\boldsymbol{\alpha}, \boldsymbol{n}_{\boldsymbol{\alpha}}) \tag{4.13}$$

Where $\boldsymbol{n}_{\boldsymbol{\alpha}}$ is the noise associated with the kinematic sensors and is often modeled as zero-mean discrete Gaussian noise with covariance $\boldsymbol{R}_{\boldsymbol{\alpha}}$. In the case where every DoF (joint) on the robot has an encoder, the measured generalized coordinates can be written as:

$$\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha} + \boldsymbol{n}_{\boldsymbol{\alpha}} \tag{4.14}$$

More complex noise models are rarely used as additive Gaussian noise more often than not produces sufficient results for state estimation purposes.

4.4.2 Force Sensors and Contact Sensors

Force (and torque) sensors can be employed to measure the internal force configuration of a robot, as well as the external contact forces. Force/Torque sensors come in all shapes and sizes, including optical, resistive, capacitive, and piezoelectric. However, these sensors are not commonly utilized for state estimation on legged robots for several reasons. One reason is that many of these sensors have large form factors and high mass, which reduces the payload

of the system for the same set of actuators. Force sensors also require precise calibration, which is often difficult or impossible to perform without the help of the manufacturer. For these reasons, force sensors are used with discretion, and often mainly to estimate contact. However, force sensors mounted near the foot to detect contact can be damaged or lose calibration after the robot has performed some dynamic motion, which restricts the motions done by the robot to be fairly conservative. If contact forces aren't measured directly, they can be approximated using the equations of motion, (4.1).

Force/Torque sensors can be modeled similarly to kinematic ones:

$$\tilde{\boldsymbol{z}}_{F/T} = \boldsymbol{f}_{F/T}(\boldsymbol{\tau}, \boldsymbol{f}_c, \boldsymbol{n}_{F/T}) \tag{4.15}$$

Where $\boldsymbol{\tau}$ are the internal forces/torques, $\boldsymbol{F}_c$ are the contact forces, and $\boldsymbol{n}_{F/T}$ is zero-mean discrete Gaussian noise with covariance $\boldsymbol{R}_{F/T}$.

Because the contact state of the robot is of paramount importance in legged state estimation, additional sensors can be utilized to detect contact when force sensors are not a compelling option. Because contact is a binary state, a simple mechanical switch or touch sensor can be used for detection. The drawback of this is that these switches can also break with dynamic motions, and many of them experience bounceback, though this can be filtered out for some cases. Another option that has become popular because of the development of proprioceptive actuators is to simply measure the current in the actuator with a shunt resistor, which should be proportional (or at least have a relatively smooth, low order mapping) to the torque. These torques are typically not accurate enough to be used for precise force estimation without additional sensors, but they can be effective for detecting contact. As a final resort, contact can be prescribed 'offline' and not detected with a sensor, thus making

assumptions of when the leg is expected to be on the ground. However, this approach is only effective for a well-known environment and should be used with discretion. An approach to contact detection that filters several different sensor modalities can be found in [10].

4.4.3 Inertial Measurement Unit

An inertial measurement unit (IMU) is a combined accelerometer and gyroscope which is used in many different types of vehicles for navigation and sensing. IMUs can also sometimes include magnetometers, barometers, and global positioning systems, but this increases their cost. The gyroscope measures the angular velocities of the sensor, and the accelerometer measures the accelerations of the sensor itself plus gravitational acceleration. These measurements are taken in the coordinate system attached to the sensor, which for convenience in notation will be collocated with the robot body origin at B. The resulting sensor measurements are ${}_B\tilde{\boldsymbol{f}}_{WB}$ for acceleration and ${}_B\tilde{\boldsymbol{\omega}}_{WB}$ for angular velocities. The current notation suggests the IMU being the kinematic center of the robot body, which is often chosen to be the case for simplicity. If the kinematic center is desired to be elsewhere, a transformation can be applied to the IMU measurements.

MEMS IMUs all experience two sources of measurement error: 1) (slowly varying) biases or offsets in the readings due to nonlinear effects and variations in temperature, and 2) high frequency electrical noise typical of MEMS sensors. High quality IMUs can be well calibrated from the manufacturer, but it is impossible to completely eliminate the bias and noise from the measurements, as minor variations in conditions over time will tend to degrade the calibration. As such, most IMU models take into account these measurement imperfections

using additive noise and bias on the IMU outputs. The resulting model can be described by:

$$ {}_B\tilde{\boldsymbol{f}}_{WB} = \boldsymbol{C}(\boldsymbol{q}_{BW})({}_W\boldsymbol{a}_B - {}_W\boldsymbol{g}) + {}_B\boldsymbol{b}_f + \boldsymbol{n}_f \tag{4.16} $$

$$ {}_B\tilde{\boldsymbol{\omega}}_{WB} = {}_B\boldsymbol{\omega}_{WB} + {}_B\boldsymbol{b}_\omega + \boldsymbol{n}_\omega \tag{4.17} $$

$$ {}_B\dot{\boldsymbol{b}}_f = \boldsymbol{n}_{bf} \tag{4.18} $$

$$ {}_B\dot{\boldsymbol{b}}_\omega = \boldsymbol{n}_{b\omega} \tag{4.19} $$

Where ${}_W\boldsymbol{g}$ is the gravitational acceleration expressed in the world frame W, bias terms ${}_B\boldsymbol{b}_f$ and ${}_B\boldsymbol{b}_\omega$ model the measurement offset, and the additive noise terms $\boldsymbol{n}_f$, $\boldsymbol{n}_\omega$ model the high frequency measurement noise. Note that the relatively slow changes in the biases are modeled as Brownian Motion processes (Random Walk in discrete time), which are generated from the integration of the bias noise parameters, $\boldsymbol{n}_{bf}$ and $\boldsymbol{n}_{b\omega}$. The covariances of the these parameters and their descriptions are shown below:

Parameter	Symbol	Units
Accelerometer White Noise	$\boldsymbol{R}_f$	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$
Gyroscope White Noise	$\boldsymbol{R}_\omega$	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$
Accelerometer Random Walk	$\boldsymbol{R}_{bf}$	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Gyroscope Random Walk	$\boldsymbol{R}_{b\omega}$	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$

Table 4.1: Covariance values and units for IMU model noise parameters.

The white noise values $\boldsymbol{R}_f$ and $\boldsymbol{R}_\omega$ can typically be found in the datasheet of an IMU, and are commonly called *noise density*, or, confusingly, *angular random walk/velocity random walk*, due to the fact that in navigation applications the integration of the white noise values

lead to a random walk in angles and velocity. The values $\boldsymbol{R}_{bf}$ and $\boldsymbol{R}_{b\omega}$ are rarely specified in datasheets, as in reality the bias does not truly behave as a random walk. Instead, the specification commonly provided is the *in-run bias (in)stability* or just *bias stability*, which describes approximately how accurately the bias can be estimated. Lower values of the bias stability parameter are desirable as this implies fewer deviations from the mean over time. If it is assumed the noise of the IMU is dominated by a combination of white noise and a random walk, then the bias stability can be used to determine reasonable values for $\boldsymbol{R}_{bf}$ and $\boldsymbol{R}_{b\omega}$. This is typically done using the Allan standard deviation, and is described in detail in [16]. Realistically, this sensor model is fairly optimistic, as it describes a stationary sensor under constant environmental conditions. Because of this, it can be useful in practice to use slightly larger values for the covariances to capture additional sources of error [39].

The attitude/orientation of the IMU can be estimated using only the measurements from the IMU, so often times an integrated IMU package will provide the attitude as an additional measurement output. Attitude estimation algorithms on IMUs often employ the assumption that the average acceleration over time is zero, which allows the measured gravity vector to be used to correct the angle obtained from time-integrating the gyroscope. For legged robots, this assumption is reasonable for conservative motions, but can be problematic for very dynamic motions which have both fast linear and fast angular components. Furthermore, the accuracy of the attitude estimate will be heavily affected by the measurement bias, as the bias will be repeatedly added onto the latest attitude estimate when the gyroscope measurements are integrated. For the case of linear velocity and position, there are typically no additional on-board measurements (IMUs with GPS are an exception) which can be used

to correct the accumulated bias errors or drift when integrating the accelerometer, which is why this information must be obtained through other means like a camera or the encoders on a legged robot.

4.4.4 Cameras and Range Sensors

Cameras and sensors that detect range such as LiDAR, RADAR, and SoNAR are some of the most effective sensors for determining position and yaw with respect to an absolute origin. For legged robots navigating human-like environments in the atmosphere, cameras and LiDAR are the most applicable. Cameras today are excellent for providing dense information (thousands if not millions of pixels) at a very low monetary, mass, and power cost. However, their performance degrades significantly when performing fast motions and in regions with poor lighting conditions, limited texture, or a high number of airborne particulates. Range sensors like LiDAR tend to have (comparatively) high monetary, mass, and power costs, but they provide much richer depth information than cameras and are not affected greatly by lighting conditions. Two additional types of cameras that are occasionally used in estimation in harsh conditions are thermal cameras and event cameras. Thermal cameras use infrared radiation to form a heat zone image and are not sensitive to lighting or particulates. Event cameras use arrays of pixels which independently and asynchronously respond to changes in brightness at high temporal speeds, meaning they do not suffer from motion blur.

Utilizing cameras and range sensors for estimation typically requires complex algorithms which calculate the state from a series of images or a cloud of points obtained through a laser scan. There is a large body of work on visual odometry approaches with cameras that show

impressive results [83, 37]. These approaches can be improved by also incorporating the IMU in visual-inertial odometry [30]. Localization approaches for laser range sensors typically utilize point cloud matching for localization such as in iterative closest point algorithms [90].

4.5 Sensor Fusion

If information from several sensors are available, all the measurements can be fused together in order to provide a better state estimate than with a single sensor alone. In its simplest form, sensor fusion is simply the process of taking a weighted average of sensor measurements in order to achieve a more accurate overall estimate of the state. For sensor measurements that are believed to be more accurate, you assign a higher weight, and for sensor measurements that are believed to be less accurate, a lower weight. This accuracy is usually characterized using statistics, such as by looking at the variance of a normal distribution. The Kalman Filter [67] is perhaps the most popular formalization of this idea, and is widely used in many areas of robotics. However, manually tuned complimentary filters are also a solution used in some applications where the noise properties of the sensors used are well known and consistent.

The weighted combination of different measurements is the defining feature of sensor fusion, but how these measurements are combined also matters. In sensor fusion, there is the notion of *loose* or *tight* fusion. Loose fusion describes a procedure in which the sensor measurements are first processed before being fused together, whereas tight fusion describes a procedure where the sensor measurements are (as much as possible) directly used for updating the state. As an example, loose fusion could involve extracting a position estimate

from several sensor measurements (such as an IMU, camera, and leg kinematics) and then fusing them together. Tight fusion with the same sensors would involve taking the IMU accelerations, camera pixel values, and joint encoder readings and combining all of these in order to produce one single position estimate. Loose fusion can be simpler to implement, but tight fusion tends to produce better results because they take advantage of measurement models better and often reduce the amount of assumptions being made.

4.5.1 (Nonlinear) Kalman Filters

The Kalman filter is a stochastic filter that performs sensor fusion by producing optimal gains (weights) which minimize the mean square error of the estimate according to the covariances of the measurements and state [46]. The classic Kalman Filter is defined for linear systems, but the estimation problems posed by many mobile robots (including legged ones) do not adhere to this assumption, so a nonlinear extension is required. The aptly named *Extended Kalman Filter* [78] approximates nonlinear system dynamics by taking a local linearization (first order Taylor expansion) around the current state estimate, and then proceeds to carry out the Kalman Filter steps. Another popular nonlinear variant is the *Unscented Kalman Filter* [129], which approximates nonlinear system dynamics by sampling several points (called *sigma points*), running them through the nonlinear dynamics, and using the outputs to create a new distribution which is used for the filter update. There exist many other Kalman filter variants as well as non-Gaussian filters (such as the Particle Filter [126]), but for legged robots the EKF and UKF are popular due to their low memory usage due to being recursive and their low latency because of the (relatively) low number of

computations being performed.

4.5.1.1 Discrete-Time Extended Kalman Filter

Kalman Filters in discrete-time are typically implemented with two or more steps:

1. A high rate *process/prediction* step that simulates a simple model or processes high frequency sensor data (like from an IMU).

2. One or more *update/correction* steps that fuse in lower frequency (but typically more accurate) sensor data to correct errors accumulated in prior process/prediction steps.

The output of the process step is the predicted state estimate vector $\hat{\boldsymbol{x}}_{k+1}^{-}$ (also called the *a-priori estimate* or *predicted belief*), and the output of the update step is the posterior state estimate/belief vector $\hat{\boldsymbol{x}}_{k+1}^{+}$. The posterior estimate is obtained by taking the predicted estimate and adding on the *innovation* term $\boldsymbol{y}_{k+1}$, which is what is correcting the errors from prediction:

$$\hat{\boldsymbol{x}}_{k+1}^{-} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k}^{+}, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{n}_{f,k}) \tag{4.20}$$

$$\boldsymbol{y}_{k+1} = \boldsymbol{h}(\hat{\boldsymbol{x}}_{k+1}^{-}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{n}_{h,k+1}) \tag{4.21}$$

$$\hat{\boldsymbol{x}}_{k+1}^{+} = \hat{\boldsymbol{x}}_{k+1}^{-} \boxplus \boldsymbol{y}_{k+1} \tag{4.22}$$

Note that the predicted state is a function of the previous posterior state estimate $\hat{\boldsymbol{x}}_{k}^{+}$, process measurements $\tilde{\boldsymbol{z}}_{f,k}$, and process noise $\boldsymbol{n}_{f,k}$, and the innovation is a function of the current predicted state estimate $\hat{\boldsymbol{x}}_{k+1}^{-}$, update measurements $\tilde{\boldsymbol{z}}_{h,k+1}$, and measurement noise $\boldsymbol{n}_{h,k+1}$. The operator $\boxplus$ is used here to remind the reader that orientations must be composed

accordingly. Also note that in this formulation, the update model is slightly generalized and directly outputs the innovation term instead of performing the standard procedure which takes the difference between actual and expected measurements and gains them with the Kalman Gain $\boldsymbol{K}_{k+1}$, e.g. $\boldsymbol{y}_{k+1} = \boldsymbol{K}_{k+1}[\tilde{\boldsymbol{z}}_{h,k+1} - \boldsymbol{h}'(\hat{\boldsymbol{x}}^-_{k+1}, \boldsymbol{n}_{h,k+1})]$. As a general rule of thumb, every sensor measurement used in the update step should produce an innovation term based on the measurement and the predicted state estimate.

The Extended Kalman Filter makes use of the nonlinear equations (4.20) - (4.22) by taking their local linearizations about the current best state estimate. These linearizations are done using the first order Taylor expansion, and are often referred to the as the *Jacobians* of the EKF:

$$\boldsymbol{F}_k = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}_k, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{n}_{f,k})}{\partial \boldsymbol{x}_k}\right|_{(\hat{\boldsymbol{x}}^+_k, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{0})} \tag{4.23}$$

$$\boldsymbol{G}_k = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}_k, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{n}_{f,k})}{\partial \boldsymbol{n}_{f,k}}\right|_{(\hat{\boldsymbol{x}}^+_k, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{0})} \tag{4.24}$$

$$\boldsymbol{H}_{k+1} = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}_{k+1}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{n}_{h,k+1})}{\partial \boldsymbol{x}_{k+1}}\right|_{(\hat{\boldsymbol{x}}^-_{k+1}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{0})} \tag{4.25}$$

$$\boldsymbol{J}_{k+1} = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}_{k+1}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{n}_{h,k+1})}{\partial \boldsymbol{n}_{h,k+1}}\right|_{(\hat{\boldsymbol{x}}^-_{k+1}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{0})} \tag{4.26}$$

Where $\boldsymbol{F}_k$ and $\boldsymbol{G}_k$ are the process/prediction Jacobians and $\boldsymbol{H}_{k+1}$ and $\boldsymbol{J}_{k+1}$ are the update Jacobians. The last piece used in the Kalman Filter is the covariance $\boldsymbol{P}_k$ which tracks the level of uncertainty of each state and measurement in order to best determine how to fuse them together. In practice, this covariance need not be of the exact state; it can be from some related value that still contains information about the uncertainty of the state. The covariances value will also have a prediction step and an update step, resulting in the the predicted covariance $\boldsymbol{P}^-_{k+1}$ and the updated covariance $\boldsymbol{P}^+_{k+1}$.

The (recursive) filter equations can now be stated with the prediction step:

$$\hat{\boldsymbol{x}}_{k+1}^{-} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k}^{+}, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{0}) \tag{4.27}$$

$$\boldsymbol{P}_{k+1}^{-} = \boldsymbol{F}_k \boldsymbol{P}_k^{+} \boldsymbol{F}_k^{T} + \boldsymbol{G}_k \boldsymbol{R}_{f,k} \boldsymbol{G}_k^{T} \tag{4.28}$$

And the update step:

$$\boldsymbol{S}_{k+1} = \boldsymbol{H}_{k+1} \boldsymbol{P}_{k+1}^{-} \boldsymbol{H}_{k+1}^{T} + \boldsymbol{J}_{k+1} \boldsymbol{R}_{h,k+1} \boldsymbol{J}_{k+1}^{T} \tag{4.29}$$

$$\boldsymbol{K}_{k+1} = \boldsymbol{P}_{k+1}^{-} \boldsymbol{H}_{k+1}^{T} \boldsymbol{S}_{k+1}^{-1} \tag{4.30}$$

$$\boldsymbol{P}_{k+1}^{+} = (\mathbb{I}_3 - \boldsymbol{K}_{k+1} \boldsymbol{H}_{k+1}) \boldsymbol{P}_{k+1}^{-} \tag{4.31}$$

$$\boldsymbol{y}_{k+1} = \boldsymbol{h}(\hat{\boldsymbol{x}}_{k+1}^{-}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{0}) \tag{4.32}$$

$$\hat{\boldsymbol{x}}_{k+1}^{+} = \hat{\boldsymbol{x}}_{k+1}^{-} \boxplus \boldsymbol{y}_{k+1} \tag{4.33}$$

4.5.1.2 Utilizing the Error State

Local linearizations of nonlinear systems is a common practice in many fields (including Kalman Filtering) due to the strong results available for linear systems. However, for the EKF, these linear approximations can degrade the performance of the filter, as the linearized matrices propagating the covariance only produce optimal Kalman gains for the local linear system, not for the actual nonlinear system. To alleviate this issue, one can utilize the *error state* rather than the actual state to propagate the covariance of the system. The resulting filter is sometimes called the *error state Kalman Filter* (ESKF) [117].

The *error state* $\delta\boldsymbol{x}_k$ in the ESKF describes the difference between a *true* (i.e. actual state

with noise and errors, $\boldsymbol{x}_{t,k}$) and a *nominal* (i.e. idealized without errors, $\boldsymbol{x}_k$) state:

$$\delta \boldsymbol{x}_k = \boldsymbol{x}_{t,k} \boxminus \boldsymbol{x}_k \tag{4.34}$$

$$\boldsymbol{x}_{t,k} = \boldsymbol{x}_k \boxplus \delta \boldsymbol{x}_k \tag{4.35}$$

The main benefit of utilizing the error state in the Kalman Filter comes from the fact that the error state dynamics are relatively small and slow (compared to the actual dynamics), which means the linear approximations hold much better. The result of this is that the covariance propagated by the more-linear error state dynamics provide a much better estimate of the expected uncertainty, meaning the resulting Kalman gain will perform a better job at fusing the different data.

In practice, utilizing the ESKF adds a bit more complexity to the implementation, but can improve performance significantly. First, the error state dynamics are derived by subtracting the nominal, deterministic dynamics of the prediction step from the true stochastic ones:

$$\delta \boldsymbol{x}_{k+1} = \boldsymbol{x}_{t,k+1} \boxminus \boldsymbol{x}_{k+1} \tag{4.36}$$

$$= \boldsymbol{f}_e(\boldsymbol{x}_k, \delta \boldsymbol{x}_k, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{n}_{f,k}) \tag{4.37}$$

Then, the Jacobians of the error state dynamics are found by taking the partial of the error state dynamics with respect to the error state, and evaluating the first term in the Taylor expansion at the current state estimate.

$$\boldsymbol{F}_k = \left. \frac{\partial \boldsymbol{f}_e(\boldsymbol{x}_k, \delta \boldsymbol{x}_k, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{n}_{f,k})}{\partial \delta \boldsymbol{x}_k} \right|_{(\hat{\boldsymbol{x}}_k^+, \boldsymbol{0}, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{0})} \tag{4.38}$$

$$\boldsymbol{G}_k = \left. \frac{\partial \boldsymbol{f}_e(\boldsymbol{x}_k, \delta \boldsymbol{x}_k, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{n}_{f,k})}{\partial \boldsymbol{n}_{f,k}} \right|_{(\hat{\boldsymbol{x}}_k^+, \boldsymbol{0}, \tilde{\boldsymbol{z}}_{f,k}, \boldsymbol{0})} \tag{4.39}$$

The resulting error state prediction Jacobians are utilized the propagate the covariance in the prediction step. For the update step, the measurement model is differentiated with respect to the error state as well, yielding error state update Jacobians which are used for the measurement update:

$$\boldsymbol{H}_{k+1} = \frac{\partial \boldsymbol{h}(\boldsymbol{x}_{k+1}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{n}_{h,k+1})}{\partial \delta \boldsymbol{x}_{k+1}}\bigg|_{(\hat{\boldsymbol{x}}^-_{k+1}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{0})} \tag{4.40}$$

$$= \frac{\partial \boldsymbol{h}_{k+1}}{\partial \boldsymbol{x}_{k+1}}\bigg|_{(\hat{\boldsymbol{x}}^-_{k+1}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{0})} \frac{\partial \boldsymbol{x}_{k+1}}{\partial \delta \boldsymbol{x}_{k+1}}\bigg|_{(\hat{\boldsymbol{x}}^-_{k+1}, \tilde{\boldsymbol{z}}_{h,k+1}, \boldsymbol{0})} \tag{4.41}$$

Where $\boldsymbol{J}_{k+1}$ is unchanged, and the calculation of $\boldsymbol{H}_{k+1}$ can be done with the chain rule as shown. The innovation term can now be viewed as the estimated error state, which is then injected into the nominal state to produce the posterior state estimate. It is important to note that once the error state Jacobians are derived and used to propagate the covariance, it is not necessary to actually propagate the error state dynamics [117].

4.5.2 IMU and Leg Kinematics Fusion

The EKF/ESKF described previously is now used to (tightly) fuse the sensor readings from an IMU and the joint encoders which provide kinematic information about the legs. This fusion approach has the advantages of being able to handle uneven terrain better than approaches that simply rely on a single sensory input, such as the leg kinematics. This is possible thanks to the fact that the filter is simultaneously estimating the main pose of the body as well as the locations of the foot contacts. Interestingly, this approach can be viewed as a SLAM algorithm, where the feet position represent the map features which the robot is localizing with respect to [14].

4.5.3 Filter State Definition

The state of the filter used for the EKF fusing the leg kinematics and IMU can be thought of as three groups of variables. The first group describes how the main body of the robot is moving with respect to the world, and consists of the position ${}_W\boldsymbol{r}_{WB}$, velocity ${}_W\boldsymbol{v}_B$, and orientation $\boldsymbol{q}_{BW}$ of the main robot body. A quaternion is used here as a reduced representation of orientation that does not have any singularities. The second group is comprised of the 3D locations of the foot contact points in the world coordinate frame ${}_W\boldsymbol{r}_{WF_i}$. The third group consists of the IMU biases that are also being estimated by the filter. Put explicitly, the full system state $\boldsymbol{x}$ is:

$$\boldsymbol{x} = ({}_W\boldsymbol{r}_{WB}, {}_W\boldsymbol{v}_B, \boldsymbol{q}_{BW}, {}_W\boldsymbol{r}_{WP_1}, \ldots, {}_W\boldsymbol{r}_{WP_N}, {}_B\boldsymbol{b}_f, {}_B\boldsymbol{b}_\omega) \tag{4.42}$$

$$= (\boldsymbol{r}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N, \boldsymbol{b}_f, \boldsymbol{b}_\omega) \tag{4.43}$$

where again:

- ${}_W\boldsymbol{r}_{WB}$ is the position of the body frame B w.r.t. the world frame W as in 4.2.
- ${}_W\boldsymbol{v}_B$ is the velocity of B w.r.t. W.
- $\boldsymbol{q}_{BW}$ is the orientation of the B w.r.t. W as in 4.2.
- ${}_W\boldsymbol{r}_{WP_i}$ is the position of the i^{th} foot contact point P_i w.r.t. W as in 4.9.
- ${}_B\boldsymbol{b}_f$ is the accelerometer bias expressed in B as in 4.16.
- ${}_B\boldsymbol{b}_\omega$ is the gyroscope bias expressed in B as in 4.17.

The uncertainties of the estimated state are represented with the covariance matrix $\boldsymbol{P}$ of the cooresponding state error vector $\delta\boldsymbol{x}$:

$$\boldsymbol{P} = Cov(\delta\boldsymbol{x}) \tag{4.44}$$

$$\delta\boldsymbol{x} = (\delta\boldsymbol{r}, \delta\boldsymbol{v}, \delta\boldsymbol{\phi}, \delta\boldsymbol{p}_i, \ldots, \delta\boldsymbol{p}_N, \delta\boldsymbol{b}_f, \delta\boldsymbol{b}_\omega) \tag{4.45}$$

Note the use of $\delta\boldsymbol{\phi}$, which is a 3D rotation vector that represents the error in orientation. The rotation vector is used here to enable a minimal representation of the rotations in the covariance matrix. To go between quaternion error and rotation vector error:

$$\delta\boldsymbol{q} = \begin{bmatrix} \sin\left(\frac{1}{2}||\delta\boldsymbol{\phi}||\right)\frac{\delta\boldsymbol{\phi}}{||\delta\boldsymbol{\phi}||} \\ \\ \cos\left(\frac{1}{2}||\delta\boldsymbol{\phi}||\right) \end{bmatrix} \tag{4.46}$$

Where quaternion error $\delta\boldsymbol{q}$ is defined by:

$$\boldsymbol{q} = \delta\boldsymbol{q} \otimes \boldsymbol{q}_t \tag{4.47}$$

Where $\otimes$ is the quaternion multiplication operator.

4.5.4 Prediction Model

The prediction model used to propagate the prediction step utilize the IMU accelerometer $\tilde{\boldsymbol{f}}$ and gyroscope $\tilde{\boldsymbol{\omega}}$ measurements as described in (4.16) - (4.19) (with coordinate frame

subscripts removed for clarity) and are written in continuous time as follows:

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{4.48}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{a} = \boldsymbol{C}(\boldsymbol{q})^T(\tilde{\boldsymbol{f}} - \boldsymbol{b}_f - \boldsymbol{n}_f) + \boldsymbol{g} \tag{4.49}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\Omega}(\tilde{\boldsymbol{\omega}} - \boldsymbol{b}_\omega - \boldsymbol{n}_\omega)\boldsymbol{q} \tag{4.50}$$

$$\dot{\boldsymbol{p}}_i = \boldsymbol{C}(\boldsymbol{q})^T\boldsymbol{n}_{p,i} \tag{4.51}$$

$$\dot{\boldsymbol{b}}_f = \boldsymbol{n}_{bf} \tag{4.52}$$

$$\dot{\boldsymbol{b}}_\omega = \boldsymbol{n}_{b\omega} \tag{4.53}$$

Where $\boldsymbol{\Omega}(\cdot)$ maps a rotational rate $\boldsymbol{\omega}$ to a matrix representing the corresponding quaternion rate:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{4.54}$$

And where $\boldsymbol{n}_{p,i}$ is white noise with covariance $\boldsymbol{Q}_{p,i}$ added to the location of the foot, and is included in order to account for a certain amount of foot slippage. This value also allows for simple resetting of the foot position when it comes off the ground by setting the covariance to be very large (infinity) when the foot comes off the ground. $\boldsymbol{Q}_{p,i}$ itself is simply a diagonal matrix which can be tuned.

4.5.5 Measurement Model

The measurement model used in the update step is based on the simple model for kinematics from encoders as in 4.13. In this case, the position of the foot with respect to the body $\boldsymbol{s}_i$ (in the body frame) is defined by the kinematics as a function of joint angles $\boldsymbol{\alpha}$:

$$\boldsymbol{s}_i = \boldsymbol{f}_{kin,i}(\boldsymbol{\alpha}) + \boldsymbol{n}_{s,i} \tag{4.55}$$

Where the noise term $\boldsymbol{n}_{s,i}$ is included here to account for small errors in the kinematics. Now, the measured position $\tilde{\boldsymbol{s}}_i$ can be written as

$$\tilde{\boldsymbol{s}}_i = \boldsymbol{f}_{kin,i}(\tilde{\boldsymbol{\alpha}}) \tag{4.56}$$

$$\approx \boldsymbol{f}_{kin,i}(\boldsymbol{\alpha}) + \boldsymbol{J}_{kin,i}\boldsymbol{n}_{\boldsymbol{\alpha}} \tag{4.57}$$

$$\approx \boldsymbol{s}_i - \boldsymbol{n}_{s,i} + \boldsymbol{J}_{kin,i}\boldsymbol{n}_{\boldsymbol{\alpha}} \tag{4.58}$$

$$\approx \boldsymbol{s}_i + \boldsymbol{n}_i \tag{4.59}$$

Where $\boldsymbol{J}_{kin,i}$ is the leg Jacobian that relates the foot velocities (w.r.t. the body) to the joint velocities for leg i:

$$\boldsymbol{J}_{kin,i} := \frac{\partial \boldsymbol{f}_{kin,i}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}_i} \tag{4.60}$$

And the linearized noise effect from the encoders ($\boldsymbol{J}_{kin,i}\boldsymbol{n}_{\boldsymbol{\alpha}}$) and the noise from the foothold positions ($\boldsymbol{n}_{s,i}$) are combined into a new noise term, $\boldsymbol{n}_i$, with new covariance $\boldsymbol{R}_i$:

$$\boldsymbol{R}_i = \boldsymbol{R}_s + \boldsymbol{J}_{kin,i}\boldsymbol{R}_{\boldsymbol{\alpha}}\boldsymbol{J}_{kin,i}^T \tag{4.61}$$

The foot measurement can then be transformed from a measurement in the body frame to a measurement in the world frame by subtracting the absolute positon of the body from the

absolute position of the foot (and adding noise):

$$\tilde{\boldsymbol{s}}_i = \boldsymbol{C}(\boldsymbol{q})(\boldsymbol{p}_i - \boldsymbol{r}) + \boldsymbol{n}_i \tag{4.62}$$

Note that in implementation, the actual value of $\tilde{\boldsymbol{s}}_i$ is simply obtained by doing the forward kinematics of the leg with the joint encoder readings, as in (4.56). The filter Jacobians are omitted for brevity but the can be found in [14].

4.6 Estimation on NABi-2

NABi-2 is equipped with an VectorNav VN-100 IMU as well as high resolution (18 bit) absolute position encoders at every joint. The joint encoders are high resolution and can produce position information at several kHz, and are numerically differentiated then low-pass filtered on the actuator to produce joint velocity measurements. The IMU produces (hardware-filtered) accelerometer and gyroscope readings at a maximum of 800Hz, but for our purposes a rate of 400Hz was found to be sufficient (while maintaining low latency when communicating with the main PC). The IMU also runs an onboard filter that estimates the sensor orientation (using an additional magnetometer for yaw), meaning the orientation can be directly read from the IMU and need not be estimated externally.

Before implementing the the full Extended Kalman Filter described above, a simpler kinematics based approach was taken for jumping in place. In this approach, while the feet are in contact with the ground, the joint encoders can be used to describe the configuration of the feet in the body frame, while the IMU can be used to measure orientation. This approach can be viewed as a form of the matching technique described previously. During

flight, the positions and velocities of the body in the world frame can only be obtained through integration of the IMU, which quickly drifts away from the true values due to cumulative errors from integrating the bias in the accelerometer. In actuality, this approach to estimation simply decoupled the available sensor modalities into a simple kinematic filter for estimating positions while on the ground and an attitude estimator while in the air (or on the ground). This approach worked for jumping in place, but is not sufficient for controlled directional locomotion where there is an aerial phase when both feet leave the ground.

To achieve improved state estimation on NABi-2, an open-source implementation of the EKF described previously was modified for the purposes of implementing it on NABi-2. One implementation detail that should be taken note of is the difference in rates of the prediction and update. If the prediction runs at a much faster rate than the update, the update may not provide sufficient correction to keep the estimate from drifting due to noise. On NABi-2, the joint encoders (used in the update step) can produce data at least as fast as the IMU (used in the prediction step), so there were no issues with divergence when testing as long as there are feet placed on the ground.

Note that when there are no longer any feet in contact with the ground, there are no encoder updates being sent to the filter, meaning the estimates of position and velocity will drift. This could be a problem during the flight phases of a jumping motion, but because the filter also performs bias estimation for the IMU the position and velocity usually experience minimal drift during the relatively short flight phases. This filter also requires that the contact state of the foot be provided as an input. This was another area where the proprioceptive nature of NABi-2 was utilized, since the detection of foot contact could

be done using the proprioceptive nature of the limbs, rather than with an external touch sensor. Detection of liftoff (when the foot leaves contact with the ground) is significantly more challenging to implement. This is because liftoff occurs when the upwards acceleration of the center of mass of the (whole) robot is equal and opposite the gravitational acceleration. For NABi-2, the mass properties of the robot are approximated through CAD models, and only a single IMU is mounted on the central body link of the robot. It would be ambitious to try and obtain an accurate measure of center of mass acceleration using only the single IMU, numerically differentiated joint encoders, and approximated mass properties, so an alternative approach was taken.

Liftoff detection on NABi-2 was achieved differently for different gaits. For the jumping locomotion, once the robot extends its legs to a certain length in the thrust phase, the legs simply retract to a nominal length to prepare for touchdown, achieving flight if the legs retract faster than the robot falls (this approach is similar to that of Raibert's original hoppers). For walking locomotion, liftoff was detected when the distance between the body and foot achieved a certain threshold, at which point liftoff is triggered and the robot begins its single support phase. This is described in more detail in Chapter 5.

The use of the estimator came from the need to measure the velocity of the robot's kinematic center (hip) during the stance phase in order to predict a foot placement location per the Raibert-style hopping controller. For this purpose, the estimator works well, and produces relatively smooth velocity estimates that can be fed back into the locomotion controller. A plot of the state estimate while NABi-2 is performing directional jumping can be seen in Figure 4.1 and Figure 4.2.

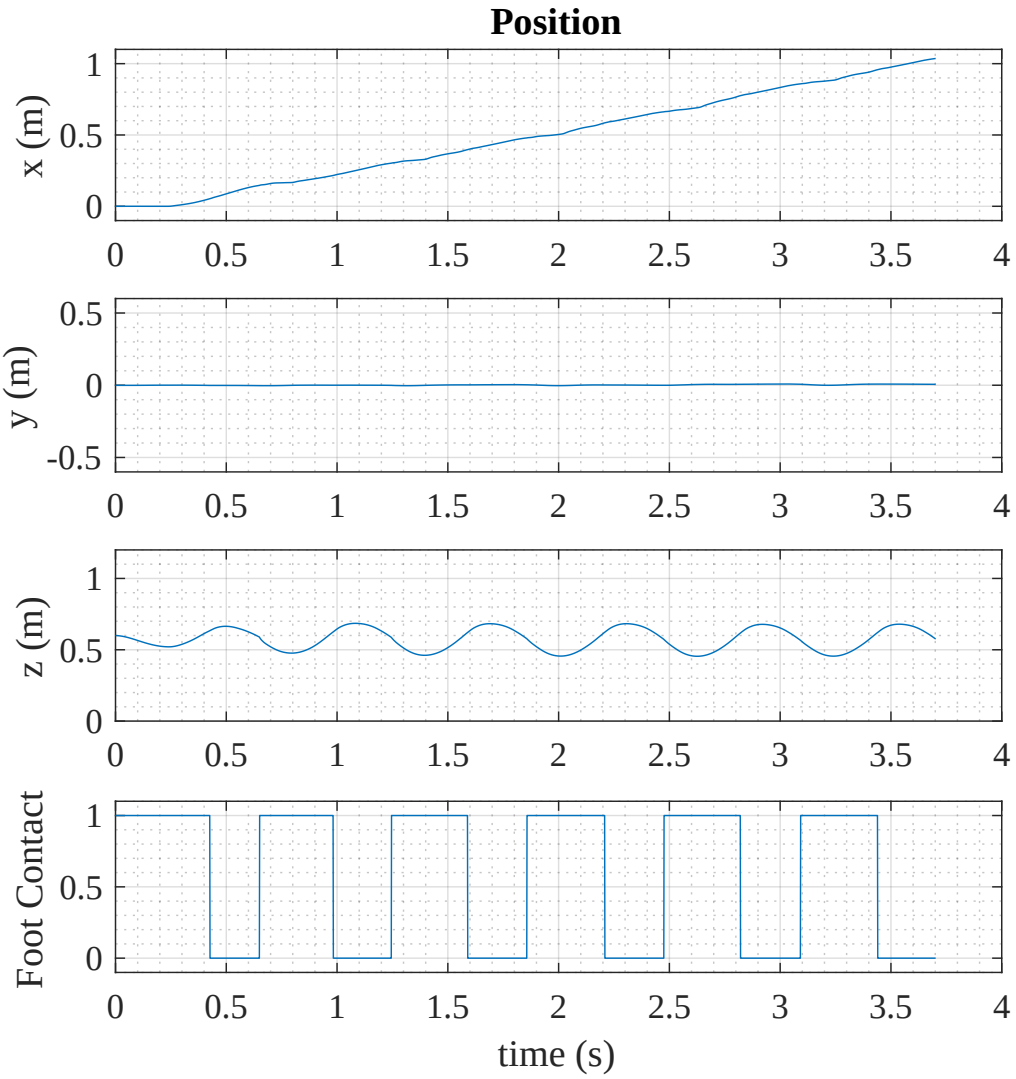


Figure 4.1: Position estimates of NABi-2 while jumping forward. A foot contact is 0 if not in contact with the ground and 1 if in contact. Short periods of no foot contact do not cause the estimation to diverge.

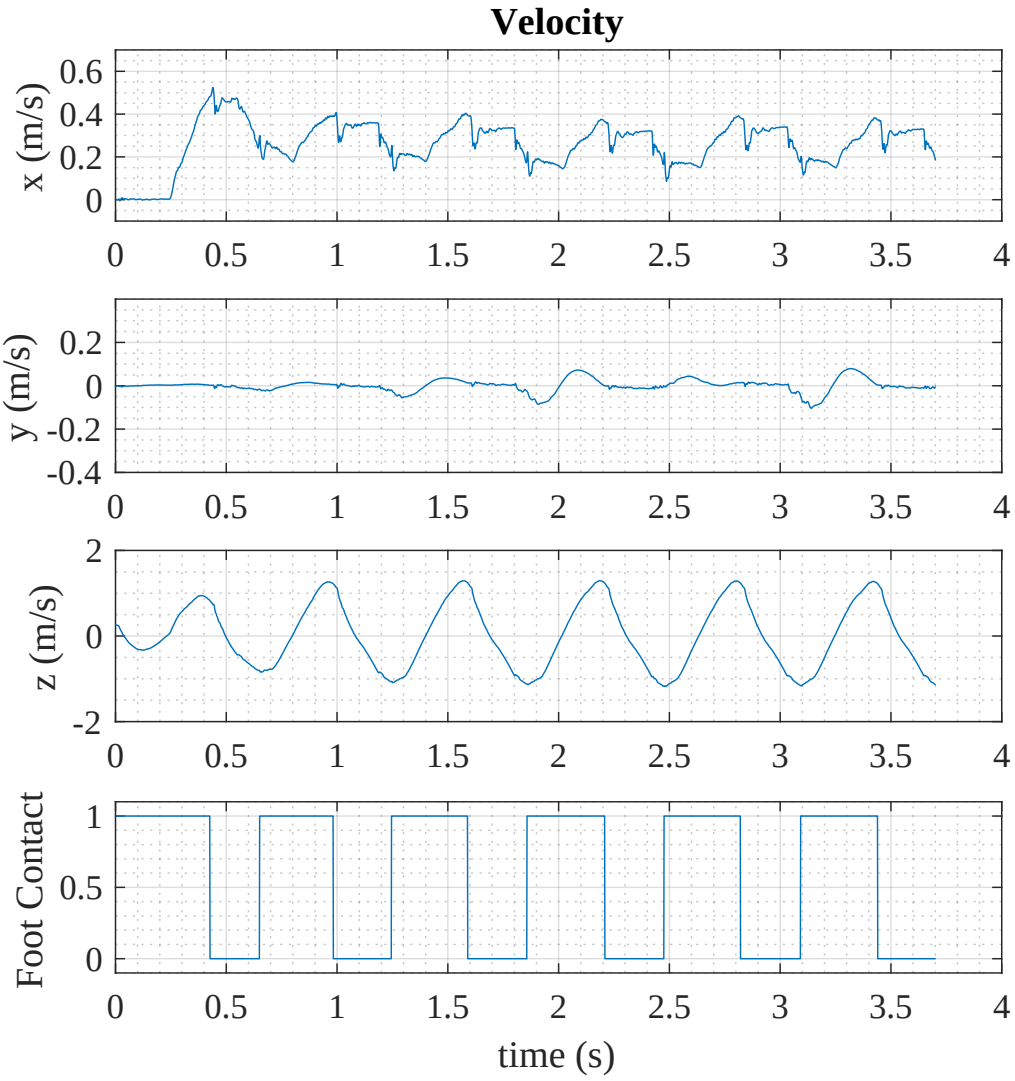


Figure 4.2: Velocity estimates of NABi-2 while jumping forward. The desired velocity to track is 0.4 m/s in the x direction, which is achieved just before liftoff.

4.7 Conclusion

State estimation is a key component for nearly all mobile autonomous robots/systems and is approached in various different ways according to the task at hand and system being implemented. In the case of legged robots, a single IMU rigidly fixed to the robot's kinematic center can be fused with the kinematic data from the leg joint encoders in order to provide a low-drift estimate of the system's position and velocity with respect to an inertial frame. On NABi-2, an open source EKF framework was implemented that predicts the state with the IMU and performs updates with data from the joint encoders. When implementing this EKF based approach, it was necessary to sense the foot contact condition. On NABi-2, it sufficed to detect touchdown with the proprioceptive sensing provided by the BEAR modules. This work implements an EKF based approach because it can run quickly and robustly on our system. It would be interesting future work to try and implement an Unscented Kalman Filter or batch optimization approach for sensor fusion on legged robots, or perhaps an alternative filter which fuses IMU, encoders, and vision data from a camera as well.

CHAPTER 5

Bipedal Locomotion with Proprioceptive Force Control on NABi-2

Abstract

Proprioceptive actuators provide capabilities never before seen in mobile legged robots thanks to their ability to accurately control their torque output and stiffness. The Back-drivable Electric Actuator for Robotics (BEAR) module is an example of such an actuator that is used for the Non-Anthropomorphic Biped Version 2 (NABi-2). However, while NABi-2 has shown impressive in-place jumping capabilities, a closed-loop gait for walking or other directional locomotion has been a challenge due to the fact that the BEAR modules are not amenable to the position-controlled, ZMP-based walking locomotion executed by the original NABi-1 platform. This work explains the process by which directional locomotion control was developed on the NABi-2 platform, and introduces two different bipedal locomotion approaches that have been shown to be effective on the robot: walking using virtual constraints in the operational space, and jumping with a multi-legged Raibert Controller. Additional discussion is then provided on the development process in order to glean some key insights that can be generalized to a broader class of proprioceptive legged robots.

5.1 Introduction

Proprioceptive actuators have recently shown an immense amount of success in mobile legged robots. Quadrupeds like the MIT Cheetah [9] and ALPHRED-2 are capable of incredible feats of agility thanks to the high torque density and force-controllability of these actuators. However, bipeds and humanoids with proprioceptive actuators have yet to achieve the same level of success as their four-legged counterparts. Modern humanoids are still largely implemented with position-controlled servos, series-elastic actuators, or hydraulics. Recent work like [136, 100] shows the potential of using proprioceptive actuators on bipedal systems, but these platforms have yet to achieve the versatility of a classic humanoid, or the mobility of a quadruped. There are several reasons for this, including the inherent stability of quadrupeds, the inherent compliance in proprioceptive actuators, and the lightweight leg design commonly seen alongside proprioceptive actuation. To see an example of all of these features, one need only look to the MIT Cheetah line of robots, one of the most successful series of robots that utilize proprioceptive actuation.

The original MIT Cheetah [113] had a stark resemblance to an actual cheetah, and came equipped with additional features not seen on many modern quadrupeds such as elastic tendons at the feet and a flexible back. This first iteration featured a hierarchical controller that utilized cyclical foot trajectories and an impedance controller for each leg to achieve a stable trot while tethered to a structure which restricts motion to the sagittal plane. The MIT Cheetah 2 was freed from the tether, but notably had removed many of the original design features (such as the foot tendons and flexible back), in favor of a simpler design with a monolithic body and four-bar linkage legs. This second iteration featured a control approach

called impulse planning which utilized the force-control capabilities of its proprioceptive actuation to modulate the impulse being delivered to the body [86]. The current generation of MIT Cheetah (MIT Cheetah 3, MIT Mini-Cheetah) simplifies the design even further, and comprises of a monolithic body structure with four, lightweight, rigid legs that are actuated by three proprioceptive actuators collocated at the hip [9]. The latest iterations utilize an MPC based approach which incorporate a simplified dynamic model of the system that abstracts the legs as force sources that are active while a foot is on the ground [31].

Bipedal robots, in contrast to robots like the MIT Cheetah, are typically designed and controlled in a very specific fashion: as fully actuated humanoids that use traditional dynamics cancellation approaches for control. Fully actuated humanoids have 6 DoF on each leg, compared to the 3 DoF on a quadruped leg, where the three additional DoF are used to control the foot orientation via an ankle. The inclusion of ankle actuators generally increases the mass and moment of inertia of the leg, meaning the lightweight or massless leg assumption commonly utilized when controlling proprioceptive legged robots becomes less applicable. Additionally, a trajectory tracking based approach to locomotion can be somewhat more difficult with proprioceptive actuators, as the inherent compliance in the actuator reduces the (position) tracking accuracy of the system. Position tracking is still utilized in proprioceptive quadrupeds, but quadrupeds themselves are somewhat more robust to falling over, because under normal operation they have a lower center of mass, larger support polygon, and more legs to assist in recovery from a disturbance. These combined effects make it quite difficult to implement a walking gait on a proprioceptive-ly actuated humanoid where all actuated DoF are replaced with proprioceptive actuators. Indeed, developing bipedal lo-

comotion using proprioceptive actuators would require a shift both in the control approach and the design of the robot itself.

To address the design problem, the Non-Anthropomorphic Biped Version 2 (NABi-2) was developed. NABi-2 is a bipedal robot whose leg configuration resembles that of a human walking sideways, and has gone through several design iterations to arrive at its current form [136]. The original NABi-1 began development as a simple, lightweight, planar biped with only 2 actuated DoF on each leg (at the hip and knee) and a compliant foot element [135]. The compliant elements could be configured in such a way that they would only be active during certain parts of a locomotion gait, such as to assist with maintaining balance during the single support phase of a walking cycle. Importantly, NABi-1 utilized off-the-shelf position controlled actuators, and thus the compliant foot elements provided some impact resistance. When moving to NABi-2, the position-controlled actuators were replaced with proprioceptive ones, and the compliant foot elements were removed because the actuators already provided compliance and impact resistance. Furthermore, NABi-2 adopted the leg design of the modern proprioceptive leg, with three actuated degrees of freedom collocated at the hip and no ankle actuation. As a result, NABi-2 represents a system that has design qualities of a proprioceptive quadruped, but with only two legs.

NABi-2 represents a paradigm shift in how bipedal systems in the future can be designed. However, the traditional control approaches for bipeds are not very applicable to these new types of actuators and robots, meaning a shift in how these systems are controlled is also necessary. This work contributes two approaches to bipedal locomotion that were developed for the NABi-2 platform: a biped with a unique leg configuration and proprioceptive

actuation. The first approach is a walking gait that takes makes use of virtual constraints in the operational space, while the second is a pronking gait that is inspired by the classic Raibert monopeds. These two approaches to locomotion are likely not directly generalizable to bipeds outside of the NABi form factor, but can still provide valuable insight on how bipedal systems with proprioceptive actuation can be controlled in the future.

The remainder of this paper is organized as follows: Section 5.2 reviews some classical approaches to bipedal locomotion and how relevant they were to NABi-2. Section 5.3 details the walking locomotion approach developed for NABi-2. Section 5.4 then explains the directional pronking gait. Section 5.5 discusses the effectiveness of these approaches, as well as how they can be used to inform locomotion on systems with similar design. Section 5.6 concludes with some final thoughts and directions for future work.

5.2 Traditional Bipedal Locomotion

There is an infinite design space in which bipedal robots can reside, but for the purposes of locomotion control and planning there are certain design choices that will favor certain locomotion approaches. One such choice is how anthropomorphic the biped is: a humanoid [55] is fully anthropomorphic, whereas the bird-inspired Cassie [103] is less so. Another aspect is the approach to leg actuation and level of compliance: a leg that is designed like a robot arm in factory automation with position-controlled actuators represents one extreme, while a mostly (or completely) passive dynamic system with compliant elements represents the opposite extreme.

The classical humanoid robot represents the limit of both these axes, being fully anthropomorphic as well as fully actuated with position-controlled servos. For robots such as these, an effective approach to locomotion is to first plan the foot and hip (kinematic center) trajectory (ie. through preview control [65] or TVLQR [123]), and then utilize a dynamics cancelling technique (ie. computed torque control [88], task space optimization [130]) to track the trajectory. The opposite end of the spectrum is something like Agility Robotics' Cassie, which features passive-compliant leg springs and a digitigrade leg design that is inspired by flightless birds [8]. These types of platforms rely on other forms of control, such as Hybrid-Zero-Dynamics (HZD) [102], hierarchical/compositional controllers [97], or scheduled gaits [58]. NABi-2 is also highly non-anthropomorphic, but lies in the middle ground of actuation, as proprioceptive legs are inherently compliant (due to the actuators) while still fully actuated (for point feet).

As a first attempt at locomotion, the ZMP-based approach for NABi-1 was ported to NABi-2. However, this approach was unable to take more than a couple of steps, as the compliance on NABi-2 made it difficult to precisely track position trajectories, and the lack of the compliant foot element made it relatively easy for the robot to tip over when taking a step. And, while it is true that the compliance of the proprioceptive actuators on NABi-2 is tunable, achieving the same stiffness as a traditional actuator with high gear ratios is difficult. This is because the low gear ratio in a proprioceptive actuator creates less inherent damping from friction, so the stiffness of the actuator must be actively controlled through a actuator-level control loop. However, sensor noise in this low-level loop limits how aggressively the controller gains can be set – gains that are too high could cause a large increase in actuator

heat output, or instability in the worst case. The end result of this stiffness limit can be seen with the attempt at position-controlled walking on NABi-2. The resulting locomotion gait worked for a couple of steps, but then failed due to undesired oscillations in the legs.

5.3 Walking with Proprioception

One beneficial feature of proprioceptive actuators is their innate ability to sense certain aspects about their state. As such, it makes sense to use a proprioceptive actuator in an event-based fashion, where certain events detected through the actuator sensors trigger certain actions. This is in contrast to traditional trajectory tracking, where actions are parameterized with respect to time. The event-based approach generally leads to compositional/hierarchical controllers that are simple state machines that cycle through a set of motions or lower-level controllers as dictated by the detection of certain events. This approach was originally used to control the classical Raibert hoppers [97].

To control directional locomotion on NABi-2, several attempts were made at implementing a compositional approach that utilized an event-based state machine to dictate the different phases of the walking cycle. There were two general frameworks tested: the first approach, called *dual-impedance*, treated each leg as a spring-loaded inverted pendulum (SLIP, or spring-mass) and walked by repeatedly applying impulses with each leg. The second approach, called *virtually constrained operational space (VCOS)*, walked by applying virtual constraints on particular operational/task space coordinates. The latter approach had the most success, but it is useful to examine the other approach to see why it was not as effective, and how it may be useful elsewhere. It is worth noting that locomotion for NABi-2

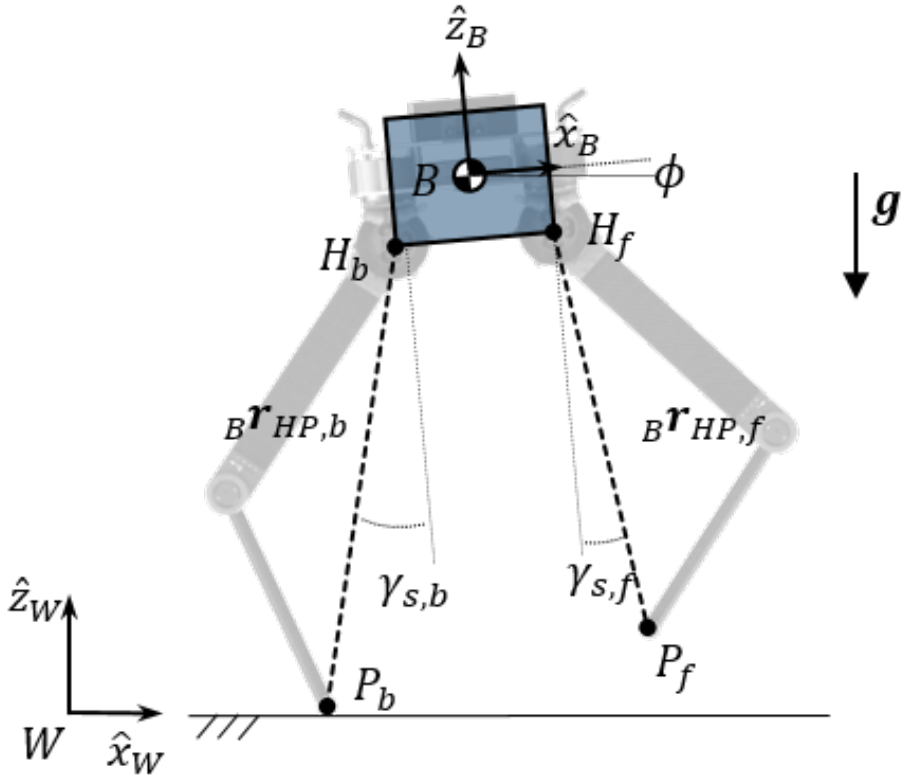


Figure 5.1: Diagram of the model used in the dual-impedance approach to walking. The legs are assumed to be massless springs whose rest length and angle with respect to the body can be controlled.

is currently restricted to the sagittal plane.

5.3.1 Dual-Impedance

In the dual-impedance approach, each leg was treated as a spring-damper system, where a virtual leg was created between the hip joint and the foot of each leg, similar to [136]. For this approach, each leg embeds the spring-mass with damping dynamics during stance as:

$$m({}_B\boldsymbol{a}_{HP_i}) = \boldsymbol{K}_{p,s}({}_B\boldsymbol{r}_{HP_i} - \boldsymbol{l}_0) + \boldsymbol{K}_{d,s}({}_B\boldsymbol{v}_{HP_i} - \boldsymbol{0}) \tag{5.1}$$

Where ${}_B\boldsymbol{r}_{HP_i}$, ${}_B\boldsymbol{v}_{HP_i}$, and ${}_B\boldsymbol{a}_{HP_i}$ are the position, velocity, and acceleration of the i^{th} foot with respect to the i^{th} hip joint expressed in the body frame, $\boldsymbol{K}_{p,s}$ and $\boldsymbol{K}_{d,s}$ are the spring constant and damping constant, and $\boldsymbol{l}_0$ is the nominal length vector of the virtual spring. The nominal length vector is parameterized with two values (due to the planar restriction): the leg length l and the leg angle γ_s of the leg with respect to the body frame. The nominal leg length vector can be described with a nominal leg length scalar value, l_0:

$$\boldsymbol{l}_0 = \begin{bmatrix} l_0 \cos\gamma_s \\ \\ l_0 \sin\gamma_s \end{bmatrix} \tag{5.2}$$

The leg angle γ_s describes the angle that the virtual leg which extends from the hip to the foot makes with respect to the vertical axis of the body frame. In practice, this can be found using forward kinematics and the `atan2` function. For implementation on NABi-2, the spring-damper constants as well as the nominal leg length are tuned empirically, and mass m is the total mass of the robot (including the mass of the legs). Now, assuming massless legs, these dynamics can be embedded with the contact foot Jacobian as:

$$\boldsymbol{\tau}_i = \boldsymbol{J}_{c,i}^T(m_B\boldsymbol{a}_{HP_i}) \tag{5.3}$$

Where $\boldsymbol{\tau}_i$ is a vector of joint torques, and $\boldsymbol{J}_{c,i}$ is the contact Jacobian relating joint rates to end effector (foot) velocities, all expressed for the i^{th} leg.

With each leg now acting as a virtual spring, a walking gait was implemented by repeatedly retracting and extending the virtual leg spring. This can be achieved by modulating the nominal leg length, l_0. At the start of the gait, each leg is commanded to the nominal leg length l_0^-, and motion begins with one leg imparting an impulse into the ground by increasing the nominal leg length to $l_0^+ = \Delta l + l_0^-$. Once the leg reaches extended length l_0^+,

it retracts to its nominal length l_0^- (lifting it off the ground) and modifies its leg angle using a Raibert style foot placement law:

$$\gamma_{TD} = \gamma^*({}_W v_{WB,x}) + K_{p,\gamma}({}_W v_{WB,x} - v_{d,x}) \tag{5.4}$$

Where γ_{TD} refers to the leg angle at touchdown, $\gamma^*({}_W v_{WB,x})$ is the nominal leg angle, and $K_{p,\gamma}$ gains the error between the current body velocity ${}_W v_{WB,x}$ and the desired body velocity $v_{d,x}$.

Additionally, the leg which is planted on the ground attempts to keep the body from pitching by applying a corrective torque around the body pitch with a PD control law:

$$\tau_\phi = -K_{p,\phi}(\phi - \phi_d) - K_{d,\phi}(\dot{\phi} - \dot{\phi}_d) \tag{5.5}$$

Where ϕ and $\dot{\phi}$ are the pitch and pitch rate measured by the IMU, and τ_ϕ is a torque applied at the hip joint of the stance leg to track a desired pitch and pitch rate, $(\phi_d, \dot{\phi}_d)$.

Once the airborne (swing) leg is detected to have touched back down, it maintains its nominal length and the extension-retraction actions are performed on the opposite leg. The resulting event-based state machine is shown in Figure 5.2. Unfortunately, this approach was never able to achieve successful walking. The observed behavior was that the swing foot would be constantly dragging on the ground, often to the point of making the robot tip over. This occurred mainly for two reasons: 1) the moment transferred to the body from the swing leg moving would cause the body to abruptly dip down towards the ground, preventing the swing leg from lifting off, and 2) the knee actuator of the supporting leg would saturate, causing the stance leg to sag down, further exacerbating the problem. Tuning was done on the parameters of the virtual leg springs and the size and height of the steps being taken,

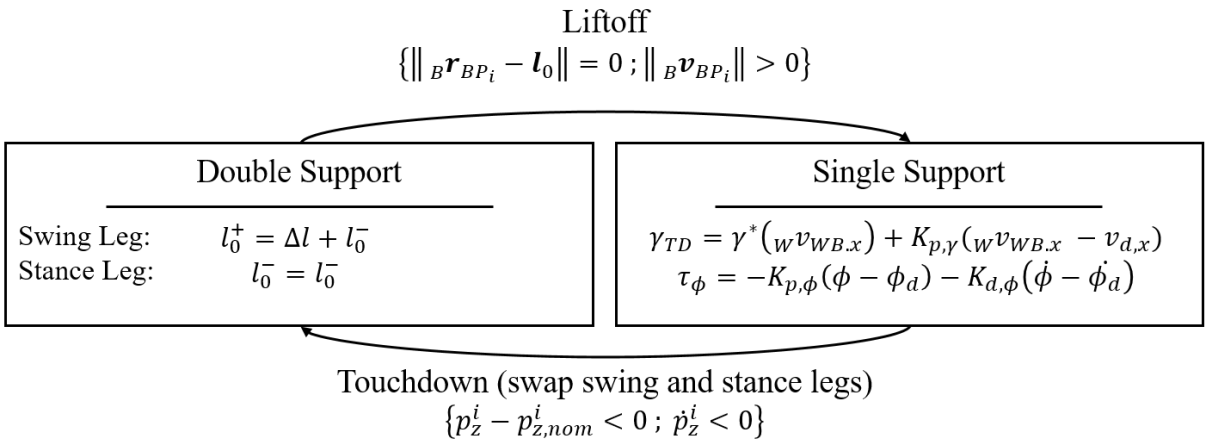


Figure 5.2: State machine describing the dual-impedance walking approach.

but the foot dragging issue still remained.

This result suggests that even though the legs of NABi-2 are designed with relatively low inertia, the comparatively similar inertia of the main body meant that this traditional Raibert style controller is less effective. Increasing the inertia of the body by adding a torso, for example, could improve the viability of this approach. Additionally, this approach may be better suited for a bounding style of gait with a completely aerial phase, as this could alleviate the oscillations from the stance leg dynamics. In fact, this approach with an aerial phase would resemble the impulse planning approach utilized for the MIT Cheetah 2. Importantly, this approach demonstrates that for asymmetric motions on NABi-2, the simple, massless leg models are not very accurate, and the dynamics of the legs need to be incorporated in the control.

5.3.2 Virtually Constrained Operational Space

In the virtually constrained operational space (VCOS) approach, the idea is to virtually constrain the body of the robot to mimic a reduced model whose behavior is more intuitive to control. This approach, like the dual-impedance one, is also a compositional approach driven by events and with the single support and double support phases. However, the VCOS approach utilizes more complex swing leg trajectories (rather than just a simple foot placement control law) that perform the full inversion of the rigid body dynamics in the operational space. The stance leg is also controlled to maintain the height and pitch of the body, making the body dynamics in single support resemble the linear inverted pendulum popularly utilized in bipedal locomotion.

The resulting gait is fairly conservative for reasons which will be discussed in the following, but this approach does represent an interesting intersection of the traditional bipedal locomotion approaches and the compositional approaches popularized by Raibert. This intersection occurs because NABi-2 is a bipedal system, but utilizes proprioceptive actuation with a unique leg configuration that introduces a level of compliance and force control into the system. To achieve walking, force control is utilized in the single support phase, where the torques being commanded to the stance leg are reacting to the change in position and pitch of the body, as opposed to simply correcting an error from a pre-planned trajectory. At the same time, the swing leg in single support *is* following a trajectory (in operational space), but can be superseded by the detection of the touchdown of the foot, transitioning the state back to double support. In double support, there is a whole-body controller commanding the joint torques to shift the body forward and backward.

5.3.2.1 Double Support

In double support, a whole body controller is controlling the body to maintain a certain height and orientation, while also commanding the feet to a certain position. This controller is a simplified version of the double support controller in [136] that assumes minimal motion (and no motion outside the sagittal plane) and walking over flat ground.

$$\boldsymbol{\tau}_i = \boldsymbol{J}_{c,i}^T \boldsymbol{f}_i \tag{5.6}$$

$$\boldsymbol{f}_i = \boldsymbol{f}_{i,e} + \boldsymbol{f}_{i,g} + \boldsymbol{f}_{i,p} \tag{5.7}$$

Where the commanded torques for the i^{th} leg, $\boldsymbol{\tau}_i \in \mathbb{R}^{2\times 1}$, are again found by multiplying the desired forces $\boldsymbol{f}_i \in \mathbb{R}^{2\times 1}$ with the contact Jacobian $\boldsymbol{J}_{c,i} \in \mathbb{R}^{2\times 2}$. In this case, the desired forces are comprised of three components: an end-effector position feedback $\boldsymbol{f}_{i,e}$, a gravity compensation term $\boldsymbol{f}_g$, and a pitch regulation term $\boldsymbol{f}_{i,p}$. These are described by:

$$\boldsymbol{f}_{i,e} = \boldsymbol{K}_{p,e}({}_B\boldsymbol{r}_{BP_i,d} - {}_B\boldsymbol{r}_{BP_i}) + \boldsymbol{K}_{d,e}({}_B\boldsymbol{v}_{BP_i,d} - {}_B\boldsymbol{v}_{BP_i}) \tag{5.8}$$

$$\boldsymbol{f}_{i,g} = K_g m \boldsymbol{g} \tag{5.9}$$

$$\boldsymbol{f}_{i,p} = \pm \begin{bmatrix} 0 & K_{p,p}(\phi_d - \phi) + K_{d,p}(\dot{\phi}_d - \dot{\phi}) \end{bmatrix} \tag{5.10}$$

Where $\boldsymbol{K}_{p,e}$ and $\boldsymbol{K}_{d,e}$ are the diagonal gain matrices around a setpoint foot position ${}_B\boldsymbol{r}_{BP_i,d}$ and velocity ${}_B\boldsymbol{v}_{BP_i,d}$, K_g is a scale factor depending on the number of legs, and $K_{p,p}$ and $K_{d,p}$ are gains for the pitch control. Note that the sign of $\boldsymbol{f}_{i,p}$ will depend on the sign of the pitch measurement, and should be equal and opposite for each leg.

In double support, the setpoint foot velocity is set to zero (${}_B\boldsymbol{v}_{BP_i,d} = 0$) and the setpoint foot position is shifted from a nominal position to a new position by adding a constant

horizontal offset Δr_x to each foot:

$$ {}_B\boldsymbol{r}_{BP_i,d} = {}_B\boldsymbol{r}_{BP_i,0} + \begin{bmatrix} \Delta r_x \\ \\ 0 \end{bmatrix} \tag{5.11} $$

Shifting both feet setpoint in the same direction an equal amount has the effect of moving the body in the opposite direction. This is the mechanism through which the forward or backward motions are introduced in the double support phase. However, it is important to note that the feet are never allowed to stabilize around the new setpoint. Instead, after the distance between the body and the upcoming stance foot has passed a certain value ($\Delta r_{x,LO}$ which is less than Δr_x), the state immediately transitions to single support. This has the effect of always entering the single support phase with the body moving at non-zero velocity, which is important for ensuring a reasonable single support phase. The conditions for the liftoff event that transitions NABi-2 from double support to single support can be written as:

$$ \{ {}_B\boldsymbol{r}_{BP_i} - \Delta r_{x,LO} = 0 | {}_B\boldsymbol{v}_{BP_i,d} < 0 \} \tag{5.12} $$

Put simply, this condition is reached when the body reaches a certain distance away from the stance foot while moving towards the stance foot.

As a matter of bookkeeping, note that the asymmetric nature of the 'sideways' walking gait on NABi-2 means that in one walking cycle, the first step *widens* the stance (the distance between feet is increased), and the following step *narrows* the stance (the distance between feet is decreased), as shown in Figure 5.3. Care should be taken to ensure all offsets for each foot are accounted for during each period of the walking cycle, and that signs are correctly assigned to the various event conditions. On NABi-2, this manifests as having one double

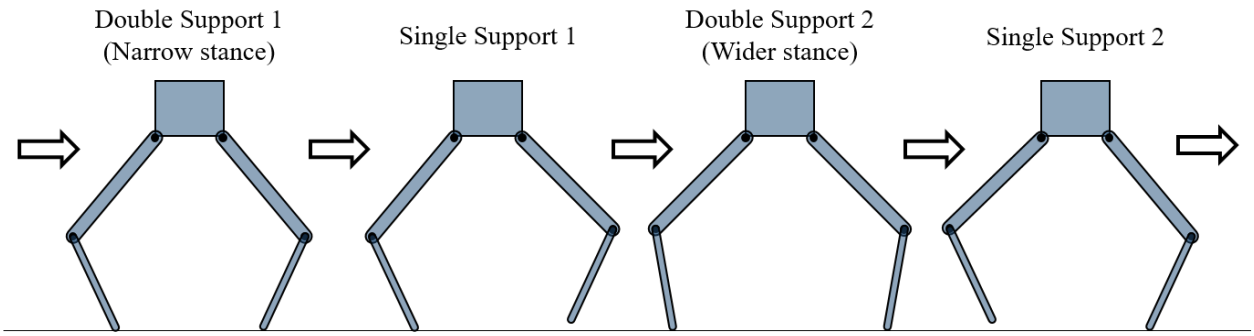


Figure 5.3: The different phases in the asymmetric VCOS walking gait.

support phase with a nominal stance width, followed by a double support phase (immediately following the first single support phase) with stance width equal to the nominal width plus the length of the first step.

5.3.2.2 Single Support

The single support phase of the VCOS approach attempts to regulate a few separate aspects of the robot by using virtual constraints in the operational space. The resulting motion that occurs is a step in the direction of travel, though the physical and temporal length of the step is not fixed, as it may be with a traditional bipedal approach. This occurs mainly for two reasons: Firstly, the swing leg trajectory being followed can be interrupted by the detection of the swing foot touching down on the ground; Secondly, the body is not attempting to maintain a horizontal position, meaning the swing foot can be in an arbitrary horizontal position when it touches down.

The NABi-2 platform undergoing planar motion has four actuated degrees of freedom available to it (two for each leg), and one un-actuated DoF where the stance foot contacts the ground. In the single support phase, the actuated DoF are used to affect different aspects

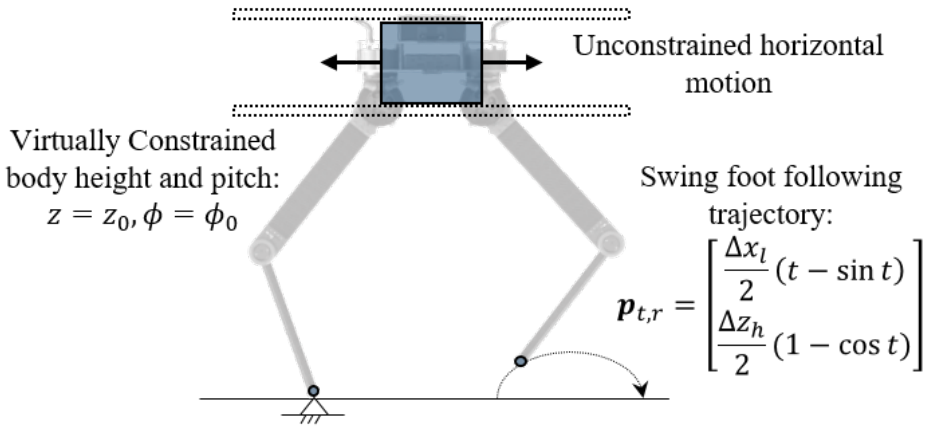


Figure 5.4: Constrained and unconstrained states in the single support phase of the VCOS approach. Note that the time variable t needs to be properly scaled for the desired single support time.

of the stepping motion. The two DoF on the swing leg are used to execute a cycloidal trajectory in the planar coordinate system that lifts the foot up and sets it back down, with user-defined values for lift height and step size. The DoF on the stance leg are used to regulate two values: the height of the body, and the pitch of the body. These constraints are illustrated with Figure 5.4.

Ideally, combining the contributions of each leg should mean the body always maintains a constant height and is parallel to the ground, meaning the swing leg has full authority to execute its step trajectory. Notice that the horizontal component of the body is not controlled, because there are not enough actuated DoF on the stance leg to regulate the horizontal and vertical positions as well as the orientation (pitch) of the body (the addition of a third ankle actuator on each leg would allow for this). This means that the size of the step being taken is dependent on the horizontal velocity of the body when entering the single

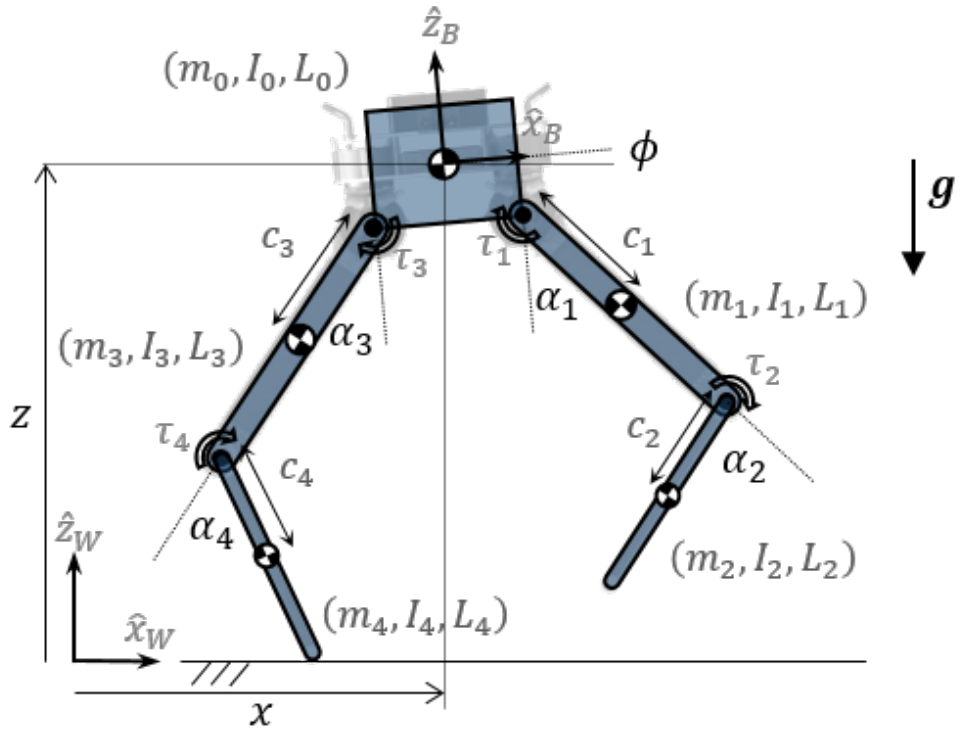


Figure 5.5: Diagram of the full rigid body model used for the single support phase of the VCOS approach to walking.

support phase.

Initially, the inertia and velocity product terms of the robot dynamics were ignored and the VCOS approach was implemented using only the Jacobian and desired task space forces (like what is used in the double support phase). However, this approach yielded similar results as the dual-impedance approach, where the body would dip down during a step, resulting in no step being taken and eventually tipping of the robot. As such, the single support is instead executed with a task/operational space dynamics inversion approach that makes use of the full equations of motion. The dynamics of the single support begin with the complete, floating body dynamics of the robot with state $\boldsymbol{x} \in \mathbb{R}^{14}$ defined as the

concatenation of a set of generalized coordinates (positions) $\boldsymbol{\theta} \in \mathbb{R}^7$ and the corresponding generalized velocities $\boldsymbol{u} \in \mathbb{R}^7$:

$$\boldsymbol{x} = (\boldsymbol{\theta}, \boldsymbol{u}) \tag{5.13}$$

$$= ({}_W\boldsymbol{r}_{WB}, \phi, \boldsymbol{\alpha}, {}_W\boldsymbol{v}_{WB}, \dot{\phi}, \dot{\boldsymbol{\alpha}}) \tag{5.14}$$

$$= (x, z, \phi, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \dot{x}, \dot{z}, \dot{\phi}, \dot{\alpha}_1, \dot{\alpha}_2, \dot{\alpha}_3, \dot{\alpha}_4) \tag{5.15}$$

Where x and z are the horizontal and vertical states of the body in the inertial frame and ϕ is the pitch of the body in the world frame. The α_i for $i = 1, 2, 3, 4$ are the relative joint angles of the (planar) system, representing the front leg hip pitch, front leg knee pitch, back leg hip pitch, and back leg knee pitch respectively (yaw joints are fixed at zero to maintain planar constraint). Note that relative angles are used for ease of implementation on the robot. The rigid body model of NABi-2 is shown in Figure 5.5.

The equations of motion can be derived with the method of Lagrange or Newton-Euler and can be written as:

$$\boldsymbol{M}(\boldsymbol{\theta})\dot{\boldsymbol{u}} + \boldsymbol{b}(\boldsymbol{\theta}, \boldsymbol{u}) + \boldsymbol{g}(\boldsymbol{\theta}) + \boldsymbol{J}^T(\boldsymbol{\theta})\boldsymbol{F} = \boldsymbol{S}_a^T\boldsymbol{\tau} \tag{5.16}$$

Where $\dot{\boldsymbol{u}} \in \mathbb{R}^7$ represents the generalized accelerations, $\boldsymbol{F} \in \mathbb{R}^m$ are the m number of (external) contact forces acting on the system (which will later be used to constrain the system), and $\boldsymbol{\tau} \in \mathbb{R}^4$ are the input joint torques (one for each α_i. The standard terms from the classical equations of motion for manipulators are the inertia matrix $\boldsymbol{M}(\boldsymbol{\theta}) \in \mathbb{R}^{7\times7}$, the velocity products (coriolis and centrigufal terms) $\boldsymbol{b}(\boldsymbol{\theta}, \boldsymbol{u}) \in \mathbb{R}^7$, and the gravity term $\boldsymbol{g}(\boldsymbol{\theta}) \in \mathbb{R}^7$. The remaining terms are the contact Jacobian $\boldsymbol{J}(\boldsymbol{\theta}) \in \mathbb{R}^{m\times7}$ which maps generalized velocities to contact point velocities, and the actuator selection matrix $\boldsymbol{S}_a \in \mathbb{R}^{4\times7}$

which distinguishes between actuated and non-actuated DoF.

The objective of the inverse dynamics function is to take in as input a desired acceleration (and current state) and output the torques that will produce that acceleration on the system. However, the generalized accelerations are not necessarily the ones that need to be controlled. In the case of single support for the VCOS walking method, the body vertical height and pitch are elements of the generalized coordinates, but the end-effector (feet) positions of the swing and stance legs are not. As such, it is necessary to make use of the *task space* or *operational space* of the desired task/objective alongside the *support space* of the desired constraints to produce a set of transformed dynamics that will accurately output the torques required to execute a certain motion.

The task space coordinates $\boldsymbol{\theta}_t \in \mathbb{R}^4$ that are being controlled in single support are the vertical body position z, body pitch ϕ, and the swing foot position ${}_B\boldsymbol{r}_{BP_t}$, which can be expressed as a function of the original generalized coordinates (using kinematics):

$$\boldsymbol{\theta}_t = \boldsymbol{\xi}_{task}(\boldsymbol{\theta}) = (z, \phi, {}_B\boldsymbol{r}_{BP_t}) \tag{5.17}$$

$$= (z, \phi, p_{t,x}, p_{t,z}) \tag{5.18}$$

The support space coordinates $\boldsymbol{\theta}_s \in \mathbb{R}^2$ constrain the position of the stance foot ${}_W\boldsymbol{r}_{WP_s}$ to a nominal fixed position ${}_W\boldsymbol{r}_{WF}$ on the ground during single support, again expressed as a function of the generalized coordinates:

$$\boldsymbol{\theta}_s = \boldsymbol{\xi}_{support}(\boldsymbol{\theta}) = {}_W\boldsymbol{r}_{WP_s} - {}_W\boldsymbol{r}_{WF} \tag{5.19}$$

Then, these coordinates are differentiated with respect to the the generalized coordinates $\boldsymbol{\theta}$

to produce the *task space Jacobian* $\boldsymbol{J}_t$ and the *support space Jacobian* $\boldsymbol{J}_s$:

$$\boldsymbol{J}_t(\boldsymbol{\theta}) := \frac{\partial \boldsymbol{\xi}_{task}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{4\times 7} \tag{5.20}$$

$$\boldsymbol{J}_s(\boldsymbol{\theta}) := \frac{\partial \boldsymbol{\xi}_{support}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{2\times 7} \tag{5.21}$$

Which relate the task and support space velocities, $\boldsymbol{u}_t$ and $\boldsymbol{u}_s$, respectively, to the generalized velocities of the system $\boldsymbol{u}$:

$$\boldsymbol{u}_t = \boldsymbol{J}_t(\boldsymbol{\theta})\boldsymbol{u} \tag{5.22}$$

$$\boldsymbol{u}_s = \boldsymbol{J}_s(\boldsymbol{\theta})\boldsymbol{u} \tag{5.23}$$

Now, these Jacobians are used to transform the dynamics into the task space while under the constraints of the support space. The equations of motion for the support space are obtained by left multiplying (5.16) by $\boldsymbol{J}_s\boldsymbol{M}^{-1}$ and utilizing the relationship $\boldsymbol{J}_s\dot{\boldsymbol{u}} = \dot{\boldsymbol{u}}_s - \dot{\boldsymbol{J}}_s\boldsymbol{u}$, which comes from the time derivative of (5.23):

$$\dot{\boldsymbol{u}}_s - \dot{\boldsymbol{J}}_s\boldsymbol{u} + \boldsymbol{J}_s\boldsymbol{M}^{-1}(\boldsymbol{b}+\boldsymbol{g}) = \boldsymbol{\Lambda}_s^{-1}\boldsymbol{f}_s + \boldsymbol{J}_s\boldsymbol{M}^{-1}\boldsymbol{S}_a^T\boldsymbol{\tau} \tag{5.24}$$

where $\dot{\boldsymbol{u}}_s$ are the support space accelerations and $\boldsymbol{\Lambda}_s = (\boldsymbol{J}_s\boldsymbol{M}^{-1}\boldsymbol{J}_s^T)^{-1}$ is the support space inertia matrix. Left multiplication of (5.24) by $\boldsymbol{\Lambda}_s$ recovers the familiar form:

$$\boldsymbol{\Lambda}_s\dot{\boldsymbol{u}}_s + \boldsymbol{\mu}_s + \boldsymbol{\rho}_s = \boldsymbol{f}_s + \bar{\boldsymbol{J}}_s^T\boldsymbol{S}_a^T\boldsymbol{\tau} \tag{5.25}$$

where $\boldsymbol{\mu}_s$, $\boldsymbol{\rho}_s$, and $\bar{\boldsymbol{J}}_s$ follow from the left multiplication:

$$\boldsymbol{\Lambda}_s = (\boldsymbol{J}_s\boldsymbol{M}^{-1}\boldsymbol{J}_s^T)^{-1} \tag{5.26}$$

$$\bar{\boldsymbol{J}}_s^T = \boldsymbol{\Lambda}_s\boldsymbol{J}_s\boldsymbol{M}^{-1} \tag{5.27}$$

$$\boldsymbol{\mu}_s = \bar{\boldsymbol{J}}_s^T\boldsymbol{b} - \boldsymbol{\Lambda}_s\dot{\boldsymbol{J}}_s\boldsymbol{u} \tag{5.28}$$

$$\boldsymbol{\rho}_s = \bar{\boldsymbol{J}}_s^T\boldsymbol{g} \tag{5.29}$$

Assuming that in stance the position of the foot is stationary, $\boldsymbol{u}_s = \dot{\boldsymbol{u}}_s = 0$, and $\boldsymbol{f}_s$, the forces at the stance foot necessary to enforce the constraint, can be solved for in (5.25):

$$\boldsymbol{f}_s = \bar{\boldsymbol{J}}_s^T \boldsymbol{S}_a^T \boldsymbol{\tau} - \boldsymbol{\mu}_s - \boldsymbol{\rho}_s \tag{5.30}$$

Now, $\boldsymbol{f}_s$ can be input into (5.16) to generate the constrained equations of motion:

$$\boldsymbol{M}\dot{\boldsymbol{u}} + \boldsymbol{N}_s^T(\boldsymbol{b} + \boldsymbol{g}) + \boldsymbol{\gamma}(\boldsymbol{\theta}, \boldsymbol{u}) = \boldsymbol{N}_s^T \boldsymbol{S}_a^T \boldsymbol{\tau} \tag{5.31}$$

where

$$\boldsymbol{\gamma}(\boldsymbol{\theta}, \boldsymbol{u}) = \boldsymbol{J}_s^T \boldsymbol{\Lambda}_s \dot{\boldsymbol{J}}_s \boldsymbol{u} \tag{5.32}$$

is a velocity dependent term that is generated through the constraint and

$$\boldsymbol{N}_s^T = \mathbb{I} - \boldsymbol{J}_s^T \boldsymbol{\Lambda}_s \boldsymbol{J}_s \boldsymbol{M}^{-1} \tag{5.33}$$

is the dynamically consistent *nullspace projector* for support.

Next, the task space dynamics can now be obtained with respect to the support-constrained dynamics in a similar fashion as the support space dynamics by left multiplying (5.31) by $\boldsymbol{J}_t \boldsymbol{M}^{-1}$, substituting $\boldsymbol{J}_t \dot{\boldsymbol{u}} = \dot{\boldsymbol{u}}_t - \dot{\boldsymbol{J}}_t \boldsymbol{u}$, and then left multiplying again by $\boldsymbol{\Lambda}_t = (\boldsymbol{J}_t \boldsymbol{M}^{-1} \boldsymbol{N}_s^T \boldsymbol{J}_t^T)^{-1}$, resulting in:

$$\boldsymbol{\Lambda}_t \dot{\boldsymbol{u}}_t + \boldsymbol{\mu}_t + \boldsymbol{\rho}_t = \boldsymbol{f}_t \tag{5.34}$$

where $\boldsymbol{\mu}_t$, $\boldsymbol{\rho}_t$, and $\boldsymbol{f}_t$ again follow from the left multiplication:

$$\boldsymbol{\Lambda}_t = (\boldsymbol{J}_t \boldsymbol{M}^{-1} \boldsymbol{N}_s \boldsymbol{J}_t^T)^{-1} \tag{5.35}$$

$$\bar{\boldsymbol{J}}_t^T = \boldsymbol{\Lambda}_t \boldsymbol{J}_t \boldsymbol{M}^{-1} \boldsymbol{N}_s \tag{5.36}$$

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{J}}_t^T \boldsymbol{b} - \boldsymbol{\Lambda}_t \dot{\boldsymbol{J}}_t \boldsymbol{u} + \boldsymbol{\Lambda}_t \boldsymbol{J}_t \boldsymbol{M}^{-1} \boldsymbol{\gamma}(\boldsymbol{\theta}, \boldsymbol{u}) \tag{5.37}$$

$$\boldsymbol{\rho}_t = \bar{\boldsymbol{J}}_t^T \boldsymbol{g} \tag{5.38}$$

$$\boldsymbol{f}_t = \bar{\boldsymbol{J}}_t^T \boldsymbol{S}_a^T \boldsymbol{\tau} \tag{5.39}$$

The desired task space accelerations $\dot{\boldsymbol{u}}_{t,d}$ can now be input into the task space dynamics, and the control torques necessary to achieve these accelerations can be found from the task space forces $\boldsymbol{f}_t$:

$$\boldsymbol{f}_t = \boldsymbol{\Lambda}_t \dot{\boldsymbol{u}}_{t,d} + \boldsymbol{\mu}_t + \boldsymbol{\rho}_t \tag{5.40}$$

$$\boldsymbol{\tau} = (\bar{\boldsymbol{J}}_t^T \boldsymbol{S}_a^T)^{-1} \boldsymbol{f}_t \tag{5.41}$$

These torques are now enforcing the desired task space dynamics onto a subset of the original system's configuration space that is consistent with the support constraints.

The desired task space accelerations are generated from PD controllers around the desired setpoints. The vertical body position and pitch are set to be constant values z_0 and ϕ_0,

respectively, and the swing foot position and velocity are defined by a cycloid function:

$$\dot{\boldsymbol{u}}_{t,d} = (\ddot{z}_d, \ddot{\phi}_d, \ddot{p}_{t,d,x}, \ddot{p}_{t,d,z}) \tag{5.42}$$

$$\ddot{z}_d = K_{p,z}(z_0 - z) + K_{d,z}(0 - \dot{z}) \tag{5.43}$$

$$\ddot{\phi}_d = K_{p,\phi}(\phi_0 - \phi) + K_{d,\phi}(0 - \dot{\phi}) \tag{5.44}$$

$$\ddot{p}_{t,d,x} = K_{p,p_{t,x}}(p_{t,r,x} - p_{t,x}) + K_{d,p_{t,x}}(p_{t,r,x} - \dot{p_{t,x}}) \tag{5.45}$$

$$\ddot{p}_{t,d,z} = K_{p,p_{t,z}}(p_{t,r,z} - p_{t,z}) + K_{d,p_{t,z}}(p_{t,r,z} - \dot{p_{t,z}}) \tag{5.46}$$

$$p_{t,r,x}(t) = \frac{\Delta x_l}{2}(t - \sin t) \tag{5.47}$$

$$p_{t,r,x}(t) = \frac{\Delta z_h}{2}(1 - \cos t) \tag{5.48}$$

Where time parametric reference trajectories $p_{t,r,x}(t)$ and $p_{t,r,z}(t)$ are the cycloid functions, with Δx_l is the step length and Δz_h is the step height. Note that the use of a cycloid function means that time needs to be properly scaled and managed from one phase to another. This approach needs to maintain the current wall time t_{wall} and the time since the last phase transition T_{lpc}. It also has a parameter which is the total time of the single support phase T_{ss}. To get the appropriate scaled time t used in the cycloid function, time goes through the following transformation:

$$t = \frac{2\pi}{T_{ss}}(t_{wall} - T_{lpc}) \tag{5.49}$$

The single support phase transitions into the double support phase after a touchdown or timeout event has occurred. Touchdown occurs when the foot has come into contact with the ground, which can be measured as a deviation of the swing foot from its cycloid trajectory:

$$\{||{}_B\boldsymbol{r}_{BP_i} - {}_B\boldsymbol{r}_{BP_i,d}|| > \epsilon\} \tag{5.50}$$

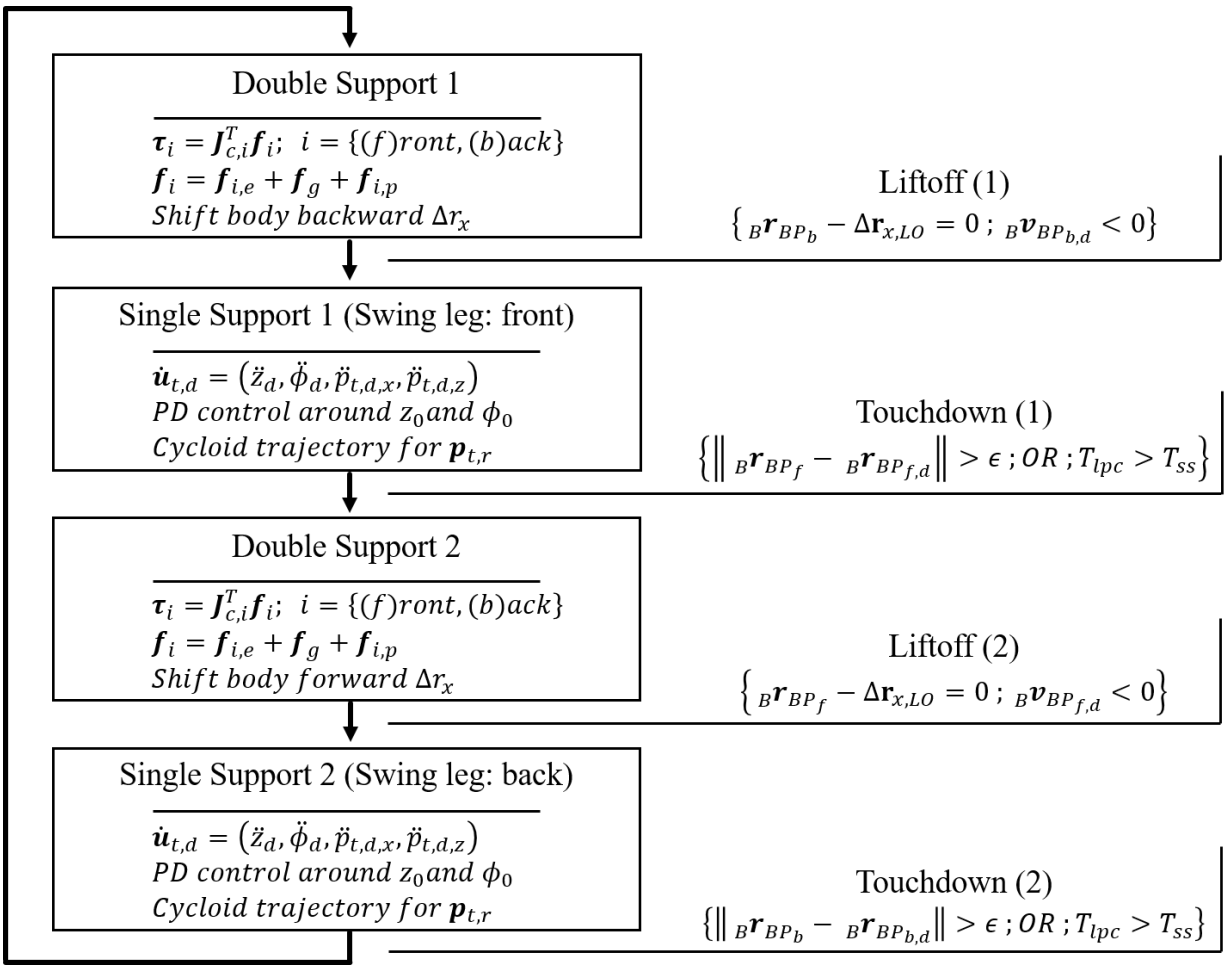


Figure 5.6: State machine describing the VCOS walking approach.

Timeout occurs when the time since the last phase change is greater than the single support time (ie. the cycloid trajectory has completed one period):

$$\{T_{lpc} > T_{ss}\} \tag{5.51}$$

In both instances, the current position of the feet are stored for use in the following double support phase. The overall state machine for the VCOS approach to walking is shown in Figure 5.6.

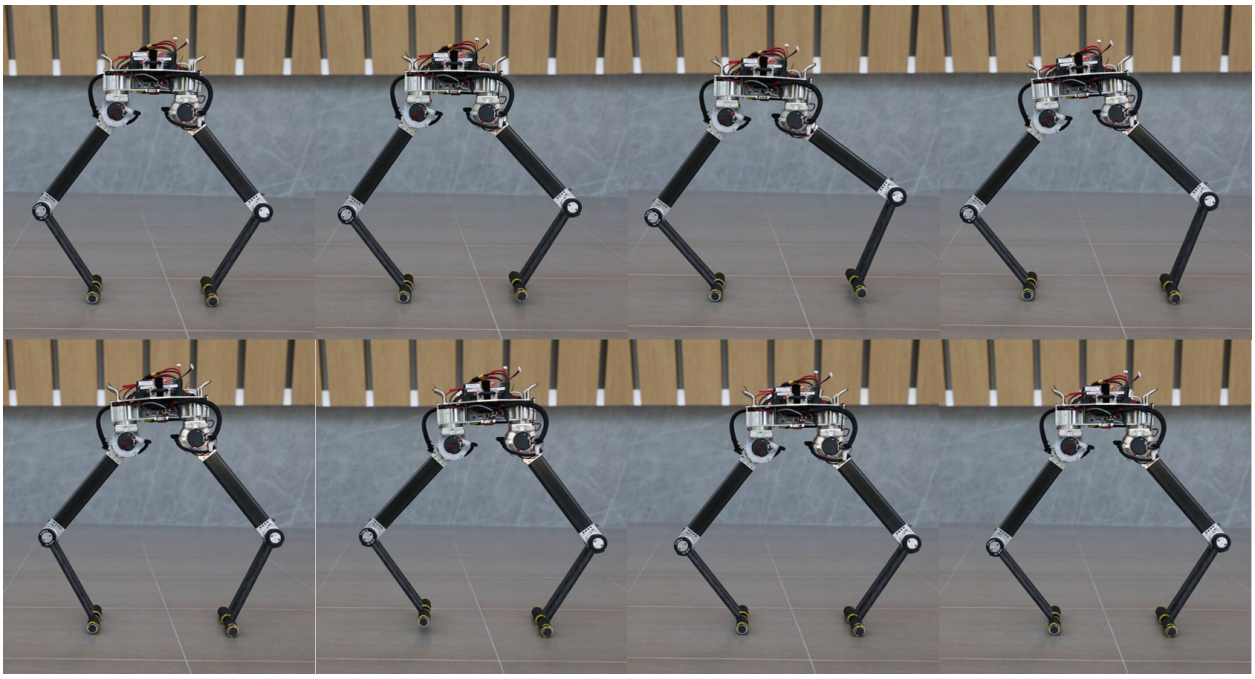


Figure 5.7: One step cycle of the VCOS controller on NABi-2.

5.3.3 Walking Results

Walking on NABi-2 is more difficult than on NABi-1 due to the fact that there are no compliant 'training wheels' on NABi-2, and because the proprioceptive bipeds do not have as many developed methods for locomotion. The dual-impedance approach to walking was unable to effectively walk, but was useful for revealing the importance of dynamics, and for demonstrating how event based compositional controllers can be used alongside proprioceptive actuators to achieve a reactive walking style that does not rely solely on accurately executing pre-planned motions. With the extension to the VCOS approach to walking, NABi-2 is able to walk without tipping over. One aspect that may improve the performance of the VCOS approach would be to find some optimal point at which to transition from the double support phase to the single support phase. Currently, this transition point is tuned

empirically, and is quite sensitive to step size. Still, the event based construction of the controller allows the robot to recover from small disturbances, such as slipping. One walking cycle is depicted in Figure 5.7.

5.4 Directional Jumping

A symmetric, in-place jumping/pronking gait was the first type of locomotion developed on NABi-2, as described in Chapter 3. However, this method only controlled the thrust and body pitch in stance, and could drift forward or backward arbitrarily. Once a global state estimate was implemented on NABi-2 it became possible to feed back the body velocity in order to extend the in-place jumping controller to also regulate the directional motion. However, directly applying a Raibert or capture-point style controller to the system proved challenging at the start due to the asymmetric distribution of mass on the robot. The solution to this was to lower the nominal stance of the robot, reducing the effects from the motion of the upper leg on the body and making the system closer resemble a massless leg system.

5.4.1 A Multi-Leg Raibert Controller

The Raibert Hopper was a seminal platform that marked one of the first uses of compliance for locomotion on a legged robot [97]. The classical hopper comprised of a single prismatic leg which had a compressed air piston that acted as a variable stiffness spring which was attached to a (relatively high inertia) body which housed hydraulic actuators which positioned the leg. The robot controller was composed of three components tied together by a state machine:

A hopping height controller, a body orientation stabilization controller, and a directional velocity controller. These control modules combined together form the eponymous Raibert Controller which has inspired many robots and controllers of the modern era [51, 58, 28]. The Raibert Controller works well on robots that resemble the Raibert hoppers: that is, robots with compliant, low inertia legs, such as modern proprioceptive platforms. The original jumping controller on NABi-2 was inspired by the Raibert Controller in that it also utilized a few controller modules tied together by a state machine. However, the NABi-2 jumping controller did not control the velocity, as there was no velocity feedback available at the time.

Once a state estimate was available to the robot, directional jumping could ostensibly be implemented by utilizing an approach similar to Raibert's. However, directly implementing Raibert's foot placement controller for velocity with no modification to the other component controllers in the state machine did not yield successful locomotion. To see the reason as to why this did not work, it is useful to examine how exactly Raibert's methods were implemented. For the Raibert hoppers, the leg was actuated to a certain angle with respect to the body during the flight phase to control the angle at which the robot leg touched down at. Raibert intuited that for any desired velocity there exists a placement of the foot (angle of the leg) prior to touchdown that will cause the robot to achieve that velocity at the end of the subsequent stance phase (while also under the influence of the hopping height and body orientation controllers). It has since been shown that this is true when the system is modeled as a point mass with a massless leg spring (the SLIP model) [43]. This model implies that the leg can be arbitrarily positioned in flight without inducing rotation in the

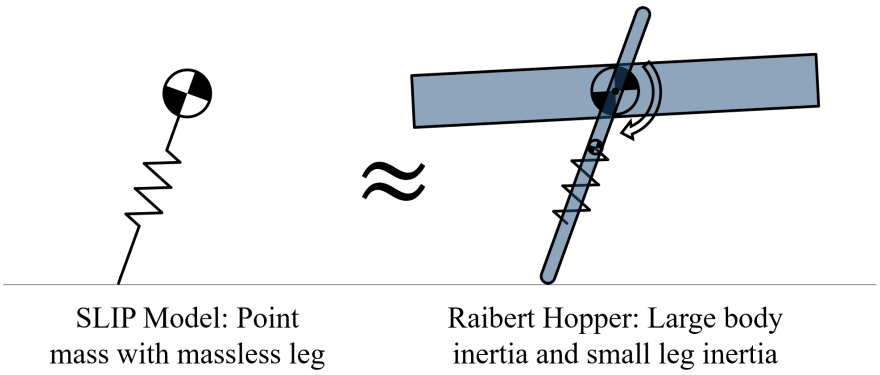


Figure 5.8: The SLIP template model is approximated by the classical Raibert Hopper which has a large body inertia compared to its leg inertia.

body, as there is not even a rigid body to be rotated. In stance, the body cannot impart moments on the leg, meaning the position of the body mass is completely reliant on the spring dynamics. Raibert intuitively designed his hoppers to closely resemble this model by designing the body like a dumbbell to ensure the leg mass and inertia was much lower compared to that of the body (Figure 5.8). This effectively produces the desired qualities of being able to arbitrarily reposition the leg in flight, and execute spring dynamics in stance. This key insight begins to reveal the reason why NABi-2 was unable to directly adopt this approach.

NABi-2 has multiple legs whose hip joints are not coaxial and perpendicular to the direction of motion (as a typical humanoid might have). This makes it more closely resemble a quadruped whose front and rear pair of legs are perfectly synchronized in motion. To perform the Raibert style foot placement controller, NABi-2 would have to actuate both its legs in the same direction to place the *virtual leg* defined as going from the body center to the centroid of all the feet at an angle which would induce the desired velocity at the end

of the subsequent stance phase. When jumping in place, the actual legs of the robot are (in theory) symmetrically configured and executing mirrored commands, so many of the lateral forces and moments induced by the legs on the body are cancelled. This then makes the virtual leg and body resemble a simple, single dimensional spring-damper system. When the angle of the virtual leg is changed, however, the actual legs are no longer performing symmetric moves, and the moments induced by the legs on the body while repositioning in flight cause a large pitching of the body to occur. This in turn causes the front leg to hit the ground prematurely, destabilizing the jumping motion.

The solution to reducing the amount of body pitching that occurs in flight was to reconfigure the nominal stance of the robot. By lowering the body closer to the ground and spreading apart the legs, the robot is able to change the angle of the virtual leg without pitching the body. In this configuration, shown in Figure 5.9, the upper leg segments (femur) are nearly parallel to the ground, and the lower leg segments (tibia) are nearly perpendicular. Now, moving the position of the foot forward or backward mainly depends on rotating the tibia at the knee joint, rather than the femur at the hip. Minimizing the femur motion minimizes the undesirable body pitching, and thus makes it possible to implement the foot placement law to control the velocity of the directional motion.

5.4.2 Implementation on NABi-2

The approach used for directional jumping extends the in-place jumping controller from [136] by adding a foot placement component which utilizes the velocity in the state estimate. The three phases of flight, stance, and thrust remain, but some additional motion is added to

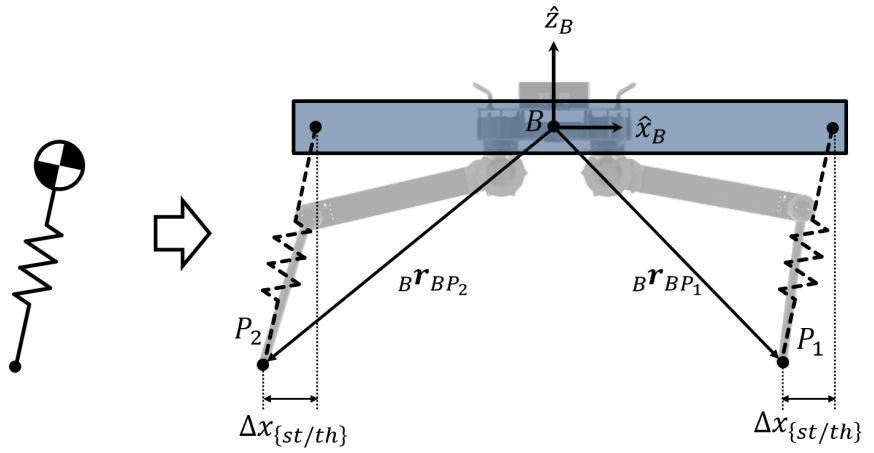


Figure 5.9: The SLIP template model is approximated on NABi-2 by lowering the body and embedding a pair of virtual spring legs.

the flight and thrust phases to accomplish the directional jumping. For this approach, the Jacobian approach for force control about a foot setpoint described by (5.6) and (5.7) is used. The benefit of this framework is that the thrust and the position of the foot can be easily modified by simply changing the position of the feet with respect to the body ${}_B\boldsymbol{r}_{BP_i}$. Initially, the feet start configured with a nominal position:

$$ {}_B\boldsymbol{r}_{BP_1,d} = \begin{bmatrix} p_{x,0}, & p_{z,0} \end{bmatrix}^T \tag{5.52} $$

$$ {}_B\boldsymbol{r}_{BP_2,d} = \begin{bmatrix} -p_{x,0}, & p_{z,0} \end{bmatrix}^T \tag{5.53} $$

Where $p_{x,0}$ and $p_{z,0}$ are the horizontal and vertical offsets for the i^{th} foot in the nominal stance. As expected, the front and back legs are symmetric about the body center, so the x component of each foot are equal and opposite. Also recall that this nominal configuration has the tibia links nearly parallel to the ground and femur links nearly perpendicular, so the physical values of the offsets are close to the lengths of the tibia (for the horizontal offset) and femur (for the vertical offset).

From this nominal configuration, the robot goes into the stance phase. In the stance phase, the robot maintains the nominal configuration, but the gains in (5.8) are significantly reduced, allowing the robot to sink down and trigger the *nadir* event. The stance phase is the same as its corresponding phase in the original in-place jumping controller (though with nominal foot positions set to (5.52) and (5.53)).

Once the robots reaches its nadir, the thrust phase activates. The thrust phase is modified from its in-place equivalent by adding a forward thrust in addition to the vertical thrust that moves the robot towards liftoff at the desired horizontal velocity, $v_{d,x}$. This is done by adding on additional offsets to the nominal foot position:

$$ {}_B\boldsymbol{r}_{BP_1,d} = \begin{bmatrix} p_{x,0} + \Delta x_{th}, & p_{z,0} + \Delta z_{th} \end{bmatrix}^T \tag{5.54} $$

$$ {}_B\boldsymbol{r}_{BP_2,d} = \begin{bmatrix} -p_{x,0} + \Delta x_{th}, & p_{z,0} + \Delta z_{th} \end{bmatrix}^T \tag{5.55} $$

$$ \Delta x_{th} = sat(-K_{step,th} v_{d,x} - K_{capt,th}({}_W v_{WB,x} - v_{d,x})) \tag{5.56} $$

$$ \Delta z_{th} = \Delta z_{ext} + \Delta z_{pitch}|_{front} \tag{5.57} $$

Where the Δx_{th} term is comprised of $K_{step} v_{d,x}$, a feedforward term based on the desired velocity, and $K_{capt}({}_W v_{WB,x} - v_{d,x})$, a feedback term that gains the error in the desired velocity and the current velocity. Note that the $sat()$ function saturates the value of Δx_{th} with a certain bound to avoid the kinematic limits of the robot. The Δz_{th} term includes the vertical leg extension Δz_{ext} which imparts the impulse needed to achieve liftoff, and one more term $\Delta z_{pitch}|_{front}$ which is a small additional extension applied only to the leading leg (the leg which faces the direction of travel) and has the effect of slightly pitching the body backwards right before liftoff to help compensate for the forward pitching that occurs in flight from the

leg motion. The thrust phase leads to the liftoff event, which is simply triggered when the legs have reached the new setpoint (this is typically not actually when litoff occurs, but for the purposes of this controller it is effective).

The final phase in the cycle is the flight phase, which again simply modifies the feet position The feet position in flight are modified as follows:

$$ {}_B\boldsymbol{r}_{BP_1,d} = \begin{bmatrix} p_{x,0} + \Delta x_{fl}, & p_{z,0} \end{bmatrix}^T \tag{5.58} $$

$$ {}_B\boldsymbol{r}_{BP_2,d} = \begin{bmatrix} -p_{x,0} + \Delta x_{fl}, & p_{z,0} \end{bmatrix}^T \tag{5.59} $$

$$ \Delta x_{fl} = sat(K_{step,fl} v_{d,x} + K_{capt,fl}({}_W v_{WB,x} - v_{d,x})) \tag{5.60} $$

Where similar to the thrust phase, there is a feedforward component and a feedback component that modifies the horizontal position of the foot, but unlike the thrust phase there is no adjustment of the vertical foot position. Note that the gains K_{step} and K_{capt} are tuning parameters that are typically not the same for flight and thrust phases. Also notice that the flight and thrust phases have similar structure for the Δx component, but have opposite signs. This can be interpreted as having the foot being behind the body right before liftoff, but ahead of the body before touchdown, as one may expect of a typical running or pronking gait. The flight phase ends with the touchdown event, returning the system back to the stance phase. One complete jump cycle of the robot jumping to the right can be seen in Figure 5.10.

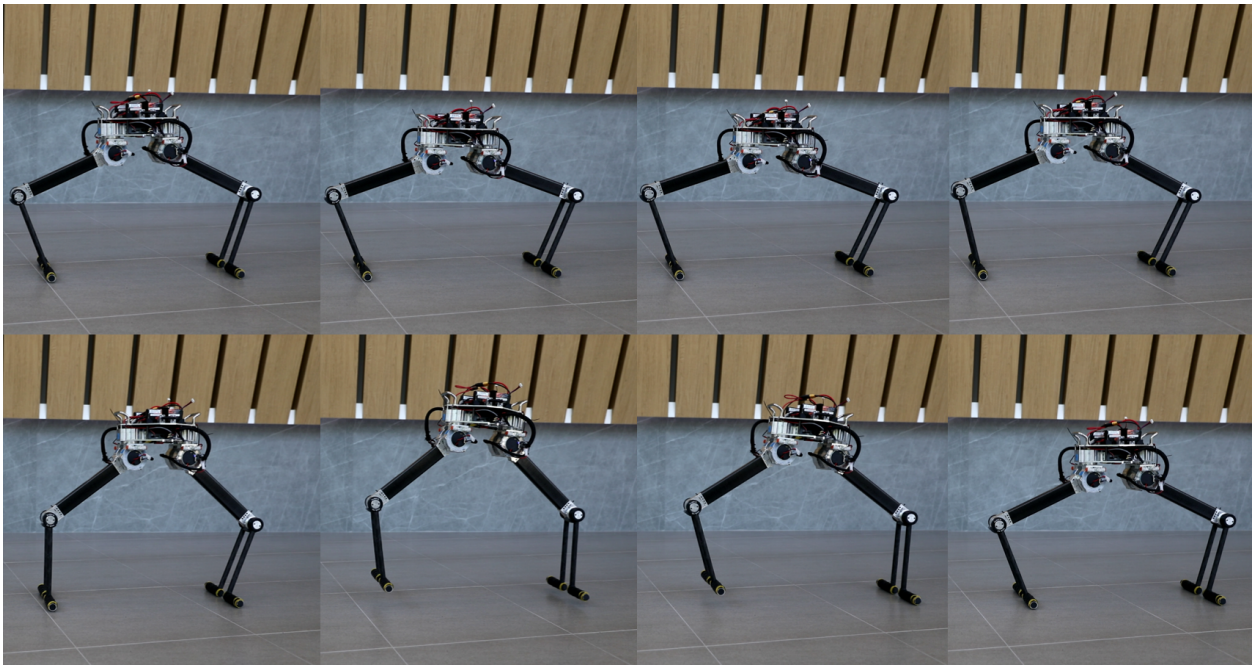


Figure 5.10: One jump cycle of the multi-leg Raibert controller on NABi-2.

5.5 Discussion

Locomotion on NABi-2 was more challenging to implement than on NABi-1 due to the relatively limited prior literature on bipedal locomotion using proprioceptive actuators. It was further hindered by the fact that the body of NABi-2 has relatively low inertia compared to the legs, which cause many of the assumptions associated with proprioceptive legged robots to be broken.

These results suggest certain improvements to the design of the NABi-2 system. For one, the inclusion of a torso would be useful in that an upper body would increase the inertia of the body as a whole, reducing the impact of leg motions on the body. On NABi-1, the cardboard box 'head' adds a significant amount of inertia to the body structure, and has a large impact on enabling the walking motion. It is likely that adding an upper body

to the current iteration of NABi-2 would help with the pitching problem that plagued the development of walking, but would also likely require more torque than the current actuators can supply. Already, during VCOS walking, the knee actuator of the leg in stance is near saturation. This could be solved with a more energy efficient controller that makes use of optimization, but would also likely be less robust to disturbances. A true redesign of NABi-2 would require some level of co-optimzation of the design and the controller.

In spite of the difficulties with walking, two legged directional jumping (pronking) works quite well when the body is lowered towards the ground. This approach to locomotion is surprisingly efficient due to the fact that the proprioceptive BEAR modules regenerate some energy during landing (in a similar fashion to regenerative braking). However, the main disadvantage of this sort of locomotion is its highly dynamic nature, meaning crashes are more catastrophic.

5.6 Conclusion

The proprioceptive NABi-2 platform is leaps and bounds ahead of its predecessor, NABi-1, in terms of dynamic capacity thanks to its proprioceptive actuation. However, There are many challenges introduced when developing a proprioceptive biped, meaning the original algorithms utilized for NABi-1 were not directly transferable to NABi-2. This resulted in the development of new approaches to locomotion on NABi-2: one for walking, and one for pronking. NABi-2 is able to walk using an operational space approach which constrains certain degrees of freedom using task space dynamics, but is able to react to events that occur along the way. Pronking is achieved using a Raibert style controller that controls

jump height, body pitch, and foot placement, but only works when the body is lowered to the ground to reduce the effects of the leg dynamics of the body. Potential future work on NABi-2 would likely involve a co-optimization of the design and control in order to allow the passive dynamics of the system to be better utilized in the locomotion.

CHAPTER 6

Energetic Efficiency of Jumping

Abstract

Embedding the dynamics of the Spring Loaded Inverted Pendulum (SLIP) and applying a compositional controller around it can simplify dynamic legged robot locomotion control, but what is the energetic cost of this convenience? This paper measures the magnitude of this effect in such a way that the results are applicable to a wide class of jumping robots. A three-link monoped model with revolute joints is used to compare the energetic costs of locomotion using two different control approaches: 1) SLIP-embedding with a Raibert-style controller optimized for energetic efficiency, and 2) a trajectory optimized only for energetic efficiency. By performing this comparison in simulation for a large number of different monopeds randomly sampled from a space of realistic robot designs, it is found that the SLIP-Raibert approach requires, on average, almost twice the energy of the trajectory-optimized controller to traverse a given distance. Furthermore, the increase in energetic cost does not depend much on the particulars of the robot design, as the SLIP-Raibert approach requires at least 50% more energy for approximately 88% of realistic robot designs.

6.1 Introduction

Jumping monopeds have been a staple of legged robotics for several decades thanks to the use of high power density series elastic actuators and intuitive heuristic controllers. The original Raibert Hopper instituted this paradigm with a prismatic pneumatic actuator that acted like a linear spring and could be pressurized during the stance phase to create a thrusting effect [97]. Many systems have since been implemented that follow an approach similar to Raibert, and the premise has been expanded to incorporate not only monopeds, but also bipeds and quadrupeds as well [109, 27, 58, 51].

Much of the success of these systems can been attributed to their mimicking of the Spring Loaded Inverted Pendulum (SLIP) model. This model has been shown to describe the center of mass (CoM) dynamics of various animals performing high speed locomotion with flight phases (i.e. running) with surprising accuracy [12]. Furthermore, a large body of work has been developed regarding the SLIP model, showing its application in both running and walking as well as its ability to be generalized to three dimensions [110]. Because of this, legged robotic systems designed to perform hopping, running, trotting, and bounding gaits are often designed with the SLIP dynamics in mind by adding series elastic elements to the physical realization or implementing the SLIP dynamics through control design.

Until recently, the most successful implementations of these SLIP-like systems were achieved through mechanical springs or pneumatic/hydraulic actuators. This is due to the fact that running and jumping requires high torques, and electric motor actuators traditionally achieve this through high gear ratio gearboxes that are likely to be damaged when faced

with the large impulses experienced during the ground impact following an aerial phase. These systems with added compliance often have complex dynamics, making them difficult to control at best and restricted in their behavior at worse. However, recent advances in electric motors have shown that electric actuators can be made to provide sufficient power and torque for dynamic running motions without the need for high gearing that can be damaged during landing [115, 33, 71]. This technology enables mechanical designs with electric motors but no springs that can be accurately modeled with torque inputs at each joint and can be controlled to follow certain dynamics with potential impact events.

To this end, several approaches to reproduce SLIP-like dynamics on articulated legged systems with revolute, torque-actuated joints without series elastic elements have been introduced. Poulakakis and Grizzle utilized geometric nonlinear control design to embed an extension of the general SLIP model on their hopping monoped system in [92]. Hutter et al. took an operational space approach to impose the SLIP dynamics on their simulated leg while also modeling the energy dissipation of ground impact to regulate hopping height and running speed in [60]. Wensing and Orin were able to embed the full 3D-SLIP dynamics into the legs of a simulated humanoid model that was able not only to run but also to maintain the configuration of its body through the use of operational space control and conic optimization in [131]. Park et al. utilized SLIP-like dynamics when implementing their Impulse Planning approach to quadruped bounding on the MIT Cheetah 2 with high torque density actuators in [86]. These approaches show the utility of the SLIP dynamics, and how they can be effectively applied to a legged system to achieve dynamic running and bounding motions.

Though the SLIP dynamics have been shown to be an effective foundation on which to

build a dynamic running controller, the energetic effects of embedding it in physical systems with articulated legs is not often discussed. In this paper, a three-link monoped is introduced and is used as a general robot architecture on which the energy efficiency of different controllers is examined. Specifically, using a Monte Carlo approach, many different robot designs with this architecture are randomly sampled, and the optimal energetic efficiency of an unconstrained controller found through pseudospectral optimal control (*trajectory-optimized*) is compared to that of a simple Raibert-style controller with embedded SLIP dynamics (*SLIP-Raibert*), in order to infer the average efficiency loss incurred across several robot designs when implementing the SLIP-Raibert controller on the monoped system. The main contribution of this work is to show how substantial the energetic costs associated with implementing a SLIP-Raibert controller on a monoped hopper with revolute joints are in order to better inform future legged robot design and implementation.

The remainder of the paper is organized as follows. Section 6.2 poses the questions to be answered through the Monte Carlo experiment and details the approach for conducting the tests. Section 6.3 presents the modeling and dynamics of the three-link monoped through a single step and introduces the design space from which specific robot designs will be sampled. Sections 6.4 and 6.5 present the two control approaches: SLIP-Raibert and trajectory-optimized. Section 6.6 presents the results of the experiment and Section 6.7 reasons about the meaning of these outcomes and identifies topics of future study. Section 6.8 ends the paper with conclusions.

6.2 Experiment Methods and Design

6.2.1 Questions

The methodology utilized in this paper is the same as that of [48], in which certain biomimetic design principles can be extracted through *in silico* testing. The experiment designed in this work is designed to answer the following two questions: 1) Is there typically a significant increase in energy use when embedding SLIP dynamics? 2) How large is this change, on average? These questions can be asked more specifically as follows:

1. For what fraction of the space of realistic three-link monoped designs do locally optimal SLIP-Raibert controllers have at least a fifty percent greater mechanical cost of transport than a locally optimal trajectory-optimized controller?

2. Across the space of realistic three-link monoped designs, what is the average increase in the mechanical cost of transport from using a locally optimal SLIP-Raibert controller over a locally optimal trajectory-optimized controller?

Here, the mechanical cost of transport (CoT) [24] is being used as a measure of the overall energetic efficiency of the system through a step. It is defined as the energy consumed by the actuators normalized by the robot's weight and distance traveled. The energy consumed is assumed to be the integral of the absolute mechanical power of the actuators, as

$$J_{CoT} = \int_0^{t_{step}} \frac{|\tau \cdot \dot{\theta}|}{M g x_{step}} dt \tag{6.1}$$

where t_{step} is the time it takes for a step to be taken, x_{step} is the distance traveled in one step, τ is the vector of actuator torques, $\dot{\theta}$ is the vector of angular rates of the joints, M is

the total system mass, and g is gravity. Here, a *step* is defined to begin at the apex of the flight phase and end at the apex of the next flight phase.

Questions 1 and 2 can be answered using Monte Carlo methods which are used to help understand the relationship κ between a controller η and the CoT $J_{CoT} = \kappa(\eta)$. This is achieved through taking random robot samples P_i from a design space P, finding the CoT of an optimal SLIP-Raibert controller, η_{SR}, and comparing the results to the CoT of a trajectory-optimized controller, η_{TO}, to make an inference about the entire population. These methods are common in biology [80] and are first used to analyze robot design features in [48].

Specifically, question 1 can be answered by using N random samples from the population to approximate the true percentage, $g(\eta_{SR})$, with the observed percentage:

$$\tilde{g}(\eta_{SR}) = \frac{\sum_{i=1}^{N} H(\kappa(\eta_{SR}, P_i) - K_{\%}\kappa(\eta_{TO}, P_i))}{N} \tag{6.2}$$

where H is the Heaviside step function, $\tilde{g}(\eta_{SR})$ is a random variable, and $K_{\%} = 1.5$ is a scaling factor based on the percent difference in question 1. Then, $\tilde{g}(\eta_{SR})N$ follows the binomial distribution of N experiments and unknown proportion of success $q = g(\eta_{SR})$, i.e. $\tilde{g}(\eta_{SR})N \sim B(N, g(\eta_{SR}))$. This information is used to test the *null hypothesis* — the likelihood of the observed $\tilde{g}(\eta_{SR})$ being produced with an assumed q_0. A binomial proportion confidence interval can then be generated on this estimate.

Question 2 is answered by using N random samples to approximate the true mean, $h(\eta_{SR})$, as the sample mean:

$$\tilde{h}(\eta_{SR}) = \frac{\sum_{i=1}^{N} \kappa(\eta_{SR}, P_i) - \kappa(\eta_{TO}, P_i)}{N} \tag{6.3}$$

Where $\tilde{h}(\eta_{SR})$ is again a random variable of unknown distribution. However, an estimate of confidence interval for the true mean can be estimated with a reference distribution using methods such as bootstrapping [19]. In bootstrapping, the reference distribution is generated by randomly selecting a number of resamples with replacement from sample differences, $\kappa(\eta_{SR}, P_i) - \kappa(\eta_{TO}, P_i)$, equal to the number of robot designs tested, taking the mean of these resamples, and repeating this process 10000 times.

6.2.2 Experiment Procedure

The process in which data was collected for this experiment is summarized in Algorithm 1. A total of 100 three-link monoped designs (P_i) are randomly sampled from the design space (P) detailed in Section 6.3. For each design, 50 attempts are made to find the most efficient locally optimal trajectory-optimized controller (η_{TO}) and 50 attempts are made to find the most efficient locally optimal SLIP-Raibert controller (η_{SR}). These many attempts at optimization are made for each approach to try and find local optima as close to the global optimum as possible. The CoT J^*_{CoT} yielded from controllers η_{TO} and η_{SR} are then compared for all robot designs using (6.2) and (6.3). This data only represent a sparse sample of a relatively large space, but the statistical methods used here still allow us to answer questions 1 and 2 with some confidence interval.

Algorithm 1 Data Collection

for $N = 100$ (Robot designs) **do**
  Randomly sample a robot design P_i
  for Control approach $j = \{SR, TO\}$ **do**
    Initialize most efficient CoT, $J^*_{CoT,j} = \infty$
    for $n = 50$ (Optimization trials) **do**
      Find a locally optimal controller η_j from a random initial guess
      if Optimization succeeds **then**
        Calculate the CoT, $J_{CoT,j} = \kappa(\eta_j, P_i)$
        if $J_{CoT,j} < J^*_{CoT,j}$ **then**
          $J^*_{CoT,j} = J_{CoT,j}$
        end if
      else
        Discard the result of this optimization trial
      end if
    end for
  end for
end for

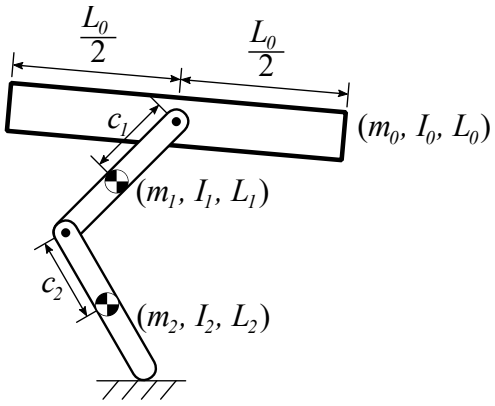


Figure 6.1: Three-link monoped model.

Parameter	Min	Max	Unit
m_0	1	10	kg
L_0	0.5	1.5	m
$\frac{L_1}{L_1+L_2}$	33.3	66.6	%
$\frac{L_1+L_2}{L_1+L_2+L_0}$	33.3	50.0	%
$\frac{m_1}{m_1+m_2}$	33.3	80.0	%
$\frac{m_1+m_2}{m_1+m_2+m_0}$	10.0	25.0	%
$\sqrt{\frac{I_0}{m_0 L_0^2}}$	33.3	66.6	%
$\sqrt{\frac{I_1}{m_1 L_1^2}}$	20.0	66.6	%
$\sqrt{\frac{I_2}{m_2 L_2^2}}$	20.0	66.6	%
$\frac{c_1}{L_1}$	25.0	75.0	%
$\frac{c_2}{L_2}$	25.0	75.0	%
Fr	1	2	-

Table 6.1: Model Dimensionless Parameters

6.3 Three-Link Monoped Model

Three-Link Monoped

The overall structure of the monoped model used for this experiment is shown in Fig. 6.1. It is comprised of a single body link that is attached at the hip to a single leg with an upper and lower link joined at the knee. This model is reminiscent of Raibert's original hopper design, with the prismatic shank replaced with a revolute knee. Each link on the robot is characterized by its mass, length, CoM location, and moment of inertia about the CoM. A single robot design is defined by a choice of these physical parameters.

Reasonable bounds on the ranges of physical parameters are imposed in order to limit the possible three-link monoped designs to those that would be physically realizable and applicable to robot designers. The limits, shown in Table 6.1, were chosen to be similar to those in [49], which were selected based on systems seen in nature and in previously designed legged robots. Some limits are more restrictive due to the fact that the system only has one leg and to avoid numerical difficulties in the optimization process. These limits are introduced by bounding the upper and lower values of dimensionless parameters that relate the various physical parameters, with the exception of the body mass and length. Note that by the Buckingham π theorem [19], this dimensionless re-parameterization makes the results applicable to robots of any scale. Body mass and length values are only given to perform the numerical calculations and provide physical interpretation. Now, a robot design can be uniquely defined by sampling from a uniform distribution on the interval between the lower and upper value of each dimensionless parameter. Some sample robot designs are shown in

Fig. 6.2.

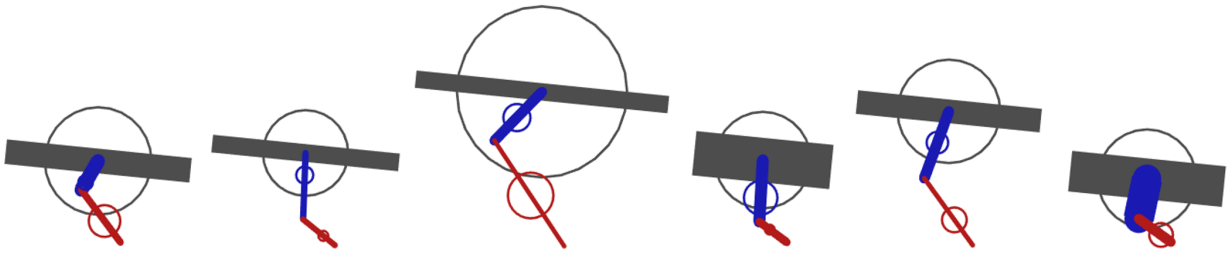


Figure 6.2: Six randomly sampled robot designs from the experiment. Link lengths and masses are represented by the length and thickness of each link. The center of each circle is the center of mass of the link, and the radius is the link radius of gyration. This variety of robot designs allows us to make inferences about the total design space.

Model Dynamics

The running motion examined in this experiment has two phases subject to gravity: a flight phase in which the robot is in the air, and a stance phase during which the foot is joined with the ground with a pin joint. In the flight phase, the system has five degrees of freedom: one angle for each link's orientation, and two Cartesian coordinates that define the hip's location in the inertial frame. In the stance phase, the system can be reduced to three degrees of freedom because the foot is constrained with respect to the inertial frame. The motion of the system is influenced by gravitational forces, ground reaction forces (GRF) in stance, and torque inputs produced by actuators at the hip and knee joints. The resulting equations of motion with all degrees of freedom present are described by the familiar form:

$$H(q)\ddot{q} + C(q,\dot{q}) + G(q) = J_s(q)^T F_s + S_a^T \tau \tag{6.4}$$

where $H \in \mathbb{R}^{5\times5}$, $C \in \mathbb{R}^5$, and $G \in \mathbb{R}^5$ are the inertia matrix, velocity products, and gravitational terms, respectively. Additionally, $F_s \in \mathbb{R}^2$ is a vector of external forces, which are the (GRF) in stance and zero in flight, and $J_s \in \mathbb{R}^{5\times2}$ is the Jacobian that relates end effector forces to joint torques. The actuator selection matrix $S_a = [0^{2\times3} \;\; \mathbb{I}^{2\times2}]$ separates the actuated and unactuated degrees of freedom, and $\tau \in \mathbb{R}^2$ is the vector of input torques.

In the transition from flight to stance (touchdown), the foot impact is modeled as an impulsive and perfectly inelastic collision, where the position states of the system after the impact remain the same and the velocity states are adjusted based on angular momentum conservation. The transition from flight to stance (touchdown) is defined as the point when the end of the leg (foot) touches the ground. The transition from stance to flight (liftoff) is defined as the point when the vertical GRF goes to zero; equivalently, when the acceleration of the CoM equals the gravitational acceleration. The equations of motion and impact dynamics are derived for each robot design from Lagrangian dynamics using the MATLAB Symbolic Math Toolbox.

6.4 SLIP Embedded Raibert Controller

2D-SLIP Model

The 2D-SLIP (also known as spring-mass) model is a point mass m at the end of a massless, springy leg, as shown in Fig. 6.3. The mass follows ballistic dynamics in flight during which the massless leg can be positioned for the upcoming stance phase. In stance, a linear spring with constant stiffness k and rest length l_0 apply forces to the mass. The flight dynamics are

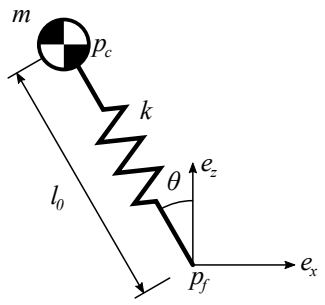


Figure 6.3: 2D-SLIP model.

simply $m\ddot{p}_c = mg$, where $g \in \mathbb{R}^2$ is the gravity vector and $p_c \in \mathbb{R}^2$ is the position of the point mass in inertial coordinates. The foot position in flight, $p_f \in \mathbb{R}^2$, is defined by the angle θ of the leg. In stance, the dynamics of the SLIP model can be written simply as follows:

$$m\ddot{p}_c = k(l_0 - ||l||)\hat{l} + mg \tag{6.5}$$

where $l = p_c - p_f$ and $\hat{l}$ is the direction along the leg.

The switching manifolds of the hybrid SLIP system are defined to occur when the end of the leg touches the ground from flight for touchdown and when the leg returns to its rest length from stance for liftoff. The switching surfaces used on the SLIP-embedded three-link monoped are the same for touchdown, but are modified for liftoff to ensure there is never a case in which the vertical GRF is negative, as this would imply that the foot is able to pull the system towards the ground. The phase switches on the three-link monoped when the state intersects the following manifolds:

$$S^*_{TD} = \left\{(p_c, \dot{p}_c) \mid e_z^T p_c = l_0^- \cos(\theta), e_z^T \dot{p}_c \leq 0\right\} \tag{6.6}$$

$$S^*_{LO} = \left\{(p_c, \dot{p}_c, \ddot{p}_c) \mid e_z^T \ddot{p}_c = e_z^T g, e_z^T \dot{p}_c \geq 0\right\} \tag{6.7}$$

where l_0^- is the length of the 'virtual' leg from the CoM to the foot of the monoped before touchdown.

Embedding Dynamics

To enforce the dynamics of (6.5) onto (6.4), operational space methods are used [29]. The approach used here is similar to that used by [60], which itself is an extension of [87]. This approach utilizes two sets of operational space coordinates: one for the foot (*support space*) and one for the whole body CoM (*task space*). First, the support space coordinate is used to constrain the equations of motion. Afterwards, the task space coordinate can be defined and controlled for desired behaviors.

The operational space equations of motion for the support space are obtained by left multiplying (6.4) by $J_s H^{-1}$ and incorporating the time derivative of the support Jacobian, J_s, as defined by $\dot{x}_s = J_s \dot{q}$:

$$\ddot{x}_s - \dot{J}_s \dot{q} + J_s H^{-1}(C + G) = \Lambda_s^{-1} F_s + J_s H^{-1} S_a^T \tau \tag{6.8}$$

where $\Lambda_s = (J_s H^{-1} J_s^T)^{-1}$. Left multiplication of (6.8) by Λ_s recovers the familiar form:

$$\Lambda_s \ddot{x}_s + \mu_s + \rho_s = F_s + \bar{J}_s^T S_a^T \tau \tag{6.9}$$

where μ_s, ρ_s, and $\bar{J}_s$ follow from the left multiplication. Assuming that in stance the position of the foot is stationary, $\dot{x}_s = \ddot{x}_s = 0$, and F_s can be solved for in (6.9) and input to (6.4) to generate the constrained equations of motion:

$$H\ddot{q} + N_s^T(C + G) + J_s^T \Lambda_s \dot{J}_s \dot{q} = N_s^T S_a^T \tau \tag{6.10}$$

where

$$N_s^T = \mathbb{I} - J_s^T \Lambda_s J_s H^{-1} \tag{6.11}$$

is the dynamically consistent nullspace projector for support.

Next, the desired task space coordinate x_t is defined along with its associated Jacobian, J_t, as defined by $\dot{x_t} = J_t \dot{q}$. The operational space dynamics for this coordinate can now be obtained with respect to the support constrained dynamics by left multiplying (6.10) by $J_t H^{-1}$ and then by $\Lambda_t = (J_t H^{-1} N_s^T J_t^T)^{-1}$. The SLIP dynamics can now be embedded as the task space dynamics, i.e. $\ddot{x}_t = \ddot{p}_c$:

$$\Lambda_t \ddot{p}_c + \mu_t + \rho_t = F_t \tag{6.12}$$

where $\ddot{p}_c$ is obtained by solving for it in (6.5), and μ_t, ρ_t, and F_t follow from the left multiplication.

The control torques necessary to generate the operational space forces F_t are found using the support reduced Jacobian, $\bar{J}_t^T = \Lambda_t J_t H^{-1} N_s$:

$$\tau = (\bar{J}_t^T S_a^T)^{-1} F_t \tag{6.13}$$

These torques are now essentially enforcing SLIP dynamics onto a subset of the original system's configuration space that is consistent with the support constraints, achieving the SLIP embedding.

Raibert Controller

The classical Raibert Controller is an empirical controller that originated from Marc Raibert's seminal work on hopping robots. The controller consisted of three control 'modules'

Variable	Min	Max	Unit
Δl	$0.01(L_1 + L_2)$	$0.5(L_1 + L_2)$	m
$\dot{x}_d$	$0.5v_{x,avg}$	$2.0v_{x,avg}$	m/s
θ^*	$-\frac{\pi}{3}$	$\frac{\pi}{3}$	rad
z_0	$L_1 + L_2$	$1.2(L_1 + L_2)$	m
l_0	$0.6(L_1 + L_2)$	$0.8(L_1 + L_2)$	m

Table 6.2: SLIP-Raibert optimization variable bounds.

that independently regulated certain aspects of Raibert's hopping robots: the vertical jumping height, the horizontal velocity of the locomotion using foot placement, and the pitch of the body. This sort of controller has been formalized as the composition of several templates in [27], which also discusses the assumptions necessary for stability.

The controller used on the SLIP embedded three-link monoped is similar to the Raibert controller in that it utilizes separate modules to control thrust delivered by the leg (6.14), the orientation of the body (6.15), and the positioning of the leg in flight to control running speed (6.16). A thrust is introduced by increasing the nominal spring rest length l_0^- after touchdown by Δl, effectively increasing the potential energy of the leg spring. The pitch control used is the same as that used in the classical Raibert controller: a proportional-derivative (PD) controller about a desired pitch using the hip actuator during stance. The stepping controller is also similar, with the leg being commanded at a joint level PD scheme through flight in order to achieve the desired touchdown angle for the upcoming stance phase.

The different controllers can be described as:

$$l_0^+ = \Delta l + l_0^- \tag{6.14}$$

$$\tau^\phi = -K_p^\phi(\phi - \phi_d) - K_d^\phi(\dot{\phi} - \dot{\phi}_d) \tag{6.15}$$

$$\theta_{TD} = \theta^*(\dot{x}) + K_p^\theta(\dot{x} - \dot{x}_d) \tag{6.16}$$

where l_0^+ is the new spring rest length after a thrust of Δl is applied, τ^ϕ is a torque applied at the hip joint to track a desired pitch and pitch rate, $(\phi_d, \dot{\phi}_d)$, and θ_{TD} is the touchdown angle to track some desired velocity, $\dot{x}_d$, given a neutral touchdown angle $\theta^*(\dot{x})$ [27].

The controller parameters were generated through varying means. The PD gains, such as those that control the leg and body pitch, were empirically tuned and then scaled by mass and inertia to generalize them to the randomly generated systems. Spring parameters such as the spring constant were chosen based on previous work done with SLIP models, notably [116], and scaled by mass. The apex height of the CoM is taken from the solution to the trajectory-optimized controller, and the spring rest length was based on the initial configuration of the leg in the trajectory-optimized controller. If there was no trajectory-optimized controller found, the apex z_0 and leg length l_0 are chosen by sampling from uniform distributions on the interval between the upper and lower bounds of each parameter from Table 6.2. The remaining controller parameters are found through optimization to minimize the CoT.

SLIP-Raibert Optimization

The leg thrust Δl, horizontal velocity $\dot{x}_d$, and neutral touchdown angle at that velocity $\theta^*(\dot{x}_d)$ are chosen by MATLAB's `fmincon` function to minimize the mechanical cost of transport using a 'shooting' approach [5], with `ode45` used for forward dynamics simulation. For this experiment:

- The average velocity across a single step $v_{x,avg}$ is constrained based on the sampled Froude number.
- A periodicity constraint is imposed on the CoM at apex.
- Actuators saturate at torque limits set in Table 6.3.

The initial guess provided to `fmincon` is chosen by sampling from uniform distributions on the interval between the upper and lower bounds of each decision variable, also in Table 6.2.

6.5 Trajectory Optimized Controller

Trajectory optimization is the formulation of an optimal control problem in which the control inputs propagate a system along a trajectory in the state space of the system while respecting the system dynamics and other nonlinear constraints to satisfy some optimality condition. These problems can be formulated as nonlinear programming (NLP) problems, often with sparse gradients, that can be solved using nonlinear program solvers such as IPOPT [7], SNOPT [44], `fmincon`, etc. There are many different methods to formulate these problems, but direct collocation approaches are common today [54, 53, 5]. The trajectory-optimized

Variable	Min	Max	Unit
Δt	0.025	0.25	s
x	0	$v_{x,avg}(3\Delta t_{max})$	m
$\dot{x}$	0	$2v_{x,avg}$	m/s
z	$0.1(L_1 + L_2)$	$2(L_1 + L_2)$	m
$\dot{z}$	$-g\Delta t_{max}$	$2v_{x,avg}$	m/s
θ_0	$-\pi/12$	$\pi/12$	rad
θ_1	0	$\pi/2$	rad
θ_2	$-\pi/2$	0	rad
$\dot{\theta}_i$	$-\frac{\theta_{i,max}-\theta_{i,min}}{\Delta t_{min}}$	$\frac{\theta_{i,max}-\theta_{i,min}}{\Delta t_{min}}$	rad/s
τ_i	$-\Gamma$	Γ	Nm

Table 6.3: Trajectory-optimized decision variable bounds.

controller utilized for the three-link monoped in this experiment was found using GPOPS [89], which implements a direct collocation approach using a pseudospectral integral approximation [89, 40].

Pseudospectral Integral Approximation

System dynamics such as 6.4 with n states and m controls can be put into the form

$$\dot{x}(t) = f(t, x(t), u(t)) \tag{6.17}$$

where t is time, $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, and f is a function describing the system dynamics. The process of the pseudospectral integral approximation begins by selecting N discrete collocation points, $t_i, i \in \{1, 2, ..., N\}$ in time at which the solution needs to exactly satisfy the dynamics of the system. A Lagrange polynomial L_i of order $N-1$ is then constructed corresponding with each t_i such that

$$L_i(t_j) = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{6.18}$$

where the Lagrange polynomial is of the form

$$L_i(t) = \prod_{j=1, h \neq i}^{N} \frac{t - t_j}{t_i - t_j} \tag{6.19}$$

These polynomials can now be used to approximate the trajectory of the state derivatives, ie. the dynamics, over the time interval using

$$\dot{x}_N(t) = \sum_{i=1}^{N} L_i(t) f_i \tag{6.20}$$

where $f_i = f(t_i, x(t_i), u(t_i))$. Now, by design, the dynamics are exactly satisfied at the collocation points, $\dot{x}_N(t_i)$, and are approximated by $\dot{x}_N(t)$ elsewhere. Now, the solution to

the differential equation defined by (6.17) is approximated by

$$x_N(t) = x(t_0) + \int_{t_0}^{t} \sum_{i=1}^{N} L_i(\tau) f_i d\tau \tag{6.21}$$

By noticing that the sum can be taken out of the integral, we can define

$$A_{ji} = \int_{t_0}^{t_j} L_i(\tau) d\tau \tag{6.22}$$

yielding, for each collocation point t_j, the following relationship:

$$x_N(t_j) = x(t_0) + \sum_{i=1}^{N} A_{ji} f_i \tag{6.23}$$

These relationships for each t_j can now be concatenated into a single matrix equation

$$X_N = X_0 + AF(X_N) \tag{6.24}$$

where the jth row of X_N is $x_N(t_j)^T$, every row of X_0 is $x(t_0)^T$, A is the integration approximation matrix defined in (6.22), and the ith row of $F(X_N)$ is f_i^T. Solving this system yields the approximate solution to (6.17), which can then be utilized by the NLP solver to solve the NLP given by:

$$argmin \sum_{i=1}^{N_p} J_{CoT} \tag{6.25}$$

subject to the system dynamics (6.4), defect constraints, boundary conditions, and other path constraints, where $N_p = 3$ is the number of hybrid phases. For this experiment

- Boundary conditions are a periodicity constraint on all robot states.
- Path constraints ensure the feet don't pass below the ground and the foot doesn't exert negative vertical GRF.

- The same average horizontal velocity constraint as in Section 6.4 is used.

The NLP solver is initialized with a random guess sampled from intervals shown in Table 6.3, where Δt is the time of a single hybrid phase, (x, z) and $(\dot{x}, \dot{z})$ are the position and velocity of the hip, θ_i and $\dot{\theta}_i$ are the absolute angle and rate of each link, and $\Gamma = g(L_1 + L_2)(m_0 + m_1 + m_2)$ is the torque required to hold the entire weight of the system at full leg extension. These limits were chosen by considering the physical parameters and average velocity of the system, and are sufficiently generous such that they were not observed to be active in any of trajectory-optimized solutions checked. Further discussion on the different optimization approaches for each controller and their effects are further discussed in Section 6.7.

6.6 Experiment Results

Optimizations successfully converged to feasible solutions for 3738/5000 trajectory-optimized trials and for 4329/5000 SLIP-Raibert trials. Fig. 6.6 shows a trajectory-optimized solution and an apex-to-apex simulation of the system with the optimized SLIP-Raibert controller. When optimization failed for the trajectory-optimized controller, it was usually due to numerical problems or locally infeasible solutions; for the SLIP-Raibert controller, it was due to the leg reaching a singular configuration or the actuators not being able to provide sufficient torque.

The mechanical cost of transport of the successful optimizations are represented in Fig. 6.4. The most efficient (i.e. lowest cost of transport) controller for both control approaches

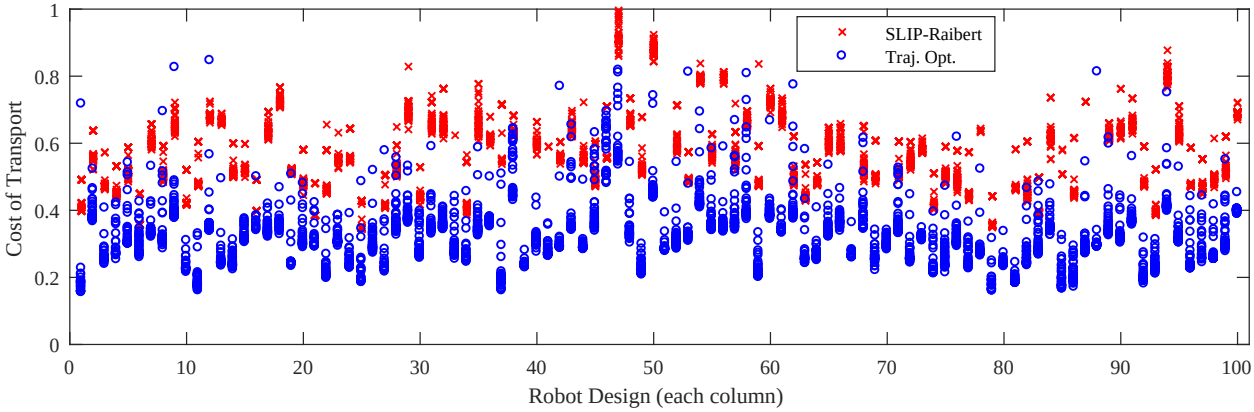


Figure 6.4: Results of the experiment showing for each robot design approximately how much greater the cost of transport is for the SLIP-Raibert controller (red X's) as compared to the trajectory optimized controller (blue O's). The data show that for all robot designs, the most efficient trajectory-optimized controller has a lower cost of transport than the most efficient SLIP-Raibert controller, and clustering suggests the global optimum of the trajectory optimized controller to be lower than that of the SLIP-Raibert controller.

and for all 100 robot designs are input to (6.2) and (6.3) to answer questions 1 and 2.

1. For the space of three-linked monopeds, the most efficient SLIP-Raibert optimized controller had at least a fifty percent greater mechanical cost of transport than the most efficient trajectory-optimized controller 88.0% of the time, with the 95% confidence interval for the true percentage of this result being 79.6% and 93.9%.

2. For the space of three-linked monopeds, the most efficient SLIP-Raibert optimized controller on average had 91.0% greater mechanical cost of transport than the most efficient trajectory optimized controller, with the 95% confidence interval for the true average increase being 83.1% and 99.2%.

Fig. 6.4 demonstrates for each robot design approximately how much larger the cost of transport is for the SLIP-Raibert controller (red X's) as compared to the trajectory optimized controller (blue O's). One might be surprised that any SLIP-Raibert controller is more energy efficient than a trajectory-optimized controller, but this is due to the fact that the data only show local optima. In these instances, the locally optimal solution can have the robot performing unnatural motions, such as oscillating the leg unnecessarily.

Importantly, the data show that with all robot designs, the most efficient trajectory-optimized controller has a lower cost of transport than the most efficient SLIP-Raibert controller. The clustering of data points can also hint at the optimality of the found controllers, because if many randomly seeded local optimizations result in similar local optima, it suggests that the global optima lies near the cluster. Observing the clustering of the CoT data does suggest the global optimum of the trajectory-optimized controller to be lower than

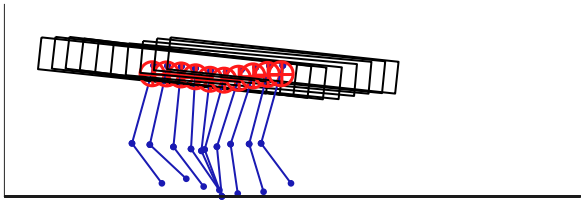


Figure 6.5: Examples of gaits using the trajectory-optimized controller.

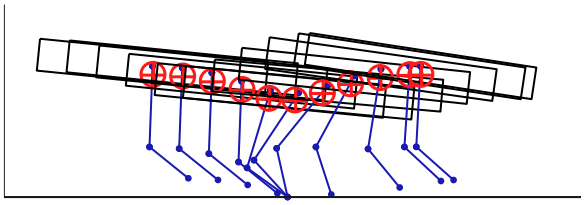


Figure 6.6: Examples of gaits using the SLIP-Raibert controller.

that of the SLIP-Raibert controller. This result, along with the fact that the SLIP-Raibert approach nearly doubled the cost of transport of the trajectory-optimized one, show how much more efficient the unconstrained trajectory-optimized controller is.

6.7 Discussion

Optimization Methodology

It is worth noting that the different controllers did not often converge to steps of similar length, as can be seen by the fact that the distance traveled by the monoped in Fig. 6.5 is much shorter than that in Fig. 6.6. This is likely due to the approach taken in the SLIP-Raibert controller optimization, with certain initial conditions being constrained. The choice to constrain the SLIP-Raibert initial conditions and use a shooting approach was

made to improve the convergence of the NLP solver, and to reduce the occurrence of poorly performing controllers that would place the leg in singular configurations or reach apex before liftoff. This approach could be improved upon by allowing the SLIP-Raibert optimization to find its own initial condition, potentially yielding lower CoT. In spite of these aspects of the SLIP-Raibert controller optimization, the results still hold for this formulation of the SLIP-Raibert controller under the assumption that the initial apex height and leg configuration are set through randomly seeded trajectory-optimized controllers.

Reasons for Energetic Losses

While the experiment methodology used here is not designed to explain the complex coupling interactions between the type of controller used and the resulting CoT, it is compelling to consider why, aside from the effects of the optimization methodology, the SLIP-Raibert controller has such a large cost of transport compared to a trajectory-optimized controller.

One possible reason could be the use of position based feedback control to command the leg configuration in flight and body orientation in stance. While this may be a simple and effective way to handle the positioning and stabilization of certain states, it should be no surprise that this stiff control requires a significant amount of energy.

Another possibility would be that the trajectory-optimized controller attempts to perform a form of swing leg retraction, thus minimizing the foot tangential speed at ground contact, reducing the energy loss at touchdown [50]. The SLIP-Raibert controller simply holds the position of the leg until touchdown, and thus does not attempt to minimize the foot tangential speed. This trend can be observed in Fig. 6.6, where the trajectory-optimized controller will

tend to retract the leg before touchdown.

Future Work

Topics of future work include improving the fidelity of the SLIP-Raibert approach to yield a better approximation of the global minimum that is not as sensitive to constraints and initial conditions. This could be done by formulating an unconstrained problem like the one used to find the trajectory-optimized controller, and then adding in constraints that enforce SLIP dynamics. Additionally, further investigation as to why the SLIP-Raibert approach is so much less energy efficient than the optimal approach could provide key insights as to how robot design can be improved to yield a more energy efficient design while maintaining the strong foundations supplied by the SLIP model and compositional controllers. Another topic would be to investigate means of stabilizing the trajectory-optimized controller over obstacles and rough terrain, and what effects that has on its efficiency. Implementation of these different control approaches on a physical robot to observe the energetic effects also is a compelling follow-up experiment.

6.8 Conclusion

The SLIP model is a very useful template that has a long history of utility in legged robots, and has been paired with the Raibert controller to achieve stable dynamic running and jumping motions. However, when the physical realization of the robotic system does not match the SLIP paradigm, energetic efficiency can be affected substantially. This work investigates a design space of three-linked monopeds and shows that using a classical Raibert

style controller with the SLIP dynamics embedded into the CoM dynamics has on average a 91% larger mechanical cost of transport compared to that of than an trajectory-optimized controller, and requires at least 50% more energy for approximately 88% of realistic robot designs. This result is subject to the qualification that the SLIP-Raibert controller optimization was somewhat restricted in its initial conditions. Interestingly, the fact that the controller was able to find solutions with these varying initial conditions shows the robustness of this SLIP-Raibert approach, despite its high energetic costs. As such, the SLIP-Raibert approach can be used as a simple means of producing a stable running controller, but may want to be avoided in applications where energy efficiency is desired.

CHAPTER 7

Conclusion

7.1 Summary

Bipedal locomotion is a challenging task that requires the careful coordination of limbs and body in order to be accomplished. This work attempts to rethink the paradigm of bipedal locomotion by developing a novel non-anthropomorphic platform that is able to walk effectively using compliant actuation.

The original NABiRoS (NABi-1) platform was developed for planar locomotion and resembles the form of a human taking steps sideways, as a fencer or ballerina might. It has a pair of 2 DoF legs that utilized off-the-shelf position controlled servos. However, the feet were designed to be compliant in order to offer the benefits of series-elasticity in the leg. NABi-1 was eventually able to walk using a ZMP-based preview control approach, and was able to perform other gaits as well through empirical tuning. NABi-1 demonstrated an interesting approach to bipedal locomotion that does not resemble the typical humanoid form, but is still capable of some dynamic locomotion thanks to the use of compliance.

NABi-2 shares the philosophy of NABi-2 in terms of the non-anthropomorphic bipedal concept, but improves on much of the original design. NABi-2 is equipped with propriocep-

tive BEAR modules, actuators which are amenable to force control without the explicit use of force/torque sensors thanks to their high torque transparency. The proprioceptive actuators provide an adjustable compliance, so the spring elements in the feet can be removed. Furthermore, an additional DoF was added at the hip, in order for the leg to be able to achieve full 3D positioning. An in-place jumping or pronking gait was initially developed for NABi-2 to demonstrate its impact resistance and dynamic capabilities. This gait was achieved using a Raibert-style controller that is implemented with a Jacobian-based force controller that only considers the static relationship between the body and the end effector (foot). The controller is event based, which is facilitated by the use of proprioception at the actuator level.

To further locomotion development on NABi-2, it was necessary to implement state estimation in order to provide a global estimate of the state, especially the linear velocity. To this end, an open-source code for an Extended Kalman Filter was modified for use on NABi-2. The EKF utilized the IMU measurements for the prediction step and the joint encoders for the update step. This approach can be viewed as analogous to visual SLAM, where instead of visual features measured through perspective geometry, the robot has foot features which are measured with the encoders and kinematics.

The development of directional walking and jumping on NABi-2 was a difficult one as the approach used originally for achieving in-place jumping was not as effective for directional locomotion. The proprioceptive leg design of NABi-2 was meant to achieve a low body inertia to leg inertia ratio, allowing for the assumption of massless legs. However, the fact that there was a very minimal body on NABi-2 made this assumption less reasonable. A walking gait

on NABi-2 was eventually implemented by applying a set of virtual constraints achieved through the use of operational space transformations and tuning of the free parameter, the horizontal position of the hip. Directional jumping (pronking) was achieved by simply lowering the body down and spreading the legs apart, minimizing the effects of the leg dynamics on the body during the flight phase.

Systems like NABi show that event-based compositional controllers with some heuristics can be effective, but the energy efficiency of these systems are not often examined. In an attempt to better characterize the efficiency of these types of controllers, a study was done to compare a classical Raibert Controller to a trajectory optimized for energy efficiency. The study was performed for many randomly sampled simple three-link monoped designs. Jumping locomotion was achieved by using the slip-raibert approach of embedding the SLIP dynamics on the monoped, and a trajectory-optimized approach where pseudospectral collocation was used to generate an optimal jumping trajectory. The results show that the slip-raibert approach is much less efficient, but suggest the energetic cost of robustness on a jumping controller.

7.2 Takeaways

The NABi family of robots offers a unique approach to bipedal locomotion thanks to its novel leg configuration and compliance achieved through either passive springs or proprioceptive actuators. These platforms utilize simple template models that are then anchored to the physical robot platforms using a variety of different approaches to achieve locomotion. This work details many of the unique facets that arise when implementing these approaches, but

it is also interesting to examine how the performance of these systems can inform the future design and implementation of legged robots.

NABi-1 offers a glimpse at how uniquely configured passive springs can be utilized alongside classical approaches for bipedal locomotion in order to produce robust bipedal walking. In this case, the design of the physical hardware system itself was what enabled the use of the simple cart-table model for ZMP preview control based locomotion. The configuration of the leg joints and passive compliant feet elements on NABi-1, along with the reciprocating nature of the locomotion, meant that the springy feet would work naturally to stabilize walking without the need of closed-loop control around the ZMP or body. This is achieved by coordinating the cart height in the cart-table model and the step timing of the walking gait to be synchronized with the passive dynamics of the feet springs. This works because modifying the cart height in the cart-table model dictates the amplitude of the cart motion, which in turn affects the displacement of the foot spring and the time afforded to take a step. When attempting to embed the SLIP model, NABi-1 is unable to walk, even though this template explicitly incorporates passive compliance. This occurs because the direction of the forces being generated from the foot springs are not directed appropriately (towards the CoM), and the lack of sensing on NABi-1 limits its ability to control this force direction. These limitations are what ultimately led to the design of NABi-2, which is able to perform a form of explicit force control, and thus no longer requires springy feet.

NABi-2 presents a case study of how proprioceptive actuation and design on a biped can greatly enhance its dynamic capabilities, but also make it less amenable certain classic models. Directly applying the approach to walking from NABi-1 to NABi-2 was ineffective, as

NABi-2 lacks the compliant elements that account for the inevitable real-world disturbances to the desired feet position and forces. To achieve any form of locomotion via template model on NABi-2, it is necessary to perform explicit force control (using torque feedback from the proprioceptive actuators) while accounting for the internal dynamics generated by the accelerating link masses. To achieve directional pronking, the internal dynamics were reconciled by a simple shift in robot configuration, at which point the robot's physical hardware design resembled the SLIP template model (similar to how NABi-1's hardware design is amenable to the cart-table). However, walking was only achieved by cancelling the internal dynamics via projection, suggesting that the physical hardware design of NABi-2 is not intended for walking, but can be made to walk with judicious control over the the forces produced at the joints.

The results on the NABi systems convey an emphasis on a particular aspect to consider when designing a legged robot for locomotion: how the robot generates forces, both internally and externally, because how well these forces can be affected determines the scope of available motions. Of course, in the real world, forces cannot be generated from nothing, and there are always engineering trade-offs that must be considered. External forces on NABi-1 were facilitated by the use of passive compliant feet elements, and internal forces were able to be ignored thanks to the choice of actuator, meaning control was simple and with low resource cost. However, this very specific design severely constrains the motions available to the system, making it inflexible in actual application. The design of NABi-2 allows it to explicitly control its internal and external forces, meaning it is more versatile in function, but now requires additional resources (in terms of added sensors and computation) to be

controlled effectively.

As an extension, one can examine the approach taken on the types of legged robots popular today in industry. Modern dynamic quadrupeds that are designed with low-inertia legs and proprioceptive actuators are essentially attempting to minimize the production of internal forces to be able to ignore them in control, and simply focus resources on controlling external forces and performing higher level motion planning. However, this means that the body needs to have sufficient inertia to allow for this massless leg assumption, which somewhat constrains the design space of these quadrupeds. Classic humanoid designs barely consider the internal forces of the robot at all (under the assumption that they can be negated through control), allowing for a much broader design space, but complicating the control to the point where robust walking is practically very challenging.

The NABi platforms can also suggest how the design of a system interacts with the use of simple template models. Designing a system to behave like a simple model as with NABi-1 or the Raibert Hopper often produces simple and robust systems that are not versatile in application. Designing a general platform and embedding a simple model as with NABi-2 or a humanoid can produce robust and versatile systems at the cost of complexity. Arguably, the whole point of using simple models in legged locomotion is to provide a convenient abstraction to intuitively understand how a system should apply forces over time. By this interpretation, perhaps these template models are simply an unnecessary middleman when it comes to locomotion control for these more general systems. However, in reality, it is often more practical (though inefficient) to make the complex system act like the template which is known to work rather than design a controller for the complex system from scratch.

7.3 Future Work

The non-anthropomorphic bipedal concept presents a unique set of challenges that are specific to the design and implementation of the physical system. Some possible areas of future work are:

- More dynamic walking could be achieved on NABi-2 through the use of optimization in the operational space. Rather than virtually constraining the body to follow certain trajectories, it may be possible to instead formulate a quadratic program that attempts to take a step while maintaining some other objective with the body, such as minimizing momentum [130].

- Additional design features could be added to NABi-2 to make it more viable for other forms of control and locomotion. Increasing the torque output of the actuators could allow for the use of more dynamic running gaits alternate with a single leg on the ground at a time.

- The addition of a torso to the body of NABi-2 could help offset the inertia of the legs, allowing for the massless leg assumption that is used on many proprioceptive quadrupeds to be viable. This could open the possibility for MPC based approaches that model the robot as a single rigid body that is moved by forces from the ends of massless legs.

- The dual-impedance approach to walking may be viable if the operational space is used and dynamics are accounted for during walking. The operational space objective

would be a SLIP or spring-mass model for the stance leg, while the swing leg attempts to take a step. This sort of dual-SLIP type of gait has been explored in the past and has been shown to exhibit some natural stability properties.

- Turning and out of plane motion has not been explored much with NABi-2. It would be interesting to see how the hip yaw actuators could be utilized for improved lateral balancing while performing pronking, or for turning the robot out its original plane of motion.

More generally, some avenues of further exploration relating to the design of legged systems as inspired by this work include:

- Designing new leg mechanisms and configurations that can provide a varying degrees of force control authority based on the type of motion desired (i.e. high-bandwidth, variable-compliance series elastic actuator).

- Determining (quantifying) when the internal dynamics of a legged system can be ignored (i.e. massless leg assumption) and when they should be accounted for in order to better inform the hardware and computational resources necessary for a particular robot design.

- Quantifying or standardizing the approach taken to analyze whether a design is capable of controlling forces adequately for the purposes of locomotion, manipulation, or other forms of interaction with an external environment.

- Designing robots that are amenable to end-to-end approaches to locomotion. That

is, designs that can execute learned or optimized motions robustly through the use of clever design, perhaps with the addition of some feedback control.

- Analyzing the trade-offs between robustness and energy efficiency on proprioceptive platforms using simple models for control and comparing them to end-to-end approaches to inform whether these templates are useful abstractions or unnecessary intermediaries.

7.4 Closing Remarks

The author believes the NABi systems show that while the commercial legged robot industry has converged to a fairly specific set of robot designs, there is still significant room for exploration and experimentation. While the NABi family of robots are just two data points in the potentially infinite design space of legged robots, the insights on design, proprioceptive actuation, simple models, and locomotion control from them are still valuable. Legged robots, as automobiles, airplanes, and drones before them, are on the cusp of becoming a part of modern society. The author hopes that the work done here can offer some insight towards the development of these incredible machines.

References

[1] Evan Ackerman and Erico Guizzo. The next generation of boston dynamics' atlas robot is quiet robust and tether free. *IEEE Spectrum*, 24:2016, 2016.

[2] Mojtaba Ahmadi and Martin Buehler. Controlled passive dynamic running experiments with the arl-monopod ii. *IEEE Transactions on Robotics*, 22(5):974–986, 2006.

[3] Yuki Asano, Toyotaka Kozuki, Soichi Ookubo, Masaya Kawamura, Shinsuke Nakashima, Takeshi Katayama, Iori Yanokura, Toshinori Hirose, Kento Kawaharazuka, Shogo Makino, et al. Human mimetic musculoskeletal humanoid kengoro toward real world physically interactive actions. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 876–883. IEEE, 2016.

[4] Andrzej Banaszuk and John Hauser. Feedback linearization of transverse dynamics for periodic orbits. *Systems & control letters*, 26(2):95–105, 1995.

[5] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*, volume 19. Siam, 2010.

[6] Pranav A Bhounsule, Jason Cortell, and Andy Ruina. Design and control of ranger: an energy-efficient, dynamic walking robot. In *Adaptive Mobile Robotics*, pages 441–448. World Scientific, 2012.

[7] Lorenz T Biegler and Victor M Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[8] Aleksandra V Birn-Jeffery, Christian M Hubicki, Yvonne Blum, Daniel Renjewski, Jonathan W Hurst, and Monica A Daley. Don't break a leg: running birds from quail to ostrich prioritise leg safety and economy on uneven terrain. *Journal of Experimental Biology*, 217(21):3786–3796, 2014.

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

[10] 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, Submitted*, 2018.

[11] Reinhard Blickhan. The spring-mass model for running and hopping. *Journal of biomechanics*, 22(11-12):1217–1227, 1989.

[12] Reinhard Blickhan and RJ Full. Similarity in multilegged locomotion: bouncing like a monopode. *Journal of Comparative Physiology A*, 173(5):509–517, 1993.

[13] Michael Bloesch, Christian Gehring, Péter Fankhauser, Marco Hutter, Mark A Hoepflinger, and Roland Siegwart. State estimation for legged robots on unstable and slippery terrain. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6058–6064. IEEE, 2013.

[14] Michael Bloesch, Marco Hutter, Mark A Hoepflinger, Stefan Leutenegger, Christian

Gehring, C David Remy, and Roland Siegwart. State estimation for legged robots-consistent fusion of leg kinematics and imu. *Robotics*, 17:17–24, 2013.

[15] Michael Bloesch, Hannes Sommer, Tristan Laidlow, Michael Burri, Gabriel Nuetzi, Péter Fankhauser, Dario Bellicoso, Christian Gehring, Stefan Leutenegger, Marco Hutter, et al. A primer on the differential calculus of 3d orientations. *arXiv preprint arXiv:1606.05285*, 2016.

[16] I Board. Ieee standard specification format guide and test procedure for single-axis interferometric fiber optic gyros. *IEEE Std*, pages 952–1997, 1998.

[17] Maarten F Bobbert, Maurice R Yeadon, and Benno M Nigg. Mechanical analysis of the landing phase in heel-toe running. *Journal of biomechanics*, 25(3):223–234, 1992.

[18] Johann Borenstein, Hobart R Everett, Liqiang Feng, and David Wehe. Mobile robot positioning: Sensors and techniques. *Journal of robotic systems*, 14(4):231–249, 1997.

[19] Percy Williams Bridgman. *Dimensional analysis*. Yale University Press, 1922.

[20] Ben Brown and Garth Zeglin. The bow leg hopping robot. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 1, pages 781–786. IEEE, 1998.

[21] Jonas Buchli, Mrinal Kalakrishnan, Michael Mistry, Peter Pastor, and Stefan Schaal. Compliant quadruped locomotion over rough terrain. In *2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 814–820. IEEE, 2009.

[22] Christine Chevallereau, Gabriel Abba, Yannick Aoustin, Franck Plestan, Eric Westervelt, Carlos Canudas De Wit, and Jessy Grizzle. Rabbit: A testbed for advanced control theory. 2003.

[23] Avis H Cohen and Peter Wallén. The neuronal correlate of locomotion in fish. *Experimental brain research*, 41(1):11–18, 1980.

[24] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[25] John J Craig, Ping Hsu, and S Shankar Sastry. Adaptive control of mechanical manipulators. *The International Journal of Robotics Research*, 6(2):16–28, 1987.

[26] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *null*, page 1403. IEEE, 2003.

[27] Avik De and Daniel E Koditschek. Parallel composition of templates for tail-energized planar hopping. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4562–4569. IEEE, 2015.

[28] Avik De and Daniel E Koditschek. The penn jerboa: A platform for exploring parallel composition of templates. *arXiv preprint arXiv:1502.05347*, 2015.

[29] Vincent De Sapio and Oussama Khatib. Operational space control of multibody systems with explicit holonomic constraints. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2950–2956. IEEE, 2005.

[30] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2502–2509. IEEE, 2018.

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

[32] Moritz Diehl, Hans Joachim Ferreau, and Niels Haverbeke. Efficient numerical methods for nonlinear mpc and moving horizon estimation. In *Nonlinear model predictive control*, pages 391–417. Springer, 2009.

[33] Y. Ding and H. W. Park. Design and experimental implementation of a quasi-direct-drive leg for optimized jumping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 300–305, Sept 2017.

[34] Boston Dynamics. Atlas. https://www.bostondynamics.com/atlas. Accessed: 2020-01-30.

[35] Boston Dynamics. Atlas. https://www.bostondynamics.com/spot. Accessed: 2020-01-30.

[36] Tom Erez, Kendall Lowrey, Yuval Tassa, Vikash Kumar, Svetoslav Kolev, and Emanuel Todorov. An integrated system for real-time model predictive control of humanoid robots. In *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*, pages 292–299. IEEE, 2013.

[37] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014.

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

[39] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1280–1286. IEEE, 2013.

[40] Divya Garg, Michael A Patterson, Camila Francolin, Christopher L Darby, Geoffrey T Huntington, William W Hager, and Anil V Rao. Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using a radau pseudospectral method. *Computational Optimization and Applications*, 49(2):335–358, 2011.

[41] Bernd Gassmann, Franziska Zacharias, J Marius Zollner, and Rüdiger Dillmann. Localization of walking robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1471–1476. IEEE, 2005.

[42] Sepehr Ghassemi and Dennis Hong. Investigation of a novel continuously rotating knee mechanism for legged robots. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2018 15th International Conference on*, 2018.

[43] Raffaele M Ghigliazza, Richard Altendorfer, Philip Holmes, and D Koditschek. A simply stabilized running model. *SIAM review*, 47(3):519–549, 2005.

[44] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[45] Ambarish Goswami, Bernard Espiau, and Ahmed Keramane. Limit cycles in a passive compass gait biped and passivity-mimicking control laws. *Autonomous Robots*, 4(3):273–286, 1997.

[46] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.

[47] Erico Guizzo and Evan Ackerman. The rise of the robot worker. *IEEE Spectrum*, 49(10):34–41, 2012.

[48] M Haberland and S Kim. On extracting design principles from biology: 1. method–general answers to high-level design questions for bioinspired robots. *Bioinspiration & biomimetics*, 10(1):016010, 2015.

[49] M Haberland and S Kim. On extracting design principles from biology: 2. case study the effect of knee direction on bipedal robot running efficiency. *Bioinspiration & biomimetics*, 10(1):016011, 2015.

[50] Matt Haberland, JG Daniël Karssen, Sangbae Kim, and Martijn Wisse. The effect of swing leg retraction on running energy efficiency. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3957–3962. IEEE, 2011.

[51] Duncan W Haldane, Justin K Yim, and Ronald S Fearing. Repetitive extreme-acceleration (14-g) spatial jumping with salto-1p. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 3345–3351. IEEE, 2017.

[52] Charles R Hargraves and Stephen W Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of guidance, control, and dynamics*, 10(4):338–342, 1987.

[53] Ayonga Hereid and Aaron D Ames. Frost: Fast robot optimization and simulation toolkit. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 719–726. IEEE, 2017.

[54] Ayonga Hereid, Eric A Cousineau, Christian M Hubicki, and Aaron D Ames. 3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1447–1454. IEEE, 2016.

[55] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of honda humanoid robot. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1321–1326. IEEE, 1998.

[56] Joshua Hooks and Dennis Hong. Implementation of a versatile 3d zmp trajectory optimization algorithm on a multi-modal legged robotic platform. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3777–3782. IEEE, 2018.

[57] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range. *Automatica*, 47(10):2279–2285, 2011.

[58] Christian Hubicki, Jesse Grimes, Mikhail Jones, Daniel Renjewski, Alexander Spröwitz, Andy Abate, and Jonathan Hurst. Atrias: Design and validation of a tether-free 3d-capable spring-mass bipedal robot. *The International Journal of Robotics Research*, 35(12):1497–1521, 2016.

[59] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. Technical report, Robotic Systems Lab, ETH Zurich Zurich Switzerland, 2016.

[60] Marco Hutter, C David Remy, Mark A Höpflinger, and Roland Siegwart. Slip running with an articulated robotic leg. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4934–4939. IEEE, 2010.

[61] Sang-Ho Hyon and Tsutomu Mita. Development of a biologically inspired hopping robot-" kenken". In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 3984–3991. IEEE, 2002.

[62] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

[63] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of

simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer, 1995.

[64] Dirk Jansen and Holger Buttner. Real-time ethernet: the ethercat solution. *Computing and Control Engineering*, 15(1):16–21, 2004.

[65] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *ICRA*, volume 3, pages 1620–1626, 2003.

[66] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 239–246. IEEE, 2001.

[67] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[68] Kenji Kaneko, Fumio Kanehiro, Mitsuharu Morisawa, Kazuhiko Akachi, Go Miyamori, Atsushi Hayashi, and Noriyuki Kanehira. Humanoid robot hrp-4-humanoid robotics platform with lightweight and slim body. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4400–4407. IEEE, 2011.

[69] Ichiro Kato, Sadamu Ohteru, Hiroshi Kobayashi, Katsuhiko Shirai, and Akihiko Uchiyama. Information-power machine with senses and limbs. In *On Theory and Practice of Robots and Manipulators*, pages 11–24. Springer, 1974.

[70] Nicholas Kellaris, Vidyacharan Gopaluni Venkata, Garrett M Smith, Shane K Mitchell, and Christoph Keplinger. Peano-hasel actuators: Muscle-mimetic, electrohydraulic transducers that linearly contract on activation. *Science Robotics*, 3(14):eaar3276, 2018.

[71] Gavin D Kenneally, Avik De, and Daniel E Koditschek. Design principles for a family of direct-drive legged robots. *IEEE Robotics and Automation Letters*, 1(2):900–907, 2016.

[72] Sangbae Kim, Patrick M Wensing, et al. Design of dynamic legged robots. *Foundations and Trends® in Robotics*, 5(2):117–190, 2017.

[73] Tobias Klamt, Diego Rodriguez, Max Schwarz, Christian Lenz, Dmytro Pavlichenko, David Droeschel, and Sven Behnke. Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[74] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[75] Dieter Kraft. On converting optimal control problems into nonlinear programming problems. In *Computational mathematical programming*, pages 261–280. Springer, 1985.

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

[77] RA Liston and RS Mosher. A versatile walking truck. In *Transportation Engineering Conference*, 1968.

[78] Lennart Ljung. Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems. *IEEE Transactions on Automatic Control*, 24(1):36–50, 1979.

[79] Ian R Manchester. Transverse dynamics and regions of stability for nonlinear hybrid limit cycles. *IFAC Proceedings Volumes*, 44(1):6285–6290, 2011.

[80] Bryan FJ Manly. *Randomization, bootstrap and Monte Carlo methods in biology*, volume 70. CRC press, 2006.

[81] Tad McGeer et al. Passive dynamic walking. *I. J. Robotic Res.*, 9(2):62–82, 1990.

[82] Joshua S Mehling, Philip Strawser, Lyndon Bridgwater, William K Verdeyen, and Roger Rovekamp. Centaur: Nasa's mobile humanoid designed for field work. In *Proceedings 2007 IEEE international conference on robotics and automation*, pages 2928–2933. IEEE, 2007.

[83] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163,

2015.

[84] Simona Nobili, Marco Camurri, Victor Barasuol, Michele Focchi, Darwin Caldwell, Claudio Semini, and MF Fallon. Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots. 2017.

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

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

[87] Jaeheung Park and Oussama Khatib. Contact consistent control framework for humanoid robots. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1963–1969. IEEE, 2006.

[88] Jong H Park and Kyong D Kim. Biped robot walking using gravity-compensated inverted pendulum mode and computed torque control. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 4, pages 3528–3533. IEEE, 1998.

[89] Michael A Patterson and Anil V Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation

methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[90] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

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

[92] Ioannis Poulakakis and Jessy 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, 2009.

[93] Matthew J Powell, Huihua Zhao, and Aaron D Ames. Motion primitives for human-inspired bipedal robotic locomotion: walking and stair climbing. In *2012 IEEE International Conference on Robotics and Automation*, pages 543–549. IEEE, 2012.

[94] Gill A Pratt and Matthew M Williamson. Series elastic actuators. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 399–406. IEEE, 1995.

[95] Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001.

[96] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. Bigdog, the

rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2):10822–10825, 2008.

[97] Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

[98] Alireza Ramezani, Jonathan W Hurst, Kaveh Akbari Hamed, and Jessy W Grizzle. Performance analysis and feedback control of atrias, a three-dimensional bipedal robot. *Journal of Dynamic Systems, Measurement, and Control*, 136(2):021012, 2014.

[99] Jaoa Ramos, Benjamin Katz, Meng Yee Chuah, and Sangbae Kim. Facilitating model-based control through software-hardware co-design. In *Robotics and automation (ICRA), 2018 IEEE international conference on*, page in press. IEEE, 2018.

[100] Joao Ramos and Sangbae Kim. Improving humanoid posture teleoperation by dynamic synchronization through operator motion anticipation. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5350–5356. IEEE, 2017.

[101] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2016.

[102] Jacob Reher, Eric A Cousineau, Ayonga Hereid, Christian M Hubicki, and Aaron D Ames. Realizing dynamic and efficient bipedal locomotion on the humanoid robot durus. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1794–1801. IEEE, 2016.

[103] Agility Robotics. Cassie. https://www.agilityrobotics.com/robots#cassie. Accessed: 2020-01-30.

[104] Agility Robotics. Digit. https://www.agilityrobotics.com/robots#digit. Accessed: 2020-01-30.

[105] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013.

[106] Gerald P Roston and Eric P Krotkov. Dead reckoning navigation for walking robots. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1991.

[107] Lewis A. Rygg. Mechanical horse, 1893. Patent No. US491927A, Filed Apr. 1892, Issued Feb. 14th, 1893.

[108] Yoshiaki Sakagami, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo Fujimura. The intelligent asimo: System overview and integration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2478–2483. IEEE, 2002.

[109] Uluc Saranli, Martin Buehler, and Daniel E Koditschek. Rhex: A simple and highly mobile hexapod robot. *The International Journal of Robotics Research*, 20(7):616–631, 2001.

[110] Justin E Seipel and Philip Holmes. Running in three dimensions: Analysis of a point-mass sprung-leg model. *The International Journal of Robotics Research*, 24(8):657–674, 2005.

[111] Claudio Semini, Nikos G Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. Design of hyq–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011.

[112] S. Seok, A. Wang, D. Otten, and S. Kim. Actuator design for high force proprioceptive control in fast legged locomotion. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1970–1975, Oct 2012.

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

[114] Sangok Seok, Albert Wang, Meng Yee Michael Chuah, Dong Jin Hyun, Jongwoo Lee, David M Otten, Jeffrey H Lang, and Sangbae Kim. Design principles for energy-efficient legged locomotion and implementation on the mit cheetah robot. *IEEE/ASME Transactions on Mechatronics*, 20(3):1117–1129, 2015.

[115] Sangok Seok, Albert Wang, David Otten, and Sangbae Kim. Actuator design for high force proprioceptive control in fast legged locomotion. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1970–1975. IEEE, 2012.

[116] Andre Seyfarth, Hartmut Geyer, Michael Günther, and Reinhard Blickhan. A movement criterion for running. *Journal of biomechanics*, 35(5):649–655, 2002.

[117] Joan Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.

[118] Mark W Spong. Partial feedback linearization of underactuated mechanical systems. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, volume 1, pages 314–321. IEEE, 1994.

[119] Mark W Spong. The swing up control problem for the acrobot. *IEEE control systems magazine*, 15(1):49–55, 1995.

[120] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and Jessy W Grizzle. A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel. *The International Journal of Robotics Research*, 30(9):1170–1193, 2011.

[121] Kentaro Takahashi, M Noda, Dragomir N Nenchev, Yuichi Tsumaki, and Akinori Sekiguchi. Static walk of a humanoid robot based on the singularity-consistent method. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5484–5489. IEEE, 2006.

[122] Russ Tedrake. Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for mit 6.832. *Working draft edition*, page 3, 2009.

[123] Russ Tedrake, Scott Kuindersma, Robin Deits, and Kanako Miura. A closed-form solution for real-time zmp gait generation and feedback stabilization. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 936–940. IEEE, 2015.

[124] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[125] Emanuel Todorov. A convex, smooth and invertible contact model for trajectory optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 1071–1076. IEEE, 2011.

[126] Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric A Wan. The unscented particle filter. In *Advances in neural information processing systems*, pages 584–590, 2001.

[127] Oskar Von Stryk and Roland Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of operations research*, 37(1):357–373, 1992.

[128] Miomir Vukobratović and Branislav Borovac. Zero-moment pointthirty five years of its life. *International journal of humanoid robotics*, 1(01):157–173, 2004.

[129] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

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

[131] Patrick M Wensing and David E Orin. High-speed humanoid running through control

with a 3d-slip model. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5134–5140. IEEE, 2013.

[132] ER Westervelt, JW Grizzle, and DE Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, 2003.

[133] Alexander W Winkler, C Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018.

[134] Jeffrey Yu, Joshua Hooks, Sepehr Ghassemi, and Dennis Hong. Exploration of turning strategies for an unconventional non-anthropomorphic bipedal robot. In *ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V05BT08A021–V05BT08A021. American Society of Mechanical Engineers, 2017.

[135] Jeffrey Yu, Joshua Hooks, Sepehr Ghassemi, Alexandra Pogue, and Dennis Hong. Investigation of a non-anthropomorphic bipedal robot with stability, agility, and simplicity. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2016 13th International Conference on*, pages 11–15. IEEE, 2016.

[136] Jeffrey Yu, Joshua Hooks, Xiaoguang Zhang, Min Sung Ahn, and Dennis Hong. A proprioceptive, force-controlled, non-anthropomorphic biped for dynamic locomotion. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE, 2018.

[137] Taoyuanmin Zhu, Joshua Hooks, and Dennis Hong. Design, modeling, and analysis of a liquid cooled proprioceptive actuator for legged robots. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 36–43. IEEE, 2019.