OPTIMIZATION AND CONTROL OF DYNAMIC HUMANOID RUNNING AND JUMPING

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy
at The Ohio State University

By

Patrick M. Wensing

* * * * *

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2014

Dissertation Committee:

David E. Orin, Advisor
Yuan F. Zheng
Andrea Serrani
Abstract

Animals in nature display a nearly seamless capability to navigate the world around them. Whether running up a steep mountain trail, weaving through a dense forest, or jumping to clear obstacles, legged animals are capable to dynamically negotiate challenging terrains with grace and efficiency. The development of legged machines with even a portion of this legged mobility would provide great benefit to applications in defense, search and rescue, and planetary exploration. The objective of this dissertation is to make a significant contribution towards the dynamic capabilities of legged machines, more specifically as applied to humanoid robots.

Humanoid robots represent one potential platform to study legged mobility. Compared to other legged morphologies, humanoids have increased potential for application in human-inhabited environments due to their structural similarity to humans. Despite the surge of work on humanoid robotics in the recent decade, current machines are not yet capable of any significant dynamic mobility. The development of control systems for dynamic humanoids is a difficult task given their high number of degrees of freedom, which require continuous coordination, as well as their complex nonlinear dynamics, which change fundamentally in the presence of contacts with the ground. Performance of dynamic movements is further complicated by frequent periods of static instability, requiring continuous motion to prevent a catastrophic fall. Even for a basic dynamic movement, such as a high-speed run, current approaches
for humanoids are unable to generalize to a range of speeds or turning rates, or have not demonstrated robustness to disturbances.

With the control approach described in this dissertation, a standing long jump, high-speed run, running turn, and running long jump are demonstrated in 3D dynamic simulation with a 26 degree of freedom (DoF) humanoid model. By focusing on the design and control of the salient features of the dynamic movements, the system is capable to run at speeds of up to 6.5 m/s, which is comparable to the speed of an Olympian in the 5000m race. Advances in whole-body humanoid task-space control are presented, where the control of centroidal momentum is shown to be an enabling approach for dynamic balance control. This approach is shown to result in emergent upper-body motions to maintain balance in a number of examples. New relationships between the dynamics of centroidal momentum and the dynamics of the humanoid in joint-space are highlighted, which will simplify future application of this emerging approach. Algorithms to compute the task-space inertia matrix and a formulation of the task-space control problem as a conic optimization problem, presented here, provide computational benefits to applications of task-space control. Further, these advances enable all the control examples for the 26 DoF humanoid model used here to be performed at real-time rates.

A high-level running controller based on a 3D-SLIP model is presented to interface with this whole-body controller, resulting in automatic footstep planning to maintain balance across a range of speeds in the face of disturbances. The controller is also shown to be general to produce high-speed running turns. For instance, when running at 3.5 m/s the humanoid is capable to execute a tight turn with a radius that is approximately 1/4 that of a 400m track. Through extensions of the 3D-SLIP
model, this framework is also shown to be general to produce a running long jump, an aperiodic movement which includes significant underactuated periods of flight. The trajectory optimization approach for this model is shown to result in long-jump strategies that match those employed in human long jumpers, and highlights the importance of takeoff-velocity angle. Rather than employing the ballistic optimum of a 45° takeoff-velocity angle, the approach matches biological findings which use a more shallow angle to maximize jump span.
To my family.
Acknowledgments

First and foremost, I’d like to thank my advisor, Dr. David Orin, for all of his support, guidance, and encouragement over these past five years. His personal investment in my academic success and overall well being has been an appreciated constant amidst the many ups and downs of PhD research. I will miss thinking together in our meetings and his helpful push for me to crystallize my ideas at a broader level.

I’d also like to say thank you to Dr. Jim Schmiedeler at The University of Notre Dame for his support of my academic journey. Whether through research feedback in our weekly group video conferences, or his encouraging career advice, his perspectives have always helped me to feel more confident with my path forward.

I’m grateful to have two professors on my committee that I much respect. Dr. Yuan F. Zheng has always stood behind my ambitions and helped me to attend conferences early in my graduate school career that greatly shaped my outlook on the field. Courses with Dr. Andrea Serrani put a spark in my first years of graduate school and provided a model for me on how to bring excitement to the classroom.

I couldn’t have asked for more pleasant company in the ECE support staff. Stephanie Muldrow, Tricia Toothman, and LuCinda Flores always provided a bright side to the necessary paperwork of the PhD process. Thank you.

In my time at Ohio State I’ve been lucky enough to become friends with some of the most brilliant, kind, and fun-to-be-around people that the engineering world
has to offer. David Hawn, Vienny Nguyen, Luke Duncan, Mike Vernier, Joe Plattenburg, Mark Scott, and Brian Day have been great friends to talk with about research and academic life over the years. I'd also like to thank my lab mates Ghassan Bin Hammam and Yiping Liu for providing friendships beyond our many chalkboard discussions. I’m grateful for my virtual lab mate Travis Brown at Notre Dame for always helping to provide a fresh perspective to my work. I’m also thankful for my colleagues in first-year engineering. Dr. Rick Freuler, Paul Clingan, Krista Kecskemety, and my GTA cohort provided a supportive engineering family in my time at Ohio State.

Most importantly, I’d like to thank all my family for their love and support. My caring parents, Pam and Dan, and loving wife, Kelly, have provided constant encouragement and understanding throughout this process. I’m blessed to have them in my life.

Finally, I’d like to acknowledge financial support from The Ohio State University, the Ohio Space Grant Consortium, and the National Science Foundation (Grant No. DGE-1343012) through graduate fellowships, as well as Grant No. CNS-0960061 from the NSF with a subaward to The Ohio State University.
Vita

April 9, 1987 .......................... Born - Euclid, Ohio, USA

June 2009 ............................... B.S. Elec. and Comp. Engineering,
The Ohio State University,
Columbus, Ohio
summa cum laude, with distinction

June 2009 - May 2010 .................. University Fellow,
The Ohio State University

June 2010 - August 2011 ............... Ohio Space Grant Consortium Fellow,
The Ohio State University

September 2011 - August 2014 .......... National Science Foundation Graduate
Research Fellow,
The Ohio State University

September 2012 - June 2014 ............ Graduate Teaching Associate,
The Ohio State University
Columbus, Ohio

December 2013 .......................... M.S. Elec. and Comp. Engineering,
The Ohio State University,
Columbus, Ohio
Publications


Fields of Study

Major Field: Department of Electrical and Computer Engineering

Studies in:

Robotics  Professors D. E. Orin, Y. F. Zheng
Optimization  Professors K. M. Passino, S. Kucukyavuz,
              R. Sioshansi, M. Srinivasan
Control Systems  Professors A. Serrani, V. I. Utkin
Machine Learning  Professors D. Wang, M. Belkin
Dynamics  Professors D. E. Orin, A. Kahraman
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>v</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>vi</td>
</tr>
<tr>
<td>Vita</td>
<td>viii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xvi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xviii</td>
</tr>
<tr>
<td><strong>Chapters</strong></td>
<td></td>
</tr>
<tr>
<td>1.   Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Previous Work</td>
<td>6</td>
</tr>
<tr>
<td>1.2.1 Inverse Dynamics Approaches</td>
<td>6</td>
</tr>
<tr>
<td>1.2.2 Trajectory Optimization for Locomotion</td>
<td>7</td>
</tr>
<tr>
<td>1.2.3 Task-Space Control</td>
<td>9</td>
</tr>
<tr>
<td>1.2.4 Heuristic and Intelligent Controllers</td>
<td>10</td>
</tr>
<tr>
<td>1.2.5 Simple-Model-Based Control Approaches</td>
<td>12</td>
</tr>
<tr>
<td>1.3 Organization and Contributions</td>
<td>13</td>
</tr>
<tr>
<td>2.   Humanoid Dynamics Model</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Humanoid Model</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1 Simulation</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.3</td>
<td>Conventions and Notation for Rigid-Body Dynamics</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>Whole-Body Dynamics</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>Summary</td>
</tr>
<tr>
<td>3.</td>
<td>A Reduced-Order Recursive Algorithm for the Computation of the</td>
</tr>
<tr>
<td></td>
<td>Operational-Space Inertia Matrix</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>A Real-Time Conic Formulation of Prioritized Task-Space Control</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>High-Speed Humanoid Running Through Control with a 3D-SLIP Model</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Numeric values of the main kinematic and dynamic parameters of the humanoid model.</td>
<td>19</td>
</tr>
<tr>
<td>2.2 Spring-damper contact model parameters.</td>
<td>20</td>
</tr>
<tr>
<td>2.3 Free modes of motion for joint types used in this work.</td>
<td>24</td>
</tr>
<tr>
<td>3.1 Extended-Force-Propagator Algorithm</td>
<td>48</td>
</tr>
<tr>
<td>4.1 Prioritized Task-Space Control (PTSC) Algorithm</td>
<td>68</td>
</tr>
<tr>
<td>5.1 Weight and gain settings for the PTSC. Where omitted, all derivative gains are set for critical damping ($K_D = 2\sqrt{K_P}$).</td>
<td>101</td>
</tr>
<tr>
<td>6.1 Kinematic bounds placed on the humanoid virtual leg angles for 3D-SLIP optimization.</td>
<td>118</td>
</tr>
<tr>
<td>7.1 Fixed parameter values for Long Jump 3D-SLIP dynamics.</td>
<td>140</td>
</tr>
<tr>
<td>7.2 Equations of motion for the Long Jump 3D-SLIP model.</td>
<td>143</td>
</tr>
<tr>
<td>7.3 Thrust step optimization summary</td>
<td>145</td>
</tr>
<tr>
<td>7.4 Optimized Long Jump 3D-SLIP parameters for a 6 m/s approach</td>
<td>148</td>
</tr>
<tr>
<td>A.1 Dynamic parameters for the torso body.</td>
<td>182</td>
</tr>
<tr>
<td>A.2 Dynamic and kinematic parameters for the left thigh body.</td>
<td>182</td>
</tr>
<tr>
<td>A.3 Dynamic and kinematic parameters for the left shank body.</td>
<td>183</td>
</tr>
</tbody>
</table>
A.4 Dynamic and kinematic parameters for the left foot body and kinematic parameters for its intermediate ankle body. . . . . . . . . . . . 184

A.5 Dynamic and kinematic parameters for the left upper arm body. . . . 185

A.6 Dynamic and kinematic parameters for the left forearm body. . . . . 185

B.1 Inward recursion of the RJK Algorithm . . . . . . . . . . . . . . . . . 187

B.2 Outward recursion of the RJK Algorithm . . . . . . . . . . . . . . . . 188

B.3 Notational correspondence between [31, Table 4] and the RJKA here. 189

B.4 Computational costs for spatial operations within the EFPA and RJKA.190
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Extreme terrain negotiation by species of goats.</td>
<td>2</td>
</tr>
<tr>
<td>1.2 The HUBO humanoid [16] developed by KAIST and the ASIMO humanoid [128] developed by Honda.</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Graphical humanoid model in the DynaMechs simulation environment.</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Definitions for main dimensions of the humanoid model.</td>
<td>18</td>
</tr>
<tr>
<td>2.3 Rigid-body model of the humanoid and its connectivity tree.</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Inward recursion of the EFPA.</td>
<td>41</td>
</tr>
<tr>
<td>3.2 Simplification to the outward recursion of the EFPA that is provided by the acceleration propagator $k_1x_1$.</td>
<td>46</td>
</tr>
<tr>
<td>3.3 Connectivity tree of the ASIMO Next-Generation robot.</td>
<td>51</td>
</tr>
<tr>
<td>3.4 Cost breakdown for three operational-space algorithms.</td>
<td>52</td>
</tr>
<tr>
<td>3.5 Cost comparison for operational-space algorithms.</td>
<td>52</td>
</tr>
<tr>
<td>4.1 Behaviors demonstrated through the use of Prioritized Task-Space Control (PTSC).</td>
<td>57</td>
</tr>
<tr>
<td>4.2 Block diagram for the application of Prioritized Task-Space Control (PTSC).</td>
<td>58</td>
</tr>
<tr>
<td>4.3 Force distribution of the spatial force on each foot to linear forces at its contact vertices.</td>
<td>64</td>
</tr>
</tbody>
</table>
4.4 State machine used for Kick control. ................................. 70
4.5 Shoulder and torso angles during the Kick state of the kick motion. 74
4.6 Control computation times for PTSC applied to a kick motion with a QP versus a conic formulation. ................................. 75
4.7 State machine used for jump control. ................................. 76
4.8 Control computation times for PTSC applied to a jump motion with a QP versus a conic solver. ................................. 77
4.9 Shoulder and torso angles during the jump motion. ............... 78
5.1 High-speed humanoid running in simulation. ......................... 82
5.2 Block diagram of the humanoid running control system. ........... 85
5.3 3D-SLIP template model for high-speed humanoid running. ....... 88
5.4 Touchdown angle definitions for running. ........................... 89
5.5 State machine used for running control. ............................ 97
5.6 Right foot trajectory relative to the CoM for running at 3.5 m/s. 98
5.7 CoM velocity tracking for running at a desired speed of 3.5 m/s. 102
5.8 Yaw angular momentum about the CoM as contributed by the legs and the upper body. ................................................. 103
5.9 Tracking of a commanded forward velocity profile. ............... 104
5.10 Footstep selection for lateral disturbance rejection. ............... 106
6.1 Humanoid turning in simulation. ....................................... 108
6.2 CoM trajectories for 3D-SLIP steering with monopod and biped turning. 109
6.3 Block diagram for the humanoid control system with 3D-SLIP CoM steering. .................................................. 111

6.4 3D-SLIP model with auxiliary coordinate system to model turning. .................................................. 113

6.5 Contrast between turn angle imposed on velocity and the resulting sweep angle about a circle. .................................................. 121

6.6 Evolution of ToF positions after each gait cycle through continuous turning. .................................................. 122

6.7 Foot positioning used by the humanoid for a running turn at 3.5 m/s with a turning rate of $\omega=0.4$ rad/s. .................................................. 127

6.8 Force profiles for 3.5 m/s running with the 3D-SLIP template. .................................................. 128

6.9 Force profiles for 3.5 m/s running and a 0.4 rad/s turn rate to the left. .................................................. 129

6.10 Foot positioning comparison for different turn rates and running speeds. .................................................. 130

6.11 Torso roll during a left turn to right turn transition. .................................................. 131

7.1 High-speed running long jump in 3D dynamic simulation .................................................. 134

7.2 Experimentally observed vertical ground reaction forces and leg stiffness for a human long jump. .................................................. 136

7.3 Extended 3D-SLIP models for a running long jump. .................................................. 137

7.4 Breakdown of the span of a long jump into three components. .................................................. 144

7.5 Simple model trajectories in the sagittal plane for an optimized long jump following a 6 m/s run. .................................................. 147

7.6 CoM velocity for an optimized long jump following a 6 m/s run. .................................................. 149

7.7 Active constraints on the optimum long jump. .................................................. 150

7.8 Effect of takeoff-velocity angle on jump span. .................................................. 151
7.9 Maximal takeoff-velocity versus takeoff-velocity angle for a 6 m/s and 4 m/s running approach. ........................................... 153

A.1 Humanoid coordinate system definitions. .......................... 180
Chapter 1

Introduction

1.1 Motivation

Animals in nature provide many amazing examples of the advantages that legs can offer. Whether climbing over piles of rubble, running up a steep mountain trail, weaving through a dense forest, or jumping to clear an obstacle, legged animals display nearly seamless capacity to negotiate challenging terrains in the world around them. In contrast to wheeled or tracked vehicles, legged animals and machines require only sparse footholds to traverse their surrounding environment. In nature, this added mobility provides extreme capabilities well beyond those of current legged machines. For instance, various species of goats can take advantage of sparse footholds to scale steep cliffs and even climb trees as shown in Figure 1.1. Providing legged machines with even a portion of these capabilities would have great application in search and rescue, planetary exploration, and defense.

The creation of machines with legged mobility comparable to animals has been a common vision for robotics research, starting as early as the 1960’s with the pioneering work of McGhee [82]. Yet, even today, the field of legged locomotion faces challenges on a number of fronts to realize this original vision. These challenges fall broadly
into the categories of environmental sensing and perception, high-power compliant actuation, high-level planning and reasoning, and whole-body control. While each of these areas remains an active and important area of robotics research, this dissertation focuses broadly on whole-body control of legged locomotion, more specifically as applied to humanoid robots.

In the past decade, humanoid robots have become a focal point for robotics research [7, 17, 19, 39, 42, 56, 68, 69, 86, 93, 97, 101, 118, 121, 127]. Much of this focus is motivated by technological advances that have begun to enable humanoids with locomotion and manipulation capabilities that are comparable to human beings. These efforts are further motivated by their structural similarity to humans, which provides humanoids with distinct advantages over other morphologies in terms of their potential for natural human-to-robot interaction, or their capacity to operate in spaces
designed for human occupancy. As a prime example, humanoids have promising applicability to assist in dangerous, human-engineered environments for disaster recovery operations such as those presented in the current DARPA Robotics Challenge [1].

The versatility of humanoid robots can be observed in the breadth of research topics they have inspired. For instance, recent efforts have included motion planning for grasping and manipulation [42, 65], image processing and compliant actuation for the performance of sensitive household tasks [6, 41], and human-to-humanoid motion transfer for entertainment through the performance of interpretive dance [93].

While humanoids have the potential to operate alongside or in place of humans in a variety of capacities, they largely have yet to make their way out of the lab and into the home or workplace. Much of this failing can be attributed to the challenges imposed by their high degree-of-freedom bipedal structure. Although their legs provide advantages in terms of mobility and agility, their many degrees of freedom are difficult to coordinate in real time and their limited footprint provides significant challenges to maintain balance. Due to the friction-limited, intermittent, unilateral contacts of the feet with the ground, humanoid robots can experience periods of underactuation which may lead to a fall. Moreover, they exhibit complex nonlinear dynamics which change fundamentally based on the presence and number of contacts. Contact force limitations place bounds on the net external wrench (force and moment) that can be exerted on the humanoid, thus limiting the achievable system accelerations. While this limitation does not seriously hinder the control of quasi-static locomotion, it adds considerable difficulty to the performance of high-speed dynamic movements which have not been well studied in the literature.
A large class of modern humanoids, such as the HUBO and ASIMO humanoids shown in Figure 1.2, partially address these limitations through the design of trajectories that satisfy the Zero Moment Point (ZMP) constraint [135]. Humanoids operating in the ZMP domain aim to command their overall Center of Pressure (CoP) to be within the convex hull of their supporting feet on the ground plane of contact. When this condition is satisfied, the humanoid is effectively fully actuated, simplifying control. While the approach is not general to uneven terrain and neglects frictional bounds on the ground reaction forces, the use of the ZMP constraint remains ubiquitous in the humanoid locomotion literature. In addition, since ZMP-based trajectory generation methods enforce the ZMP to be strictly within the convex hull of support, the methods do not naturally extend to produce dynamic running motions with extended periods of flight as studied here.
Despite significant efforts in the area of humanoid locomotion, only a small subset of this research has focused on dynamic locomotion. In relation, humanoid walking has received far greater attention [15, 20, 35, 54, 55, 90, 129, 147, 156], at least partially due to the usefulness of the ZMP constraint in this domain. During dynamic locomotion, balance is no longer an instantaneous notion. Instead dynamic movements rely on future motion to prevent a catastrophic fall. For instance, in running, during each step the system is essentially falling forward, and requires the subsequent step to maintain dynamic balance. As a result of these complexities, the state-of-the-art in the control of dynamic humanoid balance is still insufficient if humanoids are to become useful coworkers or companions. To enable such applications, humanoids must be endowed with the knowledge of how to perform dynamic, whole-body motions in order to recover from disturbances caused by terrain irregularities or unplanned contact from the environment.

As the main contribution, this dissertation departs from the common ZMP approach, and presents an optimization-based gait generation methodology that is focused on a powerful 3D Spring-Loaded Inverted Pendulum (SLIP) template model. Within this new framework, the humanoid is capable to run at Olympic speeds, perform high-speed turns, and run into a jump across a large gap. Through the first application of the 3D-SLIP model to humanoids, the system is able to automatically plan footstep locations online and is robust to push disturbances. Previous approaches for humanoid running have not been general to a range of speeds of have not shown robustness to disturbances. In addition, the high-speed movements demonstrated here are able to be controlled at real-time rates, which represents another unique characteristic of this work in comparison to many previous results.
1.2 Previous Work

The area of dynamic locomotion has been studied not just by humanoids researchers, but also in part by researchers in the areas of graphics and bipedal/multi-legged robotics. While inevitably any attempt at categorization will result in overlap, a rough segmentation of the previous work in these areas is provided in Sections 1.2.1 - 1.2.5. Methods in the first two categories, Inverse Dynamics and Trajectory Optimization, focus on the design and control of dynamic movements at the joint-space level, resulting in an incredible amount of required motion detail. Methods in the second two categories, Task-Space Control and Heuristic/Intelligent Controllers, focus on the control of aggregate features of a motion. Approaches in the final category, Simple-Model-Based Control, determine or alter motion details at runtime through the use of many different prediction models. The approaches used here largely fall into the Simple-Model and Task-Space Control categories, with drawbacks of the other approaches described.

1.2.1 Inverse Dynamics Approaches

Inverse dynamics approaches for humanoid dynamic movement require the design of whole-body joint trajectories offline, prior to the performance of the maneuver by the humanoid. Due to the many degrees of freedom (DoF) present in humanoids, it is difficult to author a full set of joint trajectories by hand. This difficulty has led to the widespread use of human motion capture for the generation of whole-body humanoid motions [2, 17, 77, 154]. Once a desired motion is generated, inverse dynamics controllers determine joint torques for the robot at runtime to realize the motion. While an exact solution to the inverse dynamics problem is easily found for
manipulators [76], the potential for underactuation in humanoids has lead to a variety of solutions to this problem [86, 154].

Humanoid inverse dynamics approaches are separated by how (or if) the original motion is modified to promote balance. For instance, Mistry et al. [86] assume sufficient friction on the feet and use a projection method to select appropriate torques and resolve contact force redundancy in double support. Improvements to their approach [109] attempt to push ground reaction forces inside their frictional boundaries. Still, no motion modification is applied, which can lead to balance failure when aggressive joint accelerations are commanded. Other approaches by Yamane and Nakamura [154] and Bin Hammam et al. [7] modify, or “filter”, the commanded motion to remained balanced. Park et al. [102] formulate a stringent motion filter as a conic optimization problem which enforces the ZMP constraint. These motion filtering ideas have been applied to movements in stance [77] or to periodic locomotion [2].

The main drawback of this technique is that the reliance on pre-designed trajectories makes it difficult to modify the gross characteristics of a motion online. That is, the resultant motion of all the above approaches, even when modification is applied, still fundamentally follows the pre-designed input motion. Since motion capture data is required for each motion, it would be costly to design a rich and adaptable library of dynamic motions with these techniques.

1.2.2 Trajectory Optimization for Locomotion

Trajectory optimization approaches offer an alternative method to the design of whole-body trajectories. Rather than relying on hand design or motion capture, these approaches use tools from optimization to semi-automatically generate a dynamically
feasible motion. In the graphics community, these methods are often referred to as “space-time” optimization, as originally introduced by Witkin and Kass [149]. In the space-time optimization framework, joint trajectories are first parameterized with B-Splines [24, 70] or through waypoints [11, 21, 138]. Objective functions to evaluate these trajectories are then designed to penalize violations in the laws of physics and to reward proper performance of the desired behavior, such as in the dynamic run demonstrated by Erez and Todorov [21] or a flip from a high-bar demonstrated by Fang and Pollard [24]. Some approaches use motion capture to provide a basis for the desired motion [115], while others use a high-level planner to provide initial seeds to the optimization [19]. These approaches take minutes to hours to compute and have yet to demonstrate viability for near real-time use.

Rather than parameterize the joint trajectories for a motion, other methods in this area parameterize a class of control policies and then use dynamic simulation to evaluate the performance of a policy. These methods are often referred to as policy search methods. Krasny and Orin [62, 63] used genetic algorithms to tune the control parameters for a gallop, running turn, and jump in a simulated quadruped. Srinivasan [126] used sequential quadratic programming (SQP) methods with a simple legged model to discover running and walking as optimized behaviors for traveling at required speeds. Mombaur [88] applied multiple shooting methods to generate an open-loop stable periodic run. The orbital stability of the resultant gait was provided through direct optimization of the eigenvalues of the resultant Poincaré return map. Once again, these methods have yet to demonstrate viability for near real-time use. Closely related are policy gradient methods from reinforcement learning that have been applied successfully to generate dynamic walking and jumping in physical robots.
However, reinforcement learning algorithms have not yet shown general applicability to high-DoF systems, and the local nature of these search methods makes them sensitive to proper initial estimates.

While powerful for offline computation, the difficulty to optimize whole-body trajectory or control details is too computationally burdensome to be viable for online planning. The work proposed here draws inspiration from task-space and simple-model-based approaches in order to alleviate these difficulties. Due to the complexity of planning in the high-dimensional continuous action space of the humanoid, these approaches simplify the control by focusing on important characteristics of locomotion instead of the exact details of whole-body trajectories.

1.2.3 Task-Space Control

Task-space, or operational-space, control provides a framework that significantly eases the burden of authoring whole-body behaviors. Task-space trajectories for a given behavior can be designed in intuitive feature spaces instead of in the entire joint space. For instance, walking motions can be specified through design of foot and center of mass trajectories [18], which significantly simplifies motion design. While this approach to control dynamic movements has been applied quite sparsely in the literature, some have used it to perform a dynamic standing jump [101], broad jump [122], low-speed jog [89], and continuous hopping [145]. Task-space control is proposed as a fundamental low-level control component in this work and is developed more fully in Chapter 4. Since task-space controllers focus on regulating the instantaneous system dynamics, and do not consider the extended time horizon implications of
control decisions, higher-level controllers with extended temporal insight are need with task-space control for successful locomotion, as is applied in Chapter 5.

Within the context of task-space control for dynamic balancing, centroidal momentum [95, 97] has recently emerged as an important feature of balanced motion. The centroidal momentum is comprised of the net linear momentum of the system along with its net angular momentum about the Center of Mass (CoM). While the linear momentum has a well-known connection to the evolution of the CoM, the centroidal angular momentum has only recently been shown to be a regulated feature in human walking [49]. The application of task-space control to regulate the centroidal angular momentum is an emerging approach with its first application to high-speed running here.

1.2.4 Heuristic and Intelligent Controllers

Rather than focusing on the trajectory details of a motion (in joint or a reduced-dimensional task space), heuristic and intelligent control techniques instead attempt to control the main features of a motion through an understanding of the salient system-level dynamics. Perhaps the best example of this approach is the celebrated work of Raibert [108] and colleagues starting in the mid 1980’s. Raibert identified key foot placement and leg thrust heuristics to provide a three-fold decomposition of control for hopping height, speed, and posture. These and other related heuristics were first applied to monopods [108], but later extended to planar bipeds by Hodgins and Raibert [46], and by Playter and Raibert [105] to produce a fully-3D biped capable of dynamic flips. Despite employing design choices of light prismatic legs and collocated hips to simplify control, the capabilities of Raibert’s biped remain the
standard to which other experimental legged machines are often compared. Hodgins et al. [47] further applied heuristic control ideas to produce simulated athletic human running and vaulting motions as well as animated jumps and flips [150]. Yin et al. [155] used simple foot placement heuristics to demonstrate omni-directional walking and disturbance recovery for a human figure in simulation. Perkins et al. [103, 104] developed heuristics for acceleration, deceleration, and turning for a biped in simulation but only demonstrated their applicability in a small set of experiments without disturbances. In general it is difficult to use heuristic methods to achieve a wide range of desired performance, since the fine tuning of how a heuristic is applied is non-trivial.

Intelligent control techniques have been used to extend the usefulness of heuristics in many applications. Control systems for quadruped bounding [80], quadruped trotting [100], biped jumping [44, 73], and monopod hopping [99] have all applied heuristics to create and/or adapt fuzzy rule bases. With these approaches, heuristics can be applied for control adaptation (policy improvement) instead of directly in the control policy. As a result, the fine tuning of their application is less important. A drawback of any heuristic-based approach is that the identification of proper heuristics requires deep intuition into the fundamental physics of the problem, which often takes much time for the control designer. In addition, when intelligent control techniques are applied, the knowledge base quickly grows impractically large and appropriate tuning heuristics become increasingly difficult to determine when the controller needs to respond to a wide range of situations or employs an increased number of controls.
1.2.5 Simple-Model-Based Control Approaches

Simple-model-based control approaches to humanoid balance attempt to control the salient characteristics of a motion, similar to heuristic controllers, but do so through principled methods that require less manual tuning. The Linear-Inverted Pendulum (LIP) model is the most commonly used simple model in the robotics community. This model captures the relationship between the ZMP and the evolution of the CoM. Kajita et al. [54] use this simple model to develop the common ZMP preview controller for humanoid walking.\footnote{Wieber [147] provides an insightful discussion of ZMP preview control within the framework of Model Predictive Control (MPC).} Yamane and Hodgins [152, 153] applied this model within the context of motion filtering to improve the balancing capabilities of inverse-dynamics based motion control. Stephens and Atkeson [127] used this template model for push recovery in the case of standing balance.

The LIP model has also been augmented to enable application to other movements. It was combined with a vertical spring-mass model to produce low-speed humanoid jogging [89]. The point-mass LIP has been augmented with inertia characteristics to model upper-body mass and aid in the selection of footsteps for push recovery [61, 107]. None of these models however have been applied to control high-speed running.

The Spring-Loaded Inverted Pendulum (SLIP) model has long been recognized in the biomechanics community as an appropriate low-order model to describe the CoM dynamics in running across a wide range of species [9]. As opposed to the LIP model, which remains in contact throughout a motion, this model captures phases of stance
and flight that are present during a high-speed run. This model is applied for the first time in humanoids here.

1.3 Organization and Contributions

This dissertation contributes broadly to the development of dynamic legged locomotion, with more specific contributions relevant to task-space and simple-model-based control for humanoid robots. Chapter 2 describes the humanoid model used in much of this dissertation and provides a background on the dynamics of humanoid systems. The dynamic simulation environment used to evaluate the control frameworks in Chapters 4 - 7 is described. Chapter 2 also makes a new connection between the dynamics of the centroidal momentum of a humanoid and its joint-space dynamic equations of motion. The humanoid control in the following chapters showcases the capabilities that centroidal momentum control can provide.

Chapters 3 and 4 focus on aspects of task-space control at the computational and implementation levels. Chapter 3 presents a new algorithm for the computation of the task-space (operational-space) inertia matrix. This matrix is an important quantity used in many task-space control applications and its fast computation is required for efficient real-time control. In comparison to other algorithms which compute the same quantity, the proposed algorithm possesses the lowest order\(^2\) and is the fastest computationally for some legged topologies. Chapter 4 describes a new formulation of the task-space control problem as a conic optimization problem. The problem is able to be solved at real-time control rates of 200Hz, approximately twice as fast as a previous Quadratic Program (QP) formulation studied. In applications to a dynamic

\(^2\)In the robotics community the use of the term “low-order” is commonly used to describe algorithms in place of the more globally used “low computational complexity”.

13
kick and a standing jump, the humanoid displays emergent upper-body motion to maintain balance without authoring from the designer. In addition, the generality of the approach to uneven terrain scenarios is shown.

Chapters 5 and 6 build upon the task-space controller through integration of simple-model-based control of high-speed running. Chapter 5 describes a new gait design methodology for humanoid robots based on a 3D-SLIP model and describes a simple deadbeat control approach for this model which provides automatic footstep planning for online control with the humanoid. With this controller, the humanoid is capable of Olympic speed running and is robust to push disturbances. Chapter 6 extends upon this forward running controller to enable high-speed turns. As opposed to previous theoretical 3D-SLIP steering work, the methods here address leg separation in the humanoid, which limits the positioning of feet and requires the inside and outside legs to generate different amounts of centripetal force during the turn. Through a few judicious modifications to the gait generation techniques for forward running, the methods are adapted to produce running turns.

Chapter 7 presents modifications to the 3D-SLIP model to perform a running long jump. Motivated by a simpler 2D model from biomechanics, the modifications and trajectory optimization approach here are shown to result in long-jump techniques that are comparable to human long jumpers. Namely, the angle of the velocity vector at takeoff is shown to be an important quantity to the performance of a maximum span long jump, with an optimum value that is significantly different from a traditional ballistic optimum of 45°.

Chapter 8 provides concluding remarks and introduces many topics for future work that have emerged from this dissertation.
Chapter 2

Humanoid Dynamics Model

2.1 Introduction

This chapter describes the humanoid model used in this dissertation and introduces the conventions and notation to model its dynamics. The model used has kinematic mobility comparable to common humanoids, in terms of its degrees of freedom (DoFs) that contribute to locomotion, and has a mass distribution similar to the average human male [148]. While many control algorithms for high-speed locomotion are simplified by the absence of significant leg mass, the leg mass modeled in this work requires the control algorithms to compensate for the destabilizing dynamic effects that the legs can have on the torso. In order to compactly represent the dynamics of the bodies in this system, spatial notation [33] is adopted, and a brief overview is given in this chapter.

As an important aggregate feature of motion, this chapter also reviews the notion of the Centroidal Momentum of a humanoid robot as originally introduced by Orin and Goswami [95, 97]. The centroidal momentum of the humanoid consists of its net linear momentum and net angular momentum about its center of mass (CoM). While the net linear momentum has a well known connection to the motion of the system’s
CoM, the centroidal angular momentum has been shown to exhibit tight regulation around zero during walking [49]. Its regulation for other movements is also a useful heuristic which contributes to many of the capabilities demonstrated in Chapters 4 - 7. This chapter describes a new link between the joint-space equations of motion and the dynamics of the centroidal momentum. These centroidal dynamics are important to the increasing number of applications of centroidal momentum control. Thus, this new link to the joint-space dynamics will have impact to simplify the process of implementing future centroidal momentum control methods.

The remainder of the chapter is organized as follows. First, Section 2.2 details the humanoid model for this work. The environment used to simulate this system is also detailed. Next, Section 2.3 introduces the conventions and notation to describe the dynamics of the humanoid. This section includes a high-level overview of spatial notation, which is foundational to the new dynamics algorithm developed in Chapter 3. Finally, Section 2.4 introduces the standard dynamic equations of motion for the humanoid in joint space, and reviews the Articulated-Body Algorithm [32] for rigid-body simulation. The new link between the joint-space equations of motion and the centroidal dynamics of the humanoid is also made, which is an important finding for the control algorithms developed in Chapter 4.

2.2 Humanoid Model

The humanoid model used in this work is shown in its simulation environment in Figure 2.1. This model consists of 26 degrees of freedom (DoF) with 20 actuated joint DoFs and a 6 DoF floating torso. Spherical (ball and socket) joints are modeled at the hips and shoulders, providing 6 DoFs in each leg, and 4 DoFs in each arm.
The remainder of this section describes the kinematic and dynamic parameters of the model and presents the details of its simulation environment.

Figure 2.2 provides the main kinematic dimensions of the humanoid with numeric values provided in Table 2.1. A complete specification of the model is provided in Appendix A. Overall, the model has a mass of 72.575 kg (160 lbs weight), and stands 1.829 m (6 feet) tall. The mass of each link, relative to the overall mass, is modeled after a 50-th percentile male as provided by Winter [148], while the relative segment lengths and locations are from the graphical model by Lee [67]. The Winter data also provides the center of mass locations for each body. Inertia tensors are estimated based on a simple equidensity mass distribution for each segment. Full details on the center of mass and inertia parameters are provided in Appendix A.
Figure 2.2: Definitions for main dimensions of the humanoid model. The black circles denote the centers of rotation for the spherical joints at the hips and shoulders as well as the universal joints at the ankles.
<table>
<thead>
<tr>
<th>Rigid Body</th>
<th>Main Geometric Parameters (m)</th>
<th>Total Mass (kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torso, Head, and Neck</td>
<td>$\ell_t = 0.615$</td>
<td>41.948</td>
</tr>
<tr>
<td></td>
<td>$w_t = 0.340$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$d_t = 0.170$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$w_h = 0.254$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$s_h = 0.053$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$w_s = 0.375$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$s_s = 0.027$</td>
<td></td>
</tr>
<tr>
<td>Thigh</td>
<td>$\ell_{th} = 0.483$</td>
<td>7.258</td>
</tr>
<tr>
<td>Shank</td>
<td>$\ell_{sh} = 0.437$</td>
<td>3.375</td>
</tr>
<tr>
<td>Foot</td>
<td>$\ell_f = 0.278$</td>
<td>1.052</td>
</tr>
<tr>
<td></td>
<td>$w_f = 0.161$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$h_f = 0.055$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$s_f = 0.070$</td>
<td></td>
</tr>
<tr>
<td>Upper Arm</td>
<td>$\ell_{ua} = 0.2707$</td>
<td>2.032</td>
</tr>
<tr>
<td>Forearm and Hand</td>
<td>$\ell_{fa} = 0.4389$</td>
<td>1.597</td>
</tr>
</tbody>
</table>

Table 2.1: Numeric values of the main kinematic and dynamic parameters of the humanoid model. Further details are provided in Appendix A.

Each of the legs is equipped with 6 DoFs. The torso is connected to each leg through a 3 DoF hip joint, where the rotational centers of the hips are separated by a distance $w_h$ which is slightly narrower than the overall torso width $w_t$. A single degree of freedom is modeled at the knee, while foot pitch and roll are modeled at the ankle. The foot is treated as a single rigid body, as current humanoids lack any foot articulation.

Each of the arms is equipped with 4 DoFs. Similarly, the torso is connected to each arm through a 3 DoF shoulder joint, where the rotational centers of the shoulders are separated by a distance $w_s$ which is slightly wider than the torso. A single degree of freedom is modeled at the elbow, while the wrists and hands are assumed to be rigidly connected. These degrees of freedom in the hands are not modeled, as they do


\begin{table}[h]
\centering
\begin{tabular}{lcc}
\hline
Direction & Compliance & Damping \\
\hline
Normal & 100 kN/m & 1200 N·s/m \\
Tangential & 100 kN/m & 2400 N·s/m \\
\hline
\end{tabular}
\caption{Spring-damper contact model parameters.}
\end{table}

not have substantial contributions to the performance of legged locomotion. Likewise, the head and the neck are assumed to be rigidly attached to the torso.

\section{2.2.1 Simulation}

Full 3D dynamic simulation of the system is carried out using the DynaMechs simulation package developed by McMillan, Orin, and McGhee [85]. This C/C++ package provides an efficient implementation of the Articulated-Body Algorithm [32, 83], summarized in Section 2.4.1, for forward dynamics simulation. It also includes basic graphic rendering of the humanoid with OpenGL, and provides the opportunity to include additional user graphics.

The package includes a compliant contact model to simulate contact interactions with the environment. In this work, the humanoid is assumed to interact with its environment only through interactions between the ground and its feet. Contact forces are modeled at the four corners of the feet, and are computed with a penalty-based spring-damper model of contact. Linear springs and dampers are used in the vertical and tangential directions, with compliance and damping characteristics summarized in Table 2.2. Static and kinetic friction coefficients of $\mu_s = 2.0$ and $\mu_k = 1.8$, respectively, limit the tangential forces and model foot slip.
2.3 Conventions and Notation for Rigid-Body Dynamics

This section outlines the conventions and notation that will be used to describe the connectivity and dynamics of a floating-base rigid-body system. The conventions in this paper match those used by Featherstone and Orin [33] and rely heavily on 6D spatial vector algebra. Further details on 6D spatial vectors can be found in the tutorials by Featherstone [26, 27].

2.3.1 Kinematic Trees - Describing Connectivity

The connectivity of any rigid body system can be represented by a graph, wherein each node of the graph represents a rigid body and each arc represents a connecting joint between bodies. The rigid body model of the humanoid is shown in Figure 2.3 along with its connectivity graph. Due to the absence of closed kinematics loops within the robot, the connectivity graph of this system is a tree.

The numbering of the bodies in this connectivity tree is selected as follows. The motion of the system is measured with respect to a fixed inertial frame, denoted as Body 0. A privileged body in the system, the torso in this work, is selected as a "floating base" and is denoted as Body 1. The remaining bodies are numbered 2 through $N_B = 13$ in any manner such that Body $i$’s predecessor (towards the floating base), denoted $p(i)$, is labeled less than $i$. Connecting joints are labeled 1 through $N_B$ such that Joint $i$ connects Body $p(i)$ to Body $i$. As shown in Figure 2.3, Joint 1 is defined as a 6-DoF virtual joint which connects the inertial frame to the floating base.

Given these conventions, a few additional definitions can be made to support the algorithm development in Chapter 3. The set $c(i)$ is defined as the set of children of
Figure 2.3: (a) Rigid-body model of the humanoid and (b) its connectivity tree. Due to the absence of universal joints within the DynaMechs simulation package, the 2-DoF ankle joint between the shank and foot is modeled through the inclusion of a zero mass intermediate ankle body (numbers 4 and 8 in the diagram). Sequential revolute joints are then modeled between the shank and ankle body, and between the ankle body and foot.
Body $i$

$$c(i) = \{ j \mid p(j) = i \}.$$  \hfill (2.1)

For example, in Figure 2.3, $c(1) = \{2, 6, 10, 12\}$ and $c(4) = \{5\}$. The set $\kappa(i)$ is defined as the set of joints that support Body $i$

$$\kappa(i) = \begin{cases} \emptyset & \text{if } i = 0 \\ \{i\} \cup \kappa(p(i)) & \text{otherwise} \end{cases}.$$  \hfill (2.2)

With this definition, a joint supports Body $i$ precisely when it lies on the path between Body $i$ and the fixed base. For example, $\kappa(3) = \{1, 2, 3\}$ and $\kappa(10) = \{1, 10\}$. Additionally, the nearest common ancestor for a pair of bodies can be defined through the use of the support sets as

$$\text{ancest}(i, j) = \max(\kappa(i) \cap \kappa(j)).$$  \hfill (2.3)

For example, in Figure 2.3, $\text{ancest}(3, 10) = 1$ and $\text{ancest}(3, 4) = 3$. Finally, the set $c^*(i)$ is defined as the set of all bodies in the subtree rooted at Body $i$

$$c^*(i) = \{ j \mid i \in \kappa(j) \}.$$  \hfill (2.4)

2.3.2 Overview of Spatial Notation

6D spatial vectors provide a compact notation to describe rigid-body dynamics and to develop rigid-body dynamics algorithms. A short introduction is provided here, while the interested reader may refer to the tutorials by Featherstone [26, 27] and the summary by Featherstone and Orin [33] for further details. Given the rigid-body system shown in Figure 2.3, a coordinate frame is attached to each body to describe motion conveniently with respect to a local basis. The kinematic relationship between neighboring bodies can be described with the general joint notation of Roberson and
Joint Type  # DoFs  \( \Phi \)
---
Revolute Joint  1  \([0, 0, 1, 0, 0, 0]^T\)
Spherical Joint  3  \([1_{3 \times 3} \ 0_{3 \times 3}]^T\)
Floating Base  6  \(1_{6 \times 6}\)

Table 2.3: Free modes of motion for joint types used in this work.

Schwertassek [110]. With this notation, if Joint \( i \) has \( n_i \) DoFs, the spatial velocity \( \mathbf{v}_i \in \mathbb{R}^6 \) of Body \( i \) is related to its predecessor as

\[
\mathbf{v}_i = \begin{bmatrix} \omega_i \\ \mathbf{v}_i \end{bmatrix} = i^{X}_{p(i)} \mathbf{v}_{p(i)} + \Phi_i \dot{\mathbf{q}}_i ,
\]

where \( \omega_i \in \mathbb{R}^3 \) and \( \mathbf{v}_i \in \mathbb{R}^3 \) are the angular and linear velocities\(^3\) of Body \( i \) and \( \dot{\mathbf{q}}_i \in \mathbb{R}^{n_i} \) collects Joint \( i \)'s joint rates. The matrix \( i^{X}_{p(i)} \in \mathbb{R}^{6 \times 6} \) above provides a transformation of spatial motion vectors from frame \( p(i) \) to frame \( i \). \( \Phi_i \in \mathbb{R}^{6 \times n_i} \) is a full-column-rank matrix with mutually orthogonal columns that describes Joint \( i \)'s free modes of motion.\(^4\) This matrix is dependent on joint type, but takes simple forms for the joints used here as described in Table 2.3.\(^5\) The total number of degrees of freedom \( n \) in the system is then given as

\[ n = \sum_{i=1}^{N_B} n_i. \]

The spatial transformation matrix \( i^{X}_{p(i)} \) in Eq. 2.5 can be formed from the position vector \( p^{(i)} \mathbf{p}_i \in \mathbb{R}^3 \) (from the origin of \( p(i) \) to the origin of \( i \)) and the rotation matrix

\(^3\)Upright characters \( \mathbf{v} \) and \( \mathbf{f} \) will be used to represent a spatial velocity and force, respectively. Script characters \( \mathbf{v} \) and \( \mathbf{f} \) will be used to denote the linear velocity or force component of a spatial quantity [33].

\(^4\)More generally, the joint matrix \( \Phi_i \) need only be full-column rank, as described in [33, 110]. However, in the special case that \( \Phi_i \) has mutually orthogonal columns, then the force created by a joint torque \( \mathbf{\tau}_i \) is \( \Phi_i \mathbf{\tau}_i \). All joints used here have mutually orthogonal columns, and possess this additional property.

\(^5\)Note that with these conventions, \( \dot{\mathbf{q}}_i \) for spherical joints provides \( i^{\omega}_{i/p(i)} \), the angular velocity of Body \( i \) relative to its predecessor and expressed in body fixed coordinates.
\( iR_{p(i)} \in \mathbb{R}^{3 \times 3} \) which transforms 3D vectors from \( p(i) \) coordinates to \( i \) coordinates

\[
iX_{p(i)} = \begin{pmatrix} iR_{p(i)} & 0 \\ iR_{p(i)}S(p(i)p_i) & iR_{p(i)} \end{pmatrix}. \tag{2.6}
\]

The quantity \( S(p) \in \mathbb{R}^{3 \times 3} \) used here is the skew-symmetric cross product matrix for \( p \in \mathbb{R}^3 \) which satisfies \( S(p) \omega = p \times \omega \) for any \( \omega \in \mathbb{R}^3 \). Similarly, the matrix \( iX_{p(i)} \) provides a spatial transformation of spatial forces from \( i \) coordinates to \( p(i) \) coordinates.

Body \( i \)'s spatial inertia tensor, \( I_i \in \mathbb{R}^{6 \times 6} \), maps spatial motion vectors to spatial momentum vectors and is represented as

\[
I_i = \begin{pmatrix} \tilde{I}_i & m_iS(c_i) \\ m_iS(c_i)^T & m_i1 \end{pmatrix}. \tag{2.7}
\]

The quantity \( c_i \in \mathbb{R}^3 \) is the vector to Body \( i \)'s center of mass (with respect to the origin of frame \( i \)), \( m_i \) is the body’s mass, and \( \tilde{I}_i \in \mathbb{R}^{3 \times 3} \) is the standard inertia tensor about the origin of frame \( i \). This quantity is related to the inertia about Body \( i \)'s CoM \( \tilde{I}_i^{\text{cm}} \) through

\[
\tilde{I}_i = \tilde{I}_i^{\text{cm}} + m_iS(c_i)S(c_i)^T. \tag{2.8}
\]

The spatial inertia provides a link between the spatial velocity of a body and its spatial momentum. The spatial momentum of Body \( i \), \( h_i \in \mathbb{R}^6 \), is then made up of its linear momentum \( l_i \in \mathbb{R}^3 \) and angular momentum \( k_i \in \mathbb{R}^3 \) (about the origin of frame \( i \)) as

\[
h_i = \begin{bmatrix} k_i \\ l_i \end{bmatrix} = I_i v_i \tag{2.9}.
\]

The net spatial force (moment and linear force) applied to a body is equal to its rate of change in spatial momentum. A key feature of spatial notation is that it allows
Newton’s and Euler’s rigid-body equations of motion to be represented compactly as

\[
f_{net}^i = I_i a_i + v_i \times^* I_i v_i
\]  

(2.10)

where \(a_i \in \mathbb{R}^6\) is the spatial acceleration of Body \(i^6\) and

\[
f_{net}^i = \begin{bmatrix}
    n_{net}^i \\
    f_{net}^i
\end{bmatrix}
\]  

(2.11)

represents the net spatial force on Body \(i\). This spatial force is comprised of the net moment \(n_{net}^i \in \mathbb{R}^3\) on Body \(i\) (about the origin of frame \(i\)) and the net force \(f_{net}^i \in \mathbb{R}^3\) on Body \(i\). The spatial motion-force cross product operator \(\times^* : \mathbb{R}^6 \times \mathbb{R}^6 \to \mathbb{R}^6\) in Eq. 2.10 is defined by the formula

\[
[\omega \ v] \times^* f = \begin{pmatrix}
    S(\omega) & S(v) \\
    0 & S(\omega)
\end{pmatrix} f.
\]  

(2.12)

Spatial inertias also provide useful compositional properties. In the case that two rigid bodies, with inertias \(I_i\) and \(I_j\), are rigidly joined together to form a composite rigid body, the spatial inertia of the composite, \(I_{tot}\) is given as

\[
I_{tot} = I_i + I_j,
\]  

(2.13)

where it is assumed that all inertias are represented with respect to the same coordinate frame. An important quantity within robot dynamics is the composite-rigid-body inertia of a subtree \(I^C_i\) defined by

\[
I^C_i = \sum_{j \in c^{*} (i)} jX_i^T I_j jX_i
\]  

(2.14)

where \(I^C_i\) represents the inertia of the subtree, as felt at the Body \(i\), given that all the joints in the subtree are locked.

\(^6\)Note that the spatial acceleration of a rigid body is a rate of change in its spatial velocity, and is comprised of rotational and linear components as expected. However, the linear component of spatial acceleration for a rigid body has a nuanced difference with the conventional linear acceleration of a point as described in [28].
2.4 Whole-Body Dynamics

The previous section has outlined compact methods to describe the dynamics of each rigid body within the humanoid model. This section will detail efficient methods which rely on spatial rigid-body dynamics to construct the dynamic equations of motion for the humanoid system as a whole. The joint-space dynamic equations of motion are first presented, followed by an overview of efficient methods for forward simulation of these equations. Finally, the centroidal dynamics of the humanoid are reviewed and a connection is made to the joint-space equations of motion.

2.4.1 Joint-Space Dynamic Equations of Motion

To begin, the velocity of the system can be described by its joint rate vector \( \dot{q} \in \mathbb{R}^n \)

\[
\dot{q} = \begin{bmatrix}
\dot{q}_1^T & \cdots & \dot{q}_N^T
\end{bmatrix}^T,
\]

which collects each of the joint rates in the system. The system acceleration vector \( \ddot{q} \in \mathbb{R}^n \) is partitioned similarly. With a slight liberty of notation, \( q \) is defined to provide the configuration of the system,\(^7\) which evolves on \( SE(3) \times SO(3)^4 \times \mathbb{R}^8 \) due here to the existence of a single floating-base joint, four spherical joints, and 8 revolute joints [136]. In implementation, the rotational configuration of the floating base and each of the spherical joints is stored using quaternions. Thus, since quaternions require 4 parameters to describe the configuration of 3 DoFs, \( q \in \mathbb{R}^{n+5} \) here. The

\(^7\)For the revolute joints in the system, the individual joint rates \( \dot{q}_i \) are simply scalars \( \dot{q}_i \) which can be integrated to form the angle \( q_i \) of joint \( i \). For the four spherical joints (hips and shoulders), where the joint configuration evolves on \( SO(3) \), or the floating base joint, which evolves on \( SE(3) = SO(3) \times \mathbb{R}^3 \), these joint rates are not the time derivative of any configuration variables. However, through a quaternion parameterization of \( SO(3) \), common formulae [136] can be used to relate the joint rates to rates of change in the configuration parameters.
standard dynamic equations of motion are then:

\[ H(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) = S_a^T \tau + J_s(q)^T F_s \]  \hspace{1cm} (2.15)

where \( H \in \mathbb{R}^{n\times n} \), \( C \dot{q} \in \mathbb{R}^n \), and \( G \in \mathbb{R}^n \) are the familiar mass matrix, velocity product terms, and gravitational terms, respectively \([33]\). Here \( F_s \in \mathbb{R}^{6N_s} \) collects spatial ground reaction forces (GRFs) for \( N_s \) feet in support and \( J_s \in \mathbb{R}^{6N_s \times n} \) is a combined support Jacobian. The matrix \( S_a = [0_{(n-6)\times 6} \hspace{0.5cm} 1_{(n-6)\times (n-6)}] \) is a selection matrix for the actuated joints. This form will be important to the results in Section 2.4.2.

The mass matrix \( H \) can be efficiently computed using the \( O(n^2) \) composite-rigid-body algorithm \([33]\) while bias terms \( C \dot{q} \) and \( G \) in Eq. 2.15 can be computed with \( O(n) \) inverse dynamics algorithms \([33]\). The contact Jacobians can be computed with \( O(n) \) algorithms from Orin and Schrader \([96]\).

**Articulated-Body Algorithm for Forward Dynamics**

Although the quantities \( H, C \dot{q}, J_s, \) etc., will be needed to perform whole-body control, solution of Eq. 2.15 for \( \ddot{q} \) is central to the problem of forward dynamics simulation. The problem to find \( \ddot{q} \), given \( q, \dot{q}, F_s \) and \( \tau \), is commonly referred to as the *forward dynamics* problem. For high degree of freedom systems, such as the humanoid here, it is often most efficient to solve the forward dynamics problem through the use of the \( O(n) \) Articulated-Body Algorithm (ABA) developed by Featherstone \([32]\). The ABA never actually forms Eq. 2.15, but instead considers the dynamics of articulated subtrees through recursive approaches.
As a competing approach, Featherstone [29] alternately showed how sparsity in the mass matrix $H$ can be exploited to solve the forward dynamics problem. Kinematic branching in rigid-body systems leads to a specific pattern of non-zero elements within $H$, which can be exploited within traditional Cholesky factorization and back-substitution methods to solve Eq. 2.15 for $\ddot{q}$. The performance of this approach is dependent on the amount of branching found in a system’s kinematic tree. As a result, branch-induced sparsity approaches are preferable to the ABA only when applied to highly-branched mechanisms.

The Articulated-Body Algorithm requires three algorithmic sweeps across the connectivity tree of the system in order to solve the forward dynamics problem for $\ddot{q}$. The first pass of the algorithm computes system kinematics and proceeds outward from base to tips. This pass calculates body velocities $v_i$ from Eq. 2.5, and determines velocity-dependent acceleration bias terms $\zeta_i \in \mathbb{R}^6$ that satisfy

$$\mathbf{a}_i = \hat{\mathbf{T}}_i \mathbf{a}_p(\mathbf{i}) + \Phi_i \ddot{\mathbf{q}} + \zeta_i.$$ (2.16)

The second pass of the algorithm, inward from tips to base, then computes quantities known as “articulated-body inertias” $\mathbf{I}_i^A$ and associated bias forces $\mathbf{p}_i^A$. Considering the subtree rooted at Body $i$, the key concept of the ABA is that the articulated-body inertias can be used to express dynamics of this subtree as [33]

$$\mathbf{f}_i = \mathbf{I}_i^A \mathbf{a}_i + \mathbf{p}_i^A,$$ (2.17)

where $\mathbf{f}_i$ is the interaction force transmitted to Body $i$ from its predecessor. In contrast to the composite-rigid-body (locked) inertia $\mathbf{I}_i^C$ (Eq. 2.14), the articulated-body inertia $\mathbf{I}_i^A$ for the subtree rooted at Body $i$ represents the inertia that would be felt by a force acting at Body $i$ when all joints in the subtree are free to rotate.
Formula for $I_i^A$ and $p_i^A$ can be derived through recursive update rules that amount to a dynamic programming sweep backwards across the tree. To see this more formally, it is noted that the joint accelerations $\ddot{q}$ must satisfy the Gauss principle of least constraint,

$$\arg \min_{q_1:N_B} \sum_{i=1}^{N_B} (a_i - a_{i,u})^T I_i (a_i - a_{i,u})$$

s.t. $a_i = X_{p(i)} a_{p(i)} + \Phi_i \ddot{q}_i + \zeta_i \quad \forall i \in 1, \ldots, N_B$

where each $a_{i,u}$ is the unconstrained acceleration which results from simulating Body $i$ individually without any joint connectivity constraints. The form of this problem matches that found in the Linear Quadratic (LQ) optimal control literature, allowing an efficient $O(n)$ solution for $\ddot{q}_{1:N_B}$ through dynamic programming approaches as originally discovered by Vereshchagin [134]. Featherstone [32] later, and independently, derived a similar solution for forward dynamics without simply applying existing dynamic programming algorithms. Through a mechanically focused derivation, the update laws for the optimal-cost-to-go used by Vereshchagin, were traded for update laws which recursively form Eq. 2.17 and provide greater insight into the the dynamics of articulated subtrees. Although a connection has been made here between LQ optimal control and forward dynamics, Rodriguez [111] also independently derived an $O(n)$ algorithm to solve the forward dynamics problem through connections to linear filtering and smoothing theory.

The final pass of the ABA proceeds outwards from base to tips, and computes joint accelerations $\ddot{q}_i$ along with body accelerations $a_i$. This pass again can be derived within the context of optimal control theory, where the body accelerations $a_i$...
take the form of optimal state trajectories and the joint accelerations $\ddot{q}_i$ are the corresponding optimal controls. The ABA will be used for forward simulation to assess the performance of the control algorithms developed in this dissertation. In addition, the efficient structure of its $O(n)$ computation will be a launching point for the new efficient dynamics algorithms developed in Chapter 3.

2.4.2 Centroidal Dynamics

The centroidal momentum of a humanoid robot was originally introduced by Orin and Goswami [95, 97]. Given a coordinate frame $G$ located at the system’s CoM, the centroidal momentum of the humanoid robot is given by the sum of its body spatial momenta (properly transformed to the CoM)

$$h_G = \sum_{i=1}^{N_B} X_G^T h_i \quad (2.20)$$

$$= \sum_{i=1}^{N_B} X_G^T I_i v_i \quad (2.21)$$

Orin and Goswami defined a centroidal momentum matrix (CMM) $A_G \in \mathbb{R}^{6 \times n}$ which relates the joint rates $\dot{q}$ to the centroidal momentum $h_G$ as

$$h_G = A_G \dot{q} \quad (2.22)$$

When performing centroidal momentum control, the derivative of Eq. 2.22 is often required

$$\dot{h}_G = A_G \ddot{q} + \dot{A}_G \dot{q} \quad (2.23)$$

where Eq. 2.23 is referred to as the centroidal dynamics of the system. Although various algorithms have been presented to compute $A_G$ and $\dot{A}_G \dot{q}$ ([18] for instance), their implementation requires specialized code and their published derivations have
been incorrect due to their complexity. The following theorem provides a connection between these quantities and the joint-space dynamic equations of motion, allowing existing efficient algorithms for $H$ and $C_\dot{q}$ to be used to compute the centroidal dynamics.

**Theorem 2.1** (Connection between the Centroidal Dynamics and the Joint-Space Equations of Motion). The Centroidal Momentum Matrix $A_G$ can be constructed from the mass matrix $H$ through the relationship

$$A_G = \frac{1}{X^T_G} S_1 H \quad (2.24)$$

where $S_1 = [1_{6\times6} \ 0_{6\times(n-6)}]$ is the floating-base selector. Similarly, the centroidal dynamics velocity-dependent bias term $\dot{A}_G \dot{q}$ can be constructed from the joint-space Coriolis force $C_\dot{q}$ by

$$\dot{A}_G \dot{q} = \frac{1}{X^T_G} S_1 C_\dot{q} \quad . \quad (2.25)$$

**Proof of Theorem 2.1.** For the purposes of this proof assume that the floating base is actuated, and consider the dynamic equations of motion in the absence of all other external forces (GRFs and gravity). This provides:

$$H\ddot{q} + C\dot{q} = [f_1^T \ \tau^T]^T \quad (2.26)$$

where $f_1 = [n_1^T \ f_1^T]^T$ is a spatial force applied to the floating-base and expressed at the floating-base coordinate system. Since all the joint torques $\tau$ are internal, $f_1$ is also the net spatial force on the system. Defining $f_G$ as the net spatial force expressed in frame $G$, the following is obtained

$$f_G = \frac{1}{X_G^T} f_1 = \frac{1}{X_G^T} S_1 (H\ddot{q} + C\dot{q}) \quad . \quad (2.27)$$
By Newton’s equations of motion, it follows that the rate of change in centroidal momentum $\dot{h}_G$ is equal to $f_G$. Thus,

$$\dot{h}_G = A_G \dot{q} + \dot{A}_G \dot{q}$$  \hspace{1cm} (2.28)

$$= \mathbf{1}X_T G S_1 (H \ddot{q} + C \dot{q})$$  \hspace{1cm} (2.29)

which provides the new relationships

$$A_G = \mathbf{1}X_T G S_1 H$$  \hspace{1cm} and  \hspace{1cm} (2.30)

$$\dot{A}_G \dot{q} = \mathbf{1}X_T G S_1 C \dot{q}$$  \hspace{1cm} (2.31)

These relationships may be unsurprising in light of observations made by Wieber [146] that the Newton and Euler equations for the whole system are embedded in Eq. 2.15 for floating-base systems. Additionally, Featherstone [30, Chap. 9] shows that the net spatial momentum of a floating-base system is embedded in its generalized (or canonical) momentum $H \dot{q}$. Still, the above equations are the first to link these relationships to the centroidal momentum described by Orin and Goswami [95, 97].

### 2.5 Summary

This chapter has described the humanoid model used in this work and has introduced the joint-space dynamic equations of motion which are used to simulate the system. The simulation environment and Articulated-Body Algorithm for forward dynamics simulation have been presented. Finally, the centroidal dynamics of the humanoid have been reviewed, and new connections to the joint-space dynamic equations have been presented. Control of centroidal momentum will be a common
thread employed in Chapters 4 - 7, and is shown to be a powerful approach. Given
the increasing use of centroidal momentum control in humanoid robotics at large, the
new connections between the centroidal and joint-space dynamics will be of important
future use in humanoid balance control.
Chapter 3

A Reduced-Order Recursive Algorithm for the Computation of the Operational-Space Inertia Matrix

3.1 Introduction

This chapter presents a new recursive algorithm to efficiently compute the Operational-Space Inertia Matrix. This matrix is central to the operational-space dynamics formulation, and is an required for any real-time operational-space control application. The new algorithm for the Operational-Space Inertia Matrix is a recursive algorithm in that it successively considers the dynamics of rigid-body subsystems, similar to the Articulated-Body Algorithm (ABA) [30, 32]. Recursive algorithms have been commonly used to provide efficient solutions to important problems in rigid-body dynamics, such as the in the Composite-Rigid-Body Algorithm (CRBA) to compute the mass matrix $H$ [137], or the Recursive-Newton-Euler Algorithm (RNEA) to solve inverse dynamics [76]. Although a recursive algorithm for the Operational-Space Inertia Matrix has been presented by Rodriguez, Jain, and Kreutz-Delgado [112], the new algorithm here has a lower computational complexity$^8$ and is shown to greatly outperform this competing algorithm.

$^8$Within the robot dynamics community, algorithms with a low computational complexity are also referred to as “low-order algorithms”.

35
The Operational-Space Inertia Matrix is a key component to the operational-space formulation, which describes the dynamics of a set of end-effectors. Given a set of \( m \) end-effectors, with their combined spatial velocity \( \dot{x} \in \mathbb{R}^{6m} \), the end-effector accelerations \( \ddot{x} \) follow

\[
\Lambda(q) \ddot{x} + \mu(q, \dot{q}) + \rho(q) = \hat{F}
\]

(3.1)

where \( \Lambda(q) \in \mathbb{R}^{(6m) \times (6m)} \) is the operational-space inertia matrix, \( \mu \in \mathbb{R}^{6m} \) and \( \rho \in \mathbb{R}^{6m} \) are velocity-dependent and gravity-dependent force biases, and \( \hat{F} \in \mathbb{R}^{6m} \) is the effective force (operational force) placed on the end-effectors through the effects of joint torques and external forces. These equations can be derived starting from the joint-space dynamics described in Section 2.4.

The algorithm presented in this chapter, named the Extended-Force-Propagator Algorithm (EFPA), computes the inverse of the operational-space inertia matrix \( \Lambda^{-1} \) and achieves a computational complexity of \( O(N_B + md + m^2) \) where \( N_B \) is the number of bodies in the system (Section 2.3), and \( d \) is the depth of the system’s connectivity tree. This computational complexity is an improvement over the \( O(N_B + m^2d) \) Rodriguez, Jain, and Kreutz-Delgado algorithm (RJKA) [112]. The key feature of the EFPA is its explicit calculation and use of matrices that propagate a force applied at an end-effector directly to an equivalent force at another body in the mechanism, which may be several joints away from the end-effector. This feature is responsible for the algorithm’s low complexity. The use of these new matrices, called extended force propagators, alters the structure of the algorithm in comparison to the RJKA, such that these algorithms have relatively few intermediate results in common.

The remainder of this chapter is organized as follows. Section 3.2 briefly reviews the operational-space dynamic equations of motion as they relate to the joint-space
dynamic equations of motion (Section 2.4). Section 3.3 gives an overview and derives the EFPA. Section 3.4 compares the algorithm’s computational performance to competing algorithms, and Section 3.5 summarizes the chapter.

### 3.2 Operational-Space Dynamics

This section briefly introduces the various quantities that are used to describe operational-space dynamics as originally formulated by Khatib [60]. Given a set of \( m \) end-effectors, consider the joint-space dynamic equations of motion (Eq. 2.15) in the absence of ground reaction forces, but instead under the influence of spatial forces that act on each of the end-effectors. The joint-space dynamics are then

\[
H(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) = S_a^T \tau + J(q)^T F \tag{3.2}
\]

where \( F \in \mathbb{R}^{6m} \) collects end-effector forces and \( J \in \mathbb{R}^{(6m) \times n} \) is a combined end-effector Jacobian that relates joint rates to end-effector velocities as,

\[
\dot{x} = J(q) \dot{q}. \tag{3.3}
\]

Premultiplication of Eq. 3.2 by \( JH^{-1} \) and incorporation of the time derivative of Eq. 3.3 provides:

\[
\ddot{x} = JH^{-1} S_a^T \tau + \Lambda^{-1} F + J \dot{q} - JH^{-1} (C + G), \tag{3.4}
\]

where

\[
\Lambda^{-1}(q) = JH^{-1} J^T. \tag{3.5}
\]
Left-multiplication of Eq. 3.4 by $\Lambda(q) = \left( JH^{-1}J^T \right)^{-1}$ then recovers Eq. 3.1 where

$$\mu = \Lambda \left( JH^{-1}C\dot{q} - J\dot{\dot{q}} \right),$$  \hspace{1cm} (3.6)
$$\rho = \Lambda JH^{-1}G, \text{ and}$$  \hspace{1cm} (3.7)
$$\hat{F} = F + \Lambda JH^{-1}S_a^T \tau.$$  \hspace{1cm} (3.8)

These classic operational-space equations of motion are a starting point for any operational-space control approach, and thus it is important to have efficient algorithms to compute each of the quantities. While the algorithm here is designed to compute only $\Lambda^{-1}$, methods to efficiently compute $\mu$ and $\rho$ can be found in [33, Section 2.5.4].

In order to compute $\Lambda^{-1}$ with recursive algorithms, it is first necessary to introduce some additional notation which describes the kinematic connectivity of the end-effectors. It is assumed that each of the $m$ end-effectors is rigidly attached to a single body in the system. The connectivity tree of the system is thus extended to accommodate these end-effectors, which are numbered from $N_B + 1$ to $N_B + m$. As a result, $p(k)$ denotes the body to which end-effector $k$ is rigidly attached. Through the use of $c^*(i)$ (Eq. 2.4), the following sets are defined to describe the end-effectors supported (ES) by each joint in the tree

$$ES(i) = c^*(i) \cap \{N_B + 1, \ldots, N_B + m\}.$$  \hspace{1cm} (3.9)

### 3.3 The Extended-Force-Propagator Algorithm

This section provides a recursive algorithm, named the Extend-Force-Propagator Algorithm (EFPA), to compute the inverse of the operational-space inertia matrix. The high-level intuition behind the EFPA is first discussed, followed by a detailed
derivation. To simplify computations, zero joint rates, zero joint torques, and no
gravity are assumed for the algorithm, since they have no effect on $\Lambda^{-1}$, as observed
through Eq. 3.5. With these assumptions, the kinematics (Eq. 2.16) and dynamics
(Eq. 2.10) of each rigid body are simplified to

$$a_i = iX_{p(i)} a_{p(i)} + \Phi_i \ddot{q}_i, \text{ and}$$

$$f_{net}^i = I_i a_i. \tag{3.11}$$

3.3.1 Algorithm Overview

The EFPA is inspired from the final two recursions of the Articulated-Body Al-
gorithm. Its main feature is the recursive calculation and re-use of matrices called
extended force propagators, denoted $^kX^T_i \in \mathbb{R}^{6 \times 6}$, which propagate a force from end-
effector $k$ to Body $i$. With these quantities, Body $i$ may be arbitrarily far away from
end-effector $k$. While the standard spatial force transform $^kX^T_i$ provides a transfor-
mation of spatial forces from $k$ to $i$ as if all intermediate bodies are locked at the
joints, the matrix $^kX^T_i$ provides a transformation as if the joints are free to move. As
a result, $^kX^T_i f_k$ describes the force that is felt at Body $i$ due to a force $f_k$ at Body $k$.
Intuitively, the relationship between $^kX^T_i$ and $^kX^T_i$ is analogous to the relationship
between $I^C_i$ and $I^A_i$. While $I^C_i$ (Eq. 2.14) is the composite inertia of the subtree
rooted at $i$ which is felt with all the joints locked, $I^A_i$ (Eq. 2.17) is the articulated
inertia which is felt with all the joints free to move.

Just as $^kX_i$ provides a spatial acceleration transform, $^kX_i$ provides an extended
acceleration propagator that transforms an acceleration from Body $i$ to Body $k$ given
that the system is free at the joints. Lilly [71] introduced these transforms over sin-
gle links, and referred to them as articulated transforms to convey their operation.

39
Mathematically, the extended force propagator $k\mathbf{X}_T^i$ was introduced by Chang and Khatib [14] as $i^1L^*$ and by Rodriguez, Jain, and Kreutz-Delgado [112] as $\psi(i, k)$. However, by drawing on their mechanical meaning here, the EFPA is the first algorithm to explicitly compute and use these matrices for efficiency gains.

### 3.3.2 Inward Recursion

The first pass of the EFPA proceeds from tips to base, and is similar in operation to the backwards pass of the Articulated-Body Algorithm (ABA). It seeks to propagate the following equation

$$f_i = I_i^A a_i - \sum_{k \in ES(i)} k\mathbf{X}_T^i f_k^e ,$$

which provides a force-acceleration relationship for the articulated subtree rooted at Body $i$. Here $f_k^e$ represents a force which acts onto end-effector $k$. The articulated transforms $k\mathbf{X}_T^i$ are extended force propagators as described previously and illustrated in Figure 3.1. The inward recursion of the EFPA, used to propagate Eq. 3.12 towards the root of the tree, is more complex in comparison to previous algorithms [14, 112], but enables computational savings in the subsequent outward recursion.

To begin, it is noted that every rigid body in the system must satisfy the dynamic force balance equation, Eq. 3.11, in a more detailed form that that accounts for contributions to the net force $f_i^{net}$ from external forces as well as internal interaction forces. These interactions forces act either between Body $i$ and its predecessor, or between Body $i$ and any of its successor Bodies in $c(i)$ (Eq. 2.1). As a result,

$$f_i = I_i^A a_i - f_i^e + \sum_{j \in c(i)} j\mathbf{X}_i^T f_j ,$$

where $f_i^e$ is the net external force on Body $i$ which satisfies

$$f_{i(k)}^e = k\mathbf{X}_i^{T(k)} f_k^e , \quad \forall k \in \{N_B + 1, \ldots, N_B + m\}$$

40
Figure 3.1: (a) Spatial quantities used to describe the dynamics of Body $i$ in the subtree rooted at $i$. (b) Recursive relationships obtained by the inward recursion of the EFPA. An articulated-body inertia is used to describe the effective inertia of all outlined bodies. The force propagator $k\chi_i^T$ transforms the dynamic effect of the end-effector force $f_k^e$ through bodies $k$ to $i$ to describe the force effect on the $i$-th body.
and \( f^\ell_i = 0 \) otherwise.

Equation 3.13 can be rewritten more generally as

\[
f_i = I_i^A a_i - \sum_{k \in ES(i)} k^{j\mathbf{X}^T} f^\ell_k + \sum_{j \in s(i)} j^{j\mathbf{X}^T} f_j,
\]

(3.15)

where quantities are initialized as \( I_i^A = I_i, s(i) = c(i) \), and \( k^{j\mathbf{X}^T} = 0 \) except for bodies with end-effectors attached, where

\[
k^{j\mathbf{X}^T} = k^{j\mathbf{X}^T}_p(k).
\]

(3.16)

The backward pass of the EFPA successively eliminates the effects of interaction forces \( f_j \) from Eq. 3.15 for its predecessor in exchange for updates to \( k^{j\mathbf{X}^T} \) and \( I_i^A \) that account for the dynamic coupling between the bodies.\(^9\) As a result, the set \( s(i) \) is updated throughout the algorithm to represent the children of Body \( i \) where this coupling has yet to be addressed.

At any Body \( i \) such that \( s(i) = \emptyset \), it is possible to eliminate the interaction force \( f_i \) from the force balance Eq. 3.15 of its predecessor \( p(i) \). The property \( s(i) = \emptyset \) holds for at least one body after initialization of the algorithm, since it holds at any leaf. At any such body, it is possible to compute \( f_i \) as a function of \( a_p(i) \) similar to the derivation of the ABA. First, \( \ddot{q}_i \) is found as a function of \( a_p(i) \). By substituting \( a_i \) from Eq. 3.10 into Eq. 3.15 and multiplying both sides by \( \Phi^T_i \), the following is

\(^9\)Throughout the derivation of the backwards pass of the EFPA, note the similarity to solving an LQ optimal control problem. In the absence of kinematic branching, Eq. 3.10 can be viewed as a linear time-varying discrete-time system with state \( a_i \) and control input \( \ddot{q}_i \). The process of eliminating an interaction force from Eq. 3.15 here is analogous to the process of propagating an optimal-cost-to-go backwards in time as discussed in Section 2.4.1. Through this lens, those familiar with the derivation of LQ optimal control may recognize the steps required here to derive update laws for \( I_i^A \) and \( k^{j\mathbf{X}^T} \).
obtained

\[ \tau_i = 0 = \Phi_i^T f_i \]

(3.17)

\[ = \Phi_i^T \left[ I_i A_i \left( i X_p(i) a_p(i) + \Phi_i \dot{q}_i \right) - \sum_{k \in ES(i)} k \chi_i^T f^e_k \right], \]

(3.18)

which uses the fact that \( \tau_i \) is equal to the component(s) of the interaction force along the free modes of motion (set to zero here). Solving for \( \ddot{q}_i \) provides

\[ \ddot{q}_i = -D_i \Phi_i^T \left( I_i A_i \left( i X_p(i) a_p(i) - \sum_{k \in ES(i)} k \chi_i^T f^e_k \right) \right), \]

(3.19)

where \( D_i = (\Phi_i^T I_i A_i \Phi_i)^{−1} \). Substitution of Eq. 3.19 into Eq. 3.10 gives

\[ a_i = L_i i X_p(i) a_p(i) + K_i \sum_{k \in ES(i)} k \chi_i^T f^e_k, \]

(3.20)

where \( K_i = \Phi_i D_i \Phi_i^T \) and \( L_i \) is an acceleration-propagator across a single joint and is given by

\[ L_i = 1_{6 \times 6} - K_i I_i A_i. \]

(3.21)

This relationship then allows for the interaction force \( f_i \) to be found from Eq. 3.15 for Body \( i \)

\[ f_i = I_i A_i L_i i X_p(i) a_p(i) - L_i^T \sum_{k \in ES(i)} k \chi_i^T f^e_k, \]

(3.22)

where \( L_i^T \) is the force propagator across Joint \( i \). Inserting this equation for \( f_i \) into Eq. 3.15 for Body \( p(i) \) provides the following updates

\[ k \chi_{p(i)}^T \rightarrow i X_{p(i)}^T L_i^T k \chi_i^T, \hspace{1cm} \forall k \in ES(i), \hspace{1cm} \text{and} \]

(3.23)

\[ I_{p(i)}^A \rightarrow I_{p(i)}^A + i X_{p(i)}^T I_i A_i L_i i X_p(i), \]

(3.24)

\[ s \left( p(i) \right) \rightarrow s \left( p(i) \right) \setminus \{i\}. \]

(3.25)
By processing these updates starting from the highest numbered body and counting downward, it follows when Body \( i \) is reached, \( s(i) \) is necessarily empty (since each child is numbered higher than its parent). Overall, Eq. 3.23 is required no more than \( d \) times for each end-effector. This leads to a computational complexity of \( O(md) \) to calculate all required extended force propagators. The articulated inertia computation has complexity \( O(N_B) \), providing a overall complexity of \( O(N_B + md) \) for this pass of the algorithm.

### 3.3.3 Outward Recursion

The final goal of the outward recursion is to construct the inverse of the operational-space inertia matrix \( \Lambda^{-1} \) given in Eq. 3.5. Under the specification of zero velocity and torque, this matrix provides the relationship

\[
\begin{bmatrix}
a_{N_B+1} \\
\vdots \\
a_{N_B+m}
\end{bmatrix}
= \Lambda^{-1}
\begin{bmatrix}
f_{N_B+1}^e \\
\vdots \\
f_{N_B+m}^e
\end{bmatrix}
\tag{3.26}
\]

which relates all of the end-effector forces to all of the end-effector accelerations. Rather than find \( \Lambda^{-1} \) as a whole, the outward recursion here forms each of the blocks of the symmetric matrix \( \Lambda^{-1} \)

\[
\Lambda^{-1} = \begin{bmatrix}
\Lambda^{-1}_{N_B+1,N_B+1} & \cdots & \Lambda^{-1}_{N_B+1,N_B+m} \\
\vdots & \ddots & \vdots \\
\Lambda^{-T}_{N_B+1,N_B+m} & \cdots & \Lambda^{-1}_{N_B+m,N_B+m}
\end{bmatrix},
\tag{3.27}
\]

where each \( \Lambda^{-1}_{k_1,k_2} \) relates an end-effector force \( f_{k_2}^e \) to an end-effector acceleration \( a_{k_1} \).

Each \( \Lambda^{-1}_{k_1,k_2} \) is arrived at recursively, from base to tips, through the solution for accelerations at intermediate links along the tree. This is accomplished through propagation of the following equation outward

\[
a_i = \sum_{k=N_B+1}^{N_B+m} \Lambda^{-1}_{k,i} f_k^e,
\tag{3.28}
\]

44
where each $\Lambda_{ik}^{-1}$ similarly describes the acceleration at Body $i$ caused by end-effector force $f_k^e$. It is noted that $\Lambda_{ik}^{-1}$ is not a block of $\Lambda^{-1}$ when $i$ refers to a body. The assumption that the inertial coordinate frame is stationary seeds this recursion with

$$\Lambda_{0k}^{-1} = 0_{6 \times 6}, \quad \forall k \in \{N_B + 1, \ldots, N_B + m\}. \quad (3.29)$$

To propagate Eq. 3.28 to Body $i$ from its predecessor, the combination of Eq. 3.28 at $p(i)$ along with Eq. 3.20 leads to

$$a_i = \sum_{k \notin ES(i)} L_i^i X_p(i) \Lambda_{p(i)k}^{-1} f_k^e + \sum_{k \in ES(i)} \left( L_i^i X_p(i) \Lambda_{p(i)k}^{-1} + K_i^k \mathcal{X}_i^T \right) f_k^e.$$

Thus, the recursive relationship between $\Lambda_{ik}^{-1}$ and $\Lambda_{p(i)k}^{-1}$ is dependent on whether or not Joint $i$ supports end-effector $k$. For each $k \in ES(i)$ the update law

$$\Lambda_{ik}^{-1} \rightarrow L_i^i X_p(i) \Lambda_{p(i)k}^{-1} + K_i^k \mathcal{X}_i^T, \quad (3.30)$$

is required, while

$$\Lambda_{ik}^{-1} \rightarrow L_i^i X_p(i) \Lambda_{p(i)k}^{-1}$$

for each $k \notin ES(i)$. That is, if $i$ does not support end-effector $k_2$, the acceleration influence of $f_{k_2}^e$ propagates from $p(i)$ to $i$ according to an articulated transform (as if Joint $i$ were free to move). Following this relationship through the tree, the acceleration influence of $f_{k_2}^e$ will continue to propagate outward according to this articulated transform for all bodies in $c^*(i)$. As a result, for any other end-effector $k_1$ in the subtree at $i$

$$\Lambda_{k_1 k_2}^{-1} = k_1 \mathcal{X}_p(i) \Lambda_{p(i)k_2}^{-1}. \quad (3.31)$$

This simplification is illustrated in Figure 3.2 for a basic example. The simplification of transforming $\Lambda_{ik_2}^{-1}$ directly to $\Lambda_{k_1 k_2}^{-1}$ is first possible when $i = \text{ancest}(k_1, k_2)$.
Figure 3.2: Simplification to the outward recursion that is provided by the acceleration propagator $k_1 \mathbf{x}_1$. Its transpose, the force propagator $\mathbf{x}^T_{k_1}$, was calculated during the inward recursion. As a result, three recursive steps for the calculation of $\Lambda_{k_1 k_2}^{-1}$, in the example here, are able to be replaced with one matrix multiplication.
In the example figure, an acceleration propagator may be applied to calculate \( \Lambda^{-1}_{k_1k_2} \) recursively through links 1, 2, and 3. Yet, computational savings can occur when \( \Lambda^{-1}_{k_1k_2} \) is computed directly at the common ancestor of \( k_1 \) and \( k_2 \) through the use of \( k_1 x_1 \). With this insight the following set is defined

\[
GCA(i) = \{(k_1, k_2) \mid k_1 < k_2 \text{ and } i = \text{ancest}(k_1, k_2)\}
\]  

(3.32)

which contains all end-effector pairs that have a greatest common ancestor at \( i \). The EFPA uses these sets to determine the links at which the cross-terms of \( \Lambda^{-1} \) may be computed. Through the restriction that \( k_1 < k_2 \) in Eq. 3.32, this definition also limits the computation of the blocks of \( \Lambda^{-1} \) to those in the upper triangle, since the symmetry of \( \Lambda^{-1} \) provides

\[
\Lambda^{-1}_{k_2, k_1} = (\Lambda^{-1}_{k_1, k_2})^T.
\]

Through the use of these simplifications, the recursion in Eq. 3.30 is required a maximum of \( d \) times for each end-effector. This process ultimately leads to the computation of each \( \Lambda^{-1}_{kk} \) and has overall complexity of \( O(md) \). These quantities represent the diagonal blocks of \( \Lambda^{-1} \). Each of the cross terms in \( \Lambda^{-1} \) can then be calculated via an appropriate version of Eq. 3.31, resulting in a cross-term calculation complexity of \( O(m^2) \). Thus, the overall complexity of this recursion is \( O(md + m^2) \). Including both passes of the algorithm, a complexity of \( O(N_B + md + m^2) \) is achieved.

The full EFPA is listed in Table 3.1. The second and third loops provide implementations of the inward and outward recursions respectively. The final loop provides the necessary transformation for each \( \Lambda^{-1}_{kk} \) into end-effector coordinates. This additional transformation loop is not required for off-diagonal terms, as the acceleration propagators \( k x_l \) transform accelerations directly to the end-effector frames.
Initialize: $\Lambda_{0k}^{-1} = 0_{6 \times 6}$, $I_i^A = I_i$

for $k = N + 1$ to $N + m$ do

$kX_p^T = kX_p^T$

end for $k$

for $i = N$ to 1 do

$D_i = (\Phi_i^T I_i^A \Phi_i)^{-1}$

$K_i = \Phi_i D_i \Phi_i^T$

$L_i = 1_{6 \times 6} - K_i I_i^A$

if $p(i) \neq 0$ then

$I_{p(i)}^A = I_{p(i)}^A + iX_p(i) I_i^A L_i X_p(i)$

for all $k \in ES(i)$ do

$kX_p^T = iX_p(i) L_i^T kX_i^T$

end for $k$

end

end for $i$

for $i = 1$ to $N$ do

for all $k \in ES(i)$ do

$\Lambda_{ik}^{-1} = L_i iX_p(i) \Lambda_{p(i)k}^{-1} + K_i kX_i^T$

end for $k$

for all $(k_1, k_2) \in GCA(i)$ do

$\Lambda_{k_1 k_2}^{-1} = k_1 X_i \Lambda_{i k_2}^{-1}$

end for $k$

end for $i$

for $k = N + 1$ to $N + m$ do

$\Lambda_{kk}^{-1} = kX_p(k) \Lambda_{p(k)k}^{-1}$

end for $i$

$\Lambda^{-1} = \begin{bmatrix}
\Lambda_{N+1, N+1}^{-1} & \cdots & \Lambda_{N+1, N+m}^{-1} \\
\vdots & \ddots & \vdots \\
\Lambda_{N+1, N+m}^{-1} & \cdots & \Lambda_{N+m, N+m}^{-1}
\end{bmatrix}$

Table 3.1: Extended-Force-Propagator Algorithm
3.4 Algorithm Comparison

This section provides a comparison of the computational performance of the EFPA to the algorithms of Rodriguez, Jain, and Kreutz-Delgado (RJK) [112] and Featherstone [31]. These competing algorithms represent the most efficient previously published algorithms for the Operational-Space Inertia Matrix. The RJK algorithm is the second lowest-order algorithm available, behind the EFPA. However, as low order does not imply low computational requirements, the algorithm of Featherstone [31] has been shown to outperform the RJK algorithm, despite its higher computational complexity. The operation of these algorithms is first briefly described to aid the discussion of their relative performance.

3.4.1 RJK Algorithm

The RJK algorithm (RJKA) is described in [112] with a dramatically optimized version given in [31]. The main differences between the RJKA and EFPA can be summarized as follows.

- Only the EFPA calculates extended force propagators.
- The RJKA calculates $\Lambda^{-1}_{kk}$ via a recursion that calculates $\Lambda^{-1}_{ii}$ from $\Lambda^{-1}_{p(i)p(i)}$, whereas the EFPA instead calculates $\Lambda^{-1}_{ik}$ from $\Lambda^{-1}_{p(i)k}$.
- The RJKA calculates $\Lambda^{-1}_{k_1k_2}$ recursively from a common ancestor using a series of articulated transforms across single joints, whereas the EFPA calculates these quantities in a single step using Eq. 3.31.

Appendix B provides further details on the RJKA and includes the full algorithm using the notation of this dissertation.
3.4.2 Sparse Factorization Algorithm (SFA)

Featherstone’s sparse factorization algorithm (SFA) [31] is based on the exploitation of branch-induced sparsity in the system mass matrix $H$ and the rows of the task Jacobian $J$. The algorithm first uses the factorization approach of Featherstone [29] to express the mass matrix as $H = L^T L$, where $L$ is a lower-triangular matrix that enjoys the same sparsity pattern as the lower triangle of $H$. The class of lower-triangular matrices that possesses this branch-induced sparsity pattern is shown to be a group, which implies the same sparsity pattern in $L^{-1}$, and thus in the rows of $Y = JL^{-1}$. This definition provides

$$\Lambda^{-1} = JH^{-1}J^T = YY^T.$$  

The properties of branch-induced sparsity are then exploited to greatly accelerate the computation of $Y$ and $\Lambda^{-1}$. The final algorithm to compute $\Lambda^{-1}$ can be shown to have computational complexity $O(nd^2 + md^2 + m^2 d)$, where again $n$ is the total number of DoFs, $d$ is the depth of the connectivity tree of the system, and $m$ is the number of end-effectors.

3.4.3 Computational Examples

To understand the comparative performance of the algorithms, this section presents the floating point operation counts (flops) for the calculation of $\Lambda^{-1}$ in a number of examples. The examples considered are mainly the same as those in [31] for the ASIMO next-generation humanoid robot and derived mechanisms. This floating-base humanoid consists of $N_B = 35$ bodies, with connectivity tree shown in Figure 3.3. All joints except the floating base are modeled as revolutes.
The first example considered is an operational space consisting of the position and orientation of the hands and feet (4 end-effectors) for this system. Figure 3.4 provides a breakdown of the cost to calculate $\Lambda^{-1}$ for each algorithm. Further details on the assessment of computational costs for the RJKA and EFPA are provided in Appendix B.

This example highlights the advantages afforded by the computation of the extended-force-propagators over previous recursive algorithms. In comparison to the RJKA, the EFPA incurs additional cost during the inward recursion to obtain each $^k\mathbf{x}_i$. Yet, this enables significant savings in the computation of the cross-terms of $\Lambda^{-1}$, which is by far the most expensive step in the RJKA. The sparsity of the system mass matrix still allows the SFA to obtain $\Lambda^{-1}$ with 78 percent of the flops when compared to the EFPA. Approximately 67 percent of the elements of $\mathbf{J}$ and 56 percent of the elements of $\mathbf{H}$ are zero in this case.

The next series of examples show the benefits of the reduced order EFPA for higher DoF systems that lack a high-degree of sparsity in their mass matrix. Figure 3.5 shows the computational costs for a series of alterations to the ASIMO Next-Generation mechanism. The examples considered include additional DoFs in the appendages, additional appendages, and the modification to 12-DoF four-fingered hands. The
Figure 3.4: Cost comparison breakdown for the Hands and Feet operational-space example for Figure 3.3. Although the EFPA incurs additional cost on the inward recursion to calculate each $k \chi_i$, this enables savings to calculate the off-diagonal terms of $\Lambda_{k_1 k_2}^{-1}$. Still, the sparsity from this topology provides advantage to the SFA.

Figure 3.5: Cost comparison for operational-space algorithms. Operational spaces are abbreviated as: HF = Hands and Feet; HFH = Hands, Feet, and Head; FTF = Fingers, Thumbs, and Feet. This figure shows the benefits of the EFPA for systems with less sparsity in the mass matrix.
EFPA provides advantages over the other recursive algorithm (RJKA) in every case. This is largely the result of RJKA’s high calculation cost for the cross-terms of $\Lambda^{-1}$.

The modifications of additional DoFs in the arms, legs, or hands adversely effect the sparsity of the system’s mass matrix and thus provide advantages for the EFPA over the SFA. The addition of extra DoFs in any of the appendages lengthens the unbranched chains in the mechanism’s connectivity tree. These unbranched chains lead to fully dense blocks in the system mass matrix, reducing the benefits of sparse techniques. Thus, the SFA exhibits a lower relative efficiency in these cases.

The EFPA is outperformed when 2 arms and 2 legs are added, an unexpected result for the low-order algorithm. This is mainly a result of the high-degree of sparsity in the mass matrix for this mechanism. The mass matrix has 73 percent zero elements, providing advantages to the SFA. Despite this sparsity increase, the cost ratio to the EFPA is nearly the same as in the base example. This property is not shared by the RJKA, whose cost ratio to the SFA increases over 50 percent in comparison to the base example.

The fore-fingers, thumbs, feet (FTF) operational-space example does highlight a potential improvement for the EFPA. Considering the fore-finger and thumb end-effectors for the left hand, these end-effectors have a long, common, support chain (from the base to the left hand). As a result, force propagators for these end-effectors have much common structure. This fact is not currently exploited, and provides opportunity for algorithmic improvements in similar cases. The final example is another case that would benefit from this type of improvement. In this example, the humanoid is balanced on one foot, which is treated as a fixed-base, with the hands and free foot as the operational space. Here, the links between the torso and grounded
foot provide a common support chain that could be exploited in the computation of many of the extended-force-propagators. This current drawback of our algorithm, repeated propagation calculations over common support chains, is not shared with the RJKA, providing advantage to it in this and similar examples.

3.5 Summary

This chapter has detailed the Extended-Force-Propagator Algorithm, the lowest-order algorithm to date for the computation of $\Lambda^{-1}$. The algorithm has been shown to provide computational efficiency benefits over the optimized RJK algorithm [31, 112], the previous benchmark for recursive $\Lambda^{-1}$ algorithms. The recursive approach is able to maintain efficient computation for systems that lack sparsity, providing benefits over the sparse techniques of Featherstone [31] for some topologies. These computational benefits have been enabled by the use of extended-force-propagators, which provide articulated transformations of spatial quantities over spans of links. This represents the first time these quantities have been applied in recursive dynamics algorithms. While this chapter has provided algorithms to form an important quantity for task-space dynamics, Chapters 4-7 will make extensive use of task-space control to provide real-time whole-body humanoid control.
Chapter 4

A Real-Time Conic Formulation of Prioritized Task-Space Control

4.1 Introduction

This chapter describes a new formulation of the task-space control (TSC) problem as a conic optimization problem that is able to be solved at real-time rates. Task-space (or operational-space) control [60, 101], which relies on the operational-space dynamics formulation described in Chapter 3, provides a framework that significantly eases the burden of authoring whole-body trajectories. Rather than design or optimize motions in the high-dimensional configuration space of a humanoid, TSC instead allows task-space trajectories to be designed in intuitive motion spaces. For instance, walking motions can be specified through design of foot and center of mass trajectories. Without the need for any inverse kinematics, online TSC has been shown to provide an elegant solution to produce whole-body walking behaviors [18].

Although TSC significantly eases the design of motions, it is difficult to perform at real-time rates, since the many joint actuators of the humanoid must be coordinated together to realize the task-space goals. This process is complicated by kinematic and dynamic constraints, such as the need for contact at the stance feet and restriction
of contact forces to be within their frictional limits. Further, it is often desirable to define a strict hierarchy of importance among the task goals. Within this context, the task-space control problem has to be solved sequentially, in order of task importance, with higher priority tasks treated as additional constraints at each subsequent priority level. Methods that define such a task hierarchy are said to perform Prioritized Task-Space Control (PTSC), which is applied here.

PTSC has been applied to generate whole-body humanoid behaviors in a number of previous papers. Projection methods of Park and Khatib [101] and Sentis et al. [121] allow task priorities to be enforced and effectively handle kinematic support foot constraints. Frictional and unidirectional limits on ground reaction forces (GRFs) are not enforced, however, which can lead to balance failure if the desired task dynamics are not carefully selected. Just as a motion filter can be applied to balance pre-designed joint trajectories that violate contact force constraints (such as ZMP), similar filters can be applied to physically unrealizable task-space trajectories. For example de Lasa et al. [18] and Salini et al. [116] both use a series of quadratic programs (QPs) to solve PTSC while satisfying GRF constraints. The new PTSC formulation presented in this chapter is inspired by their work and reformulates their QPs into a sequence of conic optimization problems which can be solved at real-time rates. This new conic formulation of PTSC is able to address full friction cone constraints at on the GRFs, and is shown to perform twice as fast as previous formulations. This improvement allows high-bandwidth control of highly dynamic movements.

To demonstrate the performance of PTSC, this approach is applied to generate dynamic behaviors such as the kick and jump shown in Figure 4.1. The common control structure used to create these behaviors is shown in Figure 4.2. A high-level state
A new formulation of the Prioritized Task-Space Control problem allows us to control dynamic behaviors such as a kick and a jump at real-time rates and in challenging environments. The use of centroidal momentum control results in rich emergent arm motions to maintain balance without any upper-body motion authoring.

Machine is used to manage the desired task dynamics, while a new conic optimization formulation of PTSC is used to select joint torques at every control step. Through the formulation of PTSC described here, the control loop is able to be closed at rates of approximately 200 Hz on a commodity laptop, which is approximately double that of previous formulations. Generated motions automatically satisfy the ZMP constraint on level terrain and are general to produce balanced motions on uneven surfaces. Additionally, centroidal momentum control \cite{95, 97} is used to alleviate the need to design specific upper-body trajectories, which instead emerge from our control strategies.
4.2 Task-Space Control for Humanoids

Throughout the remainder of the dissertation, the humanoid is considered under the dynamics equations of motion, Eq. 2.15, reiterated below

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

(4.1)

where \( J_s \in \mathbb{R}^{6N_s \times n} \) is a support Jacobian that maps joint velocities \( \dot{q} \) to the combined spatial velocity of the \( N_s \) feet in contact. As such, \( F_s \in \mathbb{R}^{6N_s} \) collects the spatial

---

**Figure 4.2:** Block diagram for behavioral controllers used in this work. A high-level state machine manages the commanded task dynamics \( \dot{r}_{t,c} \) and priorities at each instant. A prioritized task-space controller then finds joint torques \( \tau \) which keep feet planted and prevent slipping while realizing the desired task dynamics. This control loop is closed at real-time rates of approximately 200 Hz.

The remainder of the chapter is organized as follows. Section 4.2 describes previous approaches to TSC and PTSC and describes how the control of centroidal momentum can be integrated into these frameworks. Section 4.3 then details the new conic formulation of PTSC and describes the benefits over previous approaches. Section 4.4 presents the application of PTSC to the control of a dynamic kick and a standing jump. Finally, Section 4.5 reviews the chapter and summarizes its contribution.
ground reaction forces (GRFs) on the humanoid

\[ \mathbf{F}_s = \begin{bmatrix} \mathbf{f}_{s1}^T & \cdots & \mathbf{f}_{sN_s}^T \end{bmatrix}^T \]  

(4.2)

where \( \mathbf{f}_{si} \in \mathbb{R}^6 \) represents the spatial ground reaction force onto the \( i \)-th body in contact.

In order to author whole-body behaviors, it is often convenient to characterize the system’s desired dynamics in a task (or operational) space [60]. Chapter 3 presented the operational-space dynamics in the special case that the task space was comprised of a set of \( m \) end-effectors. Task-space control may be applied more generally to control any features \( \mathbf{x}_t \) of the configuration variables

\[ \mathbf{x}_t = \mathbf{g}(\mathbf{q}) . \]  

(4.3)

Task velocities \( \dot{\mathbf{x}}_t \) are then related to joint rates \( \dot{\mathbf{q}} \) by the standard relationship

\[ \dot{\mathbf{x}}_t = \mathbf{J}_t(\mathbf{q}) \dot{\mathbf{q}} \]  

(4.4)

where \( \mathbf{J}_t(\mathbf{q}) = \frac{\partial \mathbf{g}}{\partial \mathbf{q}} \) is a task Jacobian. Alternatively, a task may not arise from a Jacobian relationship. To accommodate such a generalization, this definition of a task is relaxed to any relationship of the form

\[ \mathbf{r}_t = \mathbf{A}_t(\mathbf{q}) \dot{\mathbf{q}} . \]  

(4.5)

Most notably, this generality can be exploited to describe the system’s centroidal momentum \( \mathbf{h}_G \), introduced in Section 2.4.2. Although the centroidal momentum matrix relationship Eq. 2.22 looks like a Jacobian relationship, the system’s net angular momentum \( \mathbf{k}_G \) does not admit any function \( \mathbf{g}(\mathbf{q}) \) with Jacobian \( \mathbf{J}_g = \frac{\partial \mathbf{g}}{\partial \mathbf{q}} \) such that \( \mathbf{k}_G = \mathbf{J}_g \dot{\mathbf{q}} \). This result is due to the fact that the conservation of angular momentum imposes a nonholonomic constraint [92].
Given a general set of tasks along with commanded instantaneous task dynamics \( \dot{\mathbf{r}}_{t,c} \) the task-space control problem is to find joint torques \( \mathbf{\tau} \) that result in joint accelerations \( \ddot{\mathbf{q}} \) with

\[
A_t \ddot{\mathbf{q}} + \dot{A}_t \dot{\mathbf{q}} = \dot{\mathbf{r}}_t
\]

such that \( \dot{\mathbf{r}}_t \) most closely matches \( \dot{\mathbf{r}}_{t,c} \). Depending on the choice of \( \dot{\mathbf{r}}_{t,c} \), additional freedoms may be used to match lower-priority task dynamics. For systems in support, there are a variety of ways to solve this problem, which differ in how they account for the presence of external forces \( \mathbf{F}_s \) in Eq. 4.1. These different methods are discussed in the following subsections.

### 4.2.1 Projection Methods for Task-Space Control During Support

In periods of support, the equations of motion, Eq. 4.1, are often considered under the constraint of zero acceleration at the support bodies

\[
\ddot{x}_s = J_s \ddot{\mathbf{q}} + \dot{J}_s \dot{\mathbf{q}} = 0.
\]

Taking the feet as a set of \( N_s \) end-effectors, Eq. 3.1 and Eqs. 3.6-3.8 can be used to find the support forces \( \mathbf{F}_s \) that would be required to ensure Eq. 4.7

\[
\mathbf{F}_s = \mu_s + \mathbf{\rho}_s - \mathbf{\Lambda J H}^{-1} S_a^T \mathbf{\tau}
\]

\[
= \mathbf{\Lambda J H}^{-1} (C \dot{\mathbf{q}} + \mathbf{G} - S_a^T \mathbf{\tau}) - \mathbf{\Lambda J} \dot{\mathbf{q}}.
\]

Under the assumption of sufficient friction and unidirectional forces, these support forces can be substituted into Eq. 4.1 to produce a set of constrained dynamic equations of motion

\[
\mathbf{H} \ddot{\mathbf{q}} + N_s^T (C \dot{\mathbf{q}} + \mathbf{G}) + \gamma(q, \dot{q}) = N_s^T S_a^T \mathbf{\tau}
\]
where
\[ \gamma = J_s^T \left( J_s H^{-1} J_s^T \right)^\dagger J_s \dot{q} \]  
(4.11)
is a velocity dependent term due to the constraints and
\[ N_s^T = 1 - J_s^T \left( J_s H^{-1} J_s^T \right)^\dagger J_s H^{-1} \]  
(4.12)
is the dynamically-consistent nullspace projector for support [101]. The symbol \( (\cdot)^\dagger \) denotes the Moore-Penrose pseudo-inverse of the enclosed matrix. In a rough sense, these equations of motion are a projection of the original equations of motion onto the subset of the configuration space that is consistent with the support constraints. Since no constraints are placed on the GRFs, these dynamic equations are in fact not correct if torques are supplied that would lift the foot off the ground, or cause the foot to slip.

Starting from Eq. 4.10 the work of Park and Khatib [101] and Sentis et al. [121] extend the traditional prioritized operational-space control framework for manipulators to the case of humanoids in support. Just as in the above derivation, all their control laws are predicated on the assumption that given a joint input \( \tau \), the ground is capable of supplying support forces \( F_s \) to ensure the foot remains stationary. This assumption is powerful, and often valid on level terrain since GRFs are dominated by a gravity compensation force. Still, for highly dynamic motions, additional care has to be taken to ensure that the commanded task dynamics do not require GRFs outside their unidirectional or frictional boundaries. This becomes increasingly difficult on more challenging terrains. Within their approach, task prioritization is handed through a nested series of task nullspace projectors [121].
4.2.2 Ground Reaction Force Constraint Modeling

To account for the true constraints on ground reaction forces (GRFs) during the selection of joint torques, others ([18, 78, 116, 117]) have proposed the use of constrained quadratic programming to solve the PTSC problem. These approaches treat $F_s$ as a control variable and optimize the selection of $\tau$ and $F_s$ under appropriate constraints. In order to approximate the constraints on each net spatial foot force $f_s \in \mathbb{R}^6$, it is customary [142] to represent each $f_s$ as a combination of pure forces that act at the corners of the feet, as shown in Figure 4.3. Unidirectional and frictional constraints can then easily be enforced on the individual vertex forces $f_{sij} \in \mathbb{R}^3$ shown in this figure, where $j \in \{1, \ldots, N_{Pi}\}$ and $N_{Pi}$ is the number of contact vertices for foot $i$. Through this simplification, any force underneath the foot which satisfies friction can be represented by a statically equivalent set of vertex forces which all satisfy friction.\(^{10}\) Given a coefficient of friction $\mu_i$ for foot $i$, these vertex forces must reside inside a friction cone

$$C_i := \left\{ (f_x, f_y, f_z) \in \mathbb{R}^3 \mid \sqrt{f_x^2 + f_y^2} \leq \mu_i f_z \right\}. \quad (4.13)$$

To simplify optimization, previous work has approximated friction cones by a friction pyramid $P_i \subset C_i$ as shown in Figure 4.3.

Despite only representing pure forces at the vertices here, this formulation also captures frictional limits on the component of the net rotational moment created by the foot that is normal to the local plane of contact. When the center of pressure (CoP) is within the center of the foot, the pressure distribution beneath the foot

\(^{10}\)In fact, any pure force $f$ at position $p$ underneath the foot can be represented by parallel vertex forces. To see this, let the vertex positions for foot $i$ be defined as $p_{ij}$, with the position $p$ given as a convex combination of vertex positions $p = \sum_{j=1}^{N_{Pi}} \alpha_{ij} p_{ij}$, ($\alpha_{ij} \geq 0$, $\sum_{j=1}^{N_{Pi}} \alpha_{ij} = 1$). The net effect of individual vertex forces selected as $f_{sij} = \alpha_{ij} f$ is then statically equivalent to the original force.
has significant authority to create a normal moment about the CoP. However, when the CoP is at a contact vertex, the pressure distribution becomes singular in that all of the force below the foot exists at the contact vertex. In this case, the pressure distribution is unable to create a normal moment below the foot. As a result, methods that directly optimize foot wrenches have difficulty to place appropriate bounds on the normal moments, as limitations are CoP dependent. By representing the spatial force on each foot through a combination of vertex forces, this limitation is captured inherently, without having to rely on other heuristic models to limit normal moments [154].

Given the breakdown of the spatial foot forces into forces which act at the contact vertices, Eq. 4.1 takes the more detailed form

\[
H(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = S_a^T \tau + J_{sp}(q)^T F_{sp}
\]

(4.14)

where \( F_{sp} \) collects all support forces \( f_{s_{ij}} \) and \( J_{sp} \) is the combined support point Jacobian. This support point Jacobian relates \( \dot{q} \) to the linear velocity of all the support vertices and differs from the support Jacobian \( J_s \) which related \( \dot{q} \) to the spatial velocity of all support bodies.

4.2.3 Quadratic Programming Methods for Task-Space Control During Support

Previous work [18, 117] has formulated a quadratic program (QP) to find contact forces, joint accelerations \( \ddot{q} \), and joint torques \( \tau \) that are consistent with the dynamic equations of motion, and most closely match the commanded task dynamics. An
Figure 4.3: The spatial contact force $\mathbf{f}_i \in \mathbb{R}^6$ on foot $i$ is approximated by a distribution of pure forces (no moment) to a lattice of contact vertices underneath the feet. These forces are denoted as $\mathbf{f}_{si} \in \mathbb{R}^3$ in the text. Forces are constrained to lie inside the friction cone $\mathbf{C}_i \subset \mathbb{R}^3$ (shown in blue), or in a more restrictive friction pyramid $\mathcal{P}_i \subset \mathbb{R}^3$ (shown in gray) as in previous work.

Example of such a QP is given below

$$
\min_{\mathbf{q}, \mathbf{\tau}, \mathbf{F}_{sp}} \frac{1}{2} \left\| \ddot{\mathbf{r}}_{t,c} - (A_t \dot{\mathbf{q}} + \dot{A}_t \mathbf{q}) \right\|^2
$$

subject to

$$
H \ddot{\mathbf{q}} + C \dot{\mathbf{q}} + G = S_a^T \mathbf{\tau} + J_{sp}^T \mathbf{F}_{sp}
$$

$$
\mathbf{f}_{si} \in \mathcal{P}_i \quad \forall i \in \{1, \ldots, N_S\}, \forall j \in \{1, \ldots, N_{P_i}\}
$$

$$
J_s \ddot{\mathbf{q}} + \dot{J}_s \dot{\mathbf{q}} = 0
$$

$$
\mathbf{\tau} \leq \mathbf{\tau} \leq \mathbf{\tau}^T,
$$

which seeks to most closely match the desired task dynamics while obeying many dynamic and kinematic constraints. Joint torque limits are described by the vectors $\mathbf{\tau}$ and $\mathbf{\tau}^T$. The unidirectional constraints also imposed on the GRFs through Eq. 4.16, combined with the support acceleration constraint Eq. 4.17 assure that the optimized $\ddot{\mathbf{q}}$ satisfies the ZMP constraint.
To optimize for lower-priority tasks, the QP may be modified with additional constraints included to ensure that high-priority task dynamics are not corrupted. This nested series of QPs (or stack of QPs) replaces the nested series of nullspace projectors that are required for projection-based methods. In the work of Mansard [78], an elimination of variables is employed that reduces the size of the above QP at each step, but this reduction is specific to periods of support. In the next section, a reduction is proposed that is general to periods of flight as well as support.

4.3 Conic Formulation of Prioritized Optimization

This section provides a reformulation of the PTSC problem as a conic optimization problem. This reformulation results in a reduced number of variables, reduced number of constraints, and is general to periods of flight as well as support. This reduction allows our reformulation to be solved about twice as fast as Eq. 4.15 for the examples considered. Suppose that the task matrix $A_t$ contains all possible tasks, regardless of priority. The error $e$ in achieving the commanded task dynamics is given by

$$e = \dot{r}_{t,c} - \dot{r}_t$$

$$= \dot{r}_{t,c} - A_t \ddot{q} - \dot{A}_t \dot{q}.$$  \hfill (4.19)

$$= \dot{r}_{t,c} - A_t \ddot{q} - \dot{A}_t \dot{q}.$$  \hfill (4.20)

Solving Eq. 4.14 for $\ddot{q}$, the task dynamics error $e$ obeys:

$$A_t H^{-1} S_a^T \tau + A_t H^{-1} J_{sp}^T F_{sp} + e = b_t$$

$$= \dot{r}_{t,c} + A_t H^{-1} (C \dot{q} + G) - \dot{A}_t \dot{q}.$$  \hfill (4.21)

where

$$b_t = \dot{r}_{t,c} + A_t H^{-1} (C \dot{q} + G) - \dot{A}_t \dot{q}.$$  \hfill (4.22)
In Eq. 4.21, the quantities

\[ \Lambda^{-1}_t = A_t H^{-1} S_a^T \] \hspace{1cm} \text{and} \hspace{1cm} \tag{4.23} 

\[ \Lambda^{-1}_{ts} = A_t H^{-1} J_{sp}^T \] \hspace{1cm} \tag{4.24} 

are defined with symbols \( \Lambda^{-1} \) since each of these quantities is a cross-coupling inverse operational-space (task-space) inertia matrix, similar to that in Chapter 3 and [139], which relate forces applied for one task to the dynamics of another task.

Here, the PTSC problem with \( K \) levels of prioritization is solved with a series of \( K \) conic optimization problems. At each level \( k \in \{1, \ldots, K\} \) it is assumed that the subset of the tasks to be optimized is encoded in a task optimization selector matrix \( S_{o,k} \). Each row of \( S_{o,k} \) is a unit vector that selects a single task error, and the number of rows corresponds to the number of tasks concurrently optimized at level \( k \). As an example, if the desired tasks were the centroidal angular and linear momentum along with the linear velocity of the left foot, the task vector \( r_t \) would take the form

\[ r_t = \begin{bmatrix} k_G \\ l_G \\ v_5 \end{bmatrix}. \] \hspace{1cm} \tag{4.25} 

If the centroidal linear momentum were the top priority, with the centroidal angular momentum and foot velocity as the second priority, the task selectors \( S_{o,1} \) and \( S_{o,2} \) would take the forms

\[ S_{o,1} = \begin{bmatrix} 0_{3 \times 3} & 1_{3 \times 3} & 0_{3 \times 3} \\ 1_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} \end{bmatrix} \] \hspace{1cm} \text{and} \hspace{1cm} \tag{4.26} 

\[ S_{o,2} = \begin{bmatrix} 1_{3 \times 3} & 0_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & 0_{3 \times 3} & 1_{3 \times 3} \end{bmatrix}. \] \hspace{1cm} \tag{4.27} 

At each priority level \( k \), the optimal error from the previous level is defined as \( e_{k-1}^* \), with \( e_0^* = 0 \). A similar task constraint selector \( S_{c,k} \) selects those tasks from all
previous levels, as well as any additional hard constraints. As a result, $S_{c,1}$ selects the hard task constraints imposed across all priority levels. The problem for priority level $k$ is then

$$
\begin{align*}
\min_{\tau, F_{sp}, e_k, z} & \quad z \\
\text{s.t.} & \quad \Lambda_{t\tau}^{-1} \tau + \Lambda_{t\tau}^{-1} F_{sp} + e_k = b_t \\
&& \quad \|S_{o,k} e_k\| \leq z \\
&& \quad S_{c,k} e_k = S_{c,k} e_{k-1} \\
&& \quad f_{s_{ij}} \in C_i \quad \forall i \in \{1, \ldots, N_s\}, \forall j \in \{1,\ldots,N_p\} \\
&& \quad \tau \leq \tau \leq \bar{\tau}.
\end{align*}
$$

Minimization of the scalar $z$ in Eq. 4.28 results in a minimization of error for the current task dynamics to be optimized due to constraint Eq. 4.29. The incorporation of $z$ also provides a linear objective, which is required for the conic solver used here. The constraint Eq. 4.30 ensures that the optimal errors for the higher priority tasks are not corrupted. Table 4.1 describes the entire algorithm for this series of optimization problems in more explicit detail. Note that for application to position controlled systems, an optimal $\ddot{q}$ can be obtained by solving Eq. 4.14 or through a forward dynamics computation.

The advantages of this strategy over the QP in Eq. 4.15 is multi-pronged. First, the variables $\ddot{q}$ are eliminated in favor of task error $e$. Since, at each level, only a few tasks are optimized, this reduces the number of optimization variables, and simplifies the objective. Salini [117] eliminates $\ddot{q}$ in an alternative QP formulation, but does not include $e$, resulting in a dense objective function Hessian. As a result, the formulation of Salini [117] and the QP here were found to perform comparably.
Inputs:

Task Dynamics Descriptors: $A_t, \dot{A}_t, \dot{q}_t, \dot{r}_{t,c}$
System Dynamics Descriptors: $H, Cq, G, S_o, J_{sp}$
Task Priority Descriptors: $S_{c,1}, K$
$S_{o,k} \forall k \in \{1, \ldots, K\}$

Algorithm:

$e_0^* := 0$
$\Lambda_{tr}^{-1} := A_tH^{-1}S_a^T$
$\Lambda_{ts}^{-1} := A_tH^{-1}J_{sp}^T$
$b_t := \dot{r}_{t,c} + A_tH^{-1}(C\dot{q} + G) - \dot{A}_t\dot{q}$

for $k = 1$ to $K$ do

$(\tau^*, F_{sp}^*, e_k^*, z^*) := \arg\min_{\tau, F_{sp}, e_k, z} (Eqs. 4.28 - 4.32)$
$S_{c,(k+1)} := [S_{c,k}^T S_{o,k}^T]^T$
end for $k$

Output:

$F_{sp} = F_{tp}^*$

Table 4.1: Prioritized Task-Space Control (PTSC) Algorithm
Second, polygonal approximations to the friction cones, which grow in complexity as the fidelity of the approximation is increased, are replaced by a single constraint per force $f_{s_{ij}}$. For a 4-sided polygonal approximation, the use of friction cones was not found to have any substantial effect on the computation times or simulation results. Beyond a 4-sided polygonal approximation, it was found to be advantageous to use cone constraints to enable faster solutions. Solution of all optimization formulations was provided by the interior-point optimizer in MOSEK [3], which employs the primal-dual method of Nesterov and Todd [94]. This method handles the cone constraints Eq. 4.29 and Eq. 4.31 directly and efficiently through generalizations of efficient interior-point algorithms for linear programming [58] to the setting of conic optimization.

4.4 Application to the Control of Dynamic Behaviors

This section describes the application of PTSC to the control of a dynamic kick and a standing jump. Section 4.4.1 describes application of PTSC to a dynamic kick, and outlines how the task-space control loop is closed in order to construct the commanded task dynamics $\dot{r}_{t,c}$. Section 4.4.2 describes application to a jump and shows how the contact force optimization is general to handle cases of uneven terrain. Through both examples, centroidal momentum control is a key component, which results in emergent upper-body motions.

These results were completed with an early, scaled version of the model presented in Chapter 2. Here the model stands 1.05 m tall and weighs 19.12 kg, which roughly represented a humanoid extension of a prototype biped, KURMET [73], developed at Ohio State. Chapters 5-7 return to the full-size model presented in Chapter 2.
4.4.1 Control of a Dynamic Kick

As a first example, PTSC is used along with the state machine shown in Figure 4.4, to control a dynamic kicking motion. During all states, the task prioritization (1) feet, (2) centroidal linear momentum, (3) pose and centroidal angular momentum is employed. Pose control helps to prevent configuration drift. The details of how the commanded task dynamics $\dot{r}_{t,c}$ are specified is described below.\(^{12}\)

Foot control commands linear and angular acceleration of the foot: $\dot{r}_c = \begin{bmatrix} \dot{\omega}_T c_T \\ \ddot{p}_T c_T \end{bmatrix}$. This command is set to zero for feet in support. When the foot is in the air, these rates are selected based on a position/orientation PD scheme

\[
\dot{\omega}_c = \dot{\omega}_d + K_{D,\omega}(\omega_d - \omega) + K_{P,\omega}e_\theta
\]

\[
\ddot{p}_c = \ddot{p}_d + K_{D,p}(\ddot{p}_d - \ddot{p}) + K_{P,p}(p_d - p)
\]

\(^{12}\)Throughout, the subscript \([\cdot]_d\) is used to denote the desired value of a quantity which is often determined from a desired trajectory. Feedback from the current state of the system is then used to track these desired quantities through the prescription of a task-space command, denoted with subscript \([\cdot]_c\).
where \( e_\theta \in \mathbb{R}^3 \) is an angle-axis representation of error between a desired and actual orientation, and satisfies \( ||e_\theta|| \leq \pi \). Desired positions and orientations are derived from hand-authored motion, such as the kick trajectory in the kick state.

While there are other methods to define orientation error (e.g. [75, Eq. 17]), Eq. 4.33 with \( e_\theta \) as used here provides the natural generalization of PD control to \( \text{SO}(3) \) [12]. More concretely, given a desired orientation matrix \( R_d \) and an actual orientation \( R \), \( e_\theta \) satisfies the relationship

\[
e^{S(e_\theta)} R = R_d,
\]  

(4.35)

where the matrix exponential \( e^{S(e_\theta)} \) gives a rotation matrix with rotational axis in the direction of \( e_\theta \) and with rotation angle equal to \( ||e_\theta|| \) [91, 136]. As a result, \( e_\theta \) can be found from a matrix logarithm of \( R_d R^T \) [91].

For centroidal momentum control, a rate of change \( \dot{r}_c \) is commanded separately for linear and angular momentum. For linear momentum, this command is from PD control on the desired CoM (G)

\[
\dot{l}_{G,c} = m[\ddot{p}_{G,d} + K_{D,\ell}(\dot{p}_{G,d} - \dot{p}_G) + K_{P,\ell}(p_{G,d} - p_G)]
\]  

(4.36)

where \( p_G \) is the CoM position and \( m \) is the total mass of the system. The commanded rate of change in angular momentum takes a simpler form\(^{13}\)

\[
\dot{k}_{G,c} = \dot{k}_{G,d} + K_{D,k}(k_{G,d} - k_G).
\]  

(4.37)

All of the balance states and the lift state employ \( \dot{k}_{G,d} = k_{G,d} = 0 \) which provides a dampening of any excess angular momentum.

\(^{13}\)Note again that since conservation of angular momentum is a nonholonomic constraint [92, 146], the angular momentum of a humanoid is not associated with time derivative of any configuration-dependent feature for the humanoid. As a result, the control law for angular momentum does not include any proportional term which would otherwise serve to force the system to a submanifold of the configuration space.
To achieve pose control, joint accelerations are commanded for actuated joints and the torso orientation. For all examples, this commanded acceleration takes the form of a PD law to a static nominal pose. For revolute joints

\[
\ddot{q}_{i,c} = \ddot{q}_{i,d} + K_{D,i} (\dot{q}_{i,d} - \dot{q}_i) + K_{P,i} (q_{i,d} - q_i),
\]

(4.38)

where \( \dot{q}_{i,d} = \ddot{q}_{i,d} = 0 \) in all the examples here. For spherical joints and orientation of the torso, Eq. 4.33 is employed. Since the pose task is optimized at the last level of control, it is generally not possible to fulfill all the desired pose dynamics. Weighting factors are employed that promote closer tracking on certain joints than others. For instance, the shoulder and elbow tasks are given lower weight to promote arm action in the resultant motion. These weights can be incorporated by replacing \( e \) with \( W e \) in Eq. 4.21 for an appropriate diagonal weighting matrix \( W \).

Simple spline trajectories are used throughout to generate the desired dynamic motions. Cubic spline trajectories on the CoM and right foot are used to generate the desired motions for the Balance and Lift states. In the kick state, the foot is commanded to move in an arc centered at the initial right hip position. The orientation of the right foot is commanded to remain tangent to this arc. To ease motion authoring further, the angle of the right virtual leg (from hip to foot center) relative to the vertical is used to characterize the right foot trajectory. A series of cubic splines on this virtual leg angle \( \theta_d \) provide the desired 3D accelerations of the right foot using standard formula that relate accelerations in polar coordinates to Cartesian coordinates.

During a kick, the system predominantly rotates about the stance hip, resulting in non-zero centroidal angular momentum. For the example shown, the inertial \( z \)-axis is opposite gravity and the \( y \)-axis is perpendicular to the sagittal plane. In order to
form the desired centroidal angular momentum, the system’s net moment of inertia about the $y$-axis $I_{yy}$ is first recorded at the beginning of the kick. More specifically, given the composite rigid body of the system $I^C_1$, its value in the centroid frame $G$ takes the special form

$$I^C_G = \frac{1}{X^T} I^C_1 X = \begin{bmatrix} \bar{I}^{cm}_G & 0 \\ 0 & mI \end{bmatrix}$$

(4.39)

where

$$\bar{I}^{cm}_G = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

(4.40)

is the locked Cartesian inertia of the system about the CoM. To promote a whole-body rotation about the $y$-axis, desired centroidal angular momentum and rates are then selected as

$$k_{G,d} = [0, \gamma I_{yy} \dot{\theta}_d, 0]^T,$$

$$\dot{k}_{G,d} = [0, \gamma I_{yy} \ddot{\theta}_d, 0]^T.$$

Here $\gamma = 0.8$ is a factor that accounts for the stance leg remaining stationary. While crude, these desired centroidal momentum dynamics are some of the first non-trivial ones to be designed in the literature and are sufficient to produce rich motion. For a video of the kick motion, please see the accompanying video to the conference paper version of this chapter [143] provided at


Without authoring any upper-body trajectories, the control approach yields complex upper-body motion. Shoulder and torso angles during the kick state are shown in Figure 4.5. During the kick motion, the left shoulder shows a larger displacement.
Figure 4.5: Shoulder and torso angles during the Kick state of the right-foot kick motion. Shoulder angles are measured in the sagittal plane and are taken relative to the torso. The torso angle is measured in the sagittal plane and relative to the inertial coordinate system.

than the right shoulder. This behavior emerges from the task-space controller’s regulation of angular momentum about the global vertical z-axis in response to the right foot’s kick trajectory.

The conic formulation of PTSC was found to be much faster than the QP formulation for this example. As shown in Figure 4.6, the conic formulation (Eqs. 4.28-4.32) was able to be solved in nearly half the time as the QP formulation (Eqs. 4.15-4.18) (in 55% of the time on average). A 4-sided polygonal approximation to the friction cone was employed for the QP formulation. Thus, the majority of the improved speed results here can be attributed to the variable reductions and simplified objective due to the use of $\mathbf{e}$. The times shown in this graph include the computation time of all quantities required by each algorithm ($\mathbf{H}$, $\mathbf{A}_t\dot{\mathbf{q}}$, $\Lambda_{ts}^{-1}$, etc.) as well as the optimization
time of the solver. All computational experiments were run on a 2.3 GHz Intel Core i5 MacBook Pro.

4.4.2 Control of a Standing Broad Jump

The PTSC framework was also applied to produce a standing broad jump using the state machine shown in Figure 4.7. All states during stance use the task priorities (1) feet, (2) centroidal momentum, (3) pose. Centroidal momentum control is omitted during flight, as $\dot{h}_G$ is equal to the net gravity force in the absence of GRFs, and is unable to be modified through internal joint torques.

The details of the task dynamics commanded by the jump state machine are very similar to those in [18]. During the squat state, the desired CoM is lowered though
Figure 4.7: State machine used for jump control. States shown in blue use task priorities (1) feet, (2) centroidal momentum, (3) pose. The state in red (Flight) uses task priorities (1) feet, (2) pose. State transition criteria are noted on the transition arrows, where an omission of a criterion indicates a transition that takes place based on time.

A cubic spline trajectory. At the beginning of the thrust state, the desired CoM position and velocity is discontinuously incremented in the desired direction of the jump. This results in the selection of joint torques that will generate an impulsive ground force on the system. As the legs approach full extension, a knee angle threshold triggers transition to the flight state. In flight, the foot trajectories commanded are largely ballistic, but modified smoothly to position the feet forward for landing. At touchdown, the PD gains on the CoM are softened to provide a smooth landing, with a desired CoM placed over the middle of the support.

The conic and QPs computation times are compared once again for the jump motion, as shown in Figure 4.8. The nested conic solver for PTSC was nearly twice as fast as the nested QP solver (47% faster on average) for the performance of a jump on level terrain. During periods of flight, both methods are able to obtain faster solutions, as the centroidal momentum control is omitted, and ground reaction force variables are excluded from the problem.
The video accompaniment (http://www.go.osu.edu/Wensing_Orin_ICRA2013) also shows the use of the jump controller in a number of different environments. Note that all jumps display significant arm action forward and backward to maintain balance and to help position the feet before landing. This is a feature of biological long jumps [5] that was largely not present in previous work where hand-authored upper-body motions were used [150]. The torso and shoulder angles for the level terrain jump are shown in Figure 4.9. During thrust, the arms are swung upwards to prevent the torso from pitching backwards. If the arms are forced to remain locked, the lack of this behavior results in excess torso pitch as shown.

The jump controller for level terrain is applied without modification for a jump onto uneven terrain. Once the system lands, no knowledge of the terrain is assumed,
Figure 4.9: Shoulder and torso angles during the jump motion. State transitions are shown by the vertical dotted lines, with the state denoted at the top of each corresponding section. Angles are measured the same as in Figure 4.5. The angle of the torso is also shown with arms locked, and results in greater torso pitch during Thrust.
and the feet are simply commanded to not accelerate. Since the control approach used here always attempts to push off the ground, corners of the foot that are originally not in contact are quickly pushed into contact with the unknown terrain. Based on these experiments, it appears that force feedback may not be required for reasonably stiff terrain.

In a final video demonstration, we show the performance of the jump controller when landing on a slippery surface. In this demonstration, the controller assumes that the ground has a coefficient of friction of $\mu = 0.6$, but the simulation employs a coefficient of $\mu = 0.4$. While the system does experience foot slip due to this inaccuracy, the balance controller is robust to this disturbance and results in non-authored arm windmilling to maintain balance. The desired CoM is constantly updated in the example to remain over the middle of the support polygon.

### 4.5 Summary

This chapter has described a method to solve the Prioritized Task-Space Control problem. The use of Prioritized Task-Space Control (PTSC) enables dynamically balanced motions to be performed through the design of trajectories in motion specific task-spaces. This chapter has motivated the power of this framework through application to a dynamic kick and standing jump. Despite only authoring trajectories for the feet and supplying rough set points for the CoM, natural balanced motions, with significant emergent upper arm motion, result from the application of PTSC. Through consideration of the constraints on ground reaction forces, motions for level terrain are able to be easily adapted to uneven terrain scenarios. Without any terrain
information, this approach has been shown to result in a balanced landing following a standing jump onto highly irregular terrain.

In order to solve the PTSC problem this chapter has presented a reformulation of previous QP formulations for PTSC which allows speed gains to be achieved while addressing friction constraints in their full complexity with conic optimization. This reformulation enables real-time control rates of 200 Hz. These algorithms are general to control quantities such as the system’s net angular momentum, which is an important control component to achieve rich upper-body behaviors that are not authored by the designer. Although this chapter has used exclusively hand-designed task-space trajectories as the targets for task-space control, Chapters 5 - 7 will demonstrate how trajectory optimization and control with low-dimensional template models can be applied to generate principled target task trajectories for high-speed running.
Chapter 5

High-Speed Humanoid Running Through Control with a 3D-SLIP Model

5.1 Introduction

This chapter presents a new approach to control high-speed running at speeds of up to 6.5 m/s. Based on the Prioritized Task-Space Control framework presented in Chapter 4, this chapter describes methods to automatically generate desired task-space trajectories for high-speed humanoid running. Rather than relying on hand-authored trajectories alone, a process to form target Center of Mass (CoM) dynamics based on a 3D Spring-Loaded Inverted Pendulum Model (SLIP) is presented. Through the use of PTSC, the dynamics of the 3D-SLIP model are embedded into the humanoid, as shown pictorially in Figure 5.1, resulting in robust high-speed running.

Maintaining dynamic balance during high-speed running is a much more difficult task than during the standing balance tasks studied in Chapter 4. During a high-speed run, the system is rarely statically balanced and requires continuous motion to prevent an irrecoverable fall [108]. In essence, during each individual step of a run, the system is falling forward. This falling motion is not a problem however, since
Figure 5.1: High-speed humanoid running is demonstrated in simulation by commanding the Center of Mass (CoM) dynamics of the humanoid to match that of a 3D-SLIP model. The red spring in this figure represents the compliant target CoM behavior. The humanoid employs torque control at its joints to embed the 3D-SLIP dynamics.

the system is able to catch itself with each following foot step and repeat the pattern continuously.

The difficulty to certify dynamic balance has been a major factor which has relegated modern humanoids to quasi-static operation regimes. Although there are other challenges to perform a run, such as the ability to coordinate the complex coupled effects of rapid lower and upper-body motions, dynamic balance remains the most difficult. This difficulty is influenced by the fact that dynamic balance is not an instantaneous notion, but rather one that requires an understanding of the future dynamics of the system. In order to capture the long-term dynamics of complex legged systems, template models of locomotion, as introduced by Full and Koditschek [36], have the potential to provide accurate descriptions of the salient dynamics of legged
movements. Although these templates do not address the full complexity of the systems that they abstract, the results of this chapter show that control of the salient characteristics of locomotion is an effective and tractable method to manage dynamic balance.

The use of the SLIP model as a template is motivated by its ability to describe the CoM dynamics remarkably well for high-speed locomotion in a variety of insects and animals [9]. Whether hopping, trotting, or running, creatures from cockroaches to kangaroos bounce dynamically, in close accordance with the SLIP model [9]. Despite significant interest in this simple model from the biomechanics and robotics community [4, 8, 25, 34, 38, 106, 119, 123, 133, 151], the work presented here represents the first time that this model has been applied for humanoid gait generation and control.

The use of a compliant leg model is further motivated by recent and continuing advances in high-performance compliant actuators [52]. Leg compliance is not as important in low-speed gaits, where humans typically vault over stiff legs. However, compliant leg operation enables a reduced metabolic cost over stiff gaits at high-speeds [51] and has been shown to play a role in adapting to varied terrain [34]. Despite this motivation, the sparsely studied methods for humanoid running control have been largely adapted from inverted pendulum methods for walking [57, 128]. Other methods have required intensive hand design [48] or offline optimization [21] and have not shown robustness to disturbances.

In contrast, this chapter takes a new approach through the application of the 3D-SLIP model. By applying control to this low-dimensional template model, the humanoid is able to quickly adjust the most important characteristics of this gait, namely its foot touchdown locations and CoM trajectories. In comparison to the
full humanoid model, this simple model is tractable enough that the humanoid can predict and manage the dynamics of its center of mass into the upcoming steps. When coupled with the PTSC, which manages the full dynamics of the humanoid at each instant, the use of this simple model provides an effective whole-body control approach to maintain dynamic balance. Moreover, the simplicity of the template enables the entire control strategy to be capable of real-time computation, which is not the case with more complex whole-body trajectory optimization approaches. The controller described here demonstrates running in simulation at speeds from 3.5-6.5 m/s, is able to reject push disturbances, and manages speed changes in a single step.

The block diagram of the control system used in this chapter is shown in Figure 5.2. The control includes a discrete component that selects touchdown angles at each liftoff for the next step and provides target CoM dynamics based on the 3D-SLIP after the next foot touchdown. A continuous component, provided by a state machine and task-space controller, is then capable to reproduce these target dynamics through torque control of the humanoid robot.

The remainder of the chapter is organized as follows. Section 5.2 presents the 3D-SLIP model and a new method to produce periodic 3D-SLIP trajectories that are able to be retargeted to the humanoid. In Section 5.3, a local deadbeat control approach is introduced to stabilize these trajectories. The controller is specified automatically, without any tuning required, through a local analysis of 3D-SLIP step-to-step dynamics. Section 5.4 presents methods to track these target CoM dynamics with a Task-Space Control approach similar to Chapter 4. As a key feature, the controller again applies centroidal angular momentum control. In running, this approach enables upper-body motions that reduce the required yaw moment for the
Figure 5.2: Block diagram of the control system. Once per step (at each liftoff) a 3D-SLIP controller selects touchdown angles and target CoM compliance characteristics to achieve a desired speed by the end of the next step. A humanoid state machine then selects appropriate task dynamics for the foot and CoM to continuously track the SLIP CoM dynamics and realize the desired foot touchdown locations. The state machine also selects desired angular momentum rates in support to promote balance. A Humanoid Task-Space Controller then selects whole-body joint torques at real-time rates to realize these desired tasks.

motion. Section 5.5 presents running results for single-speed running, speed transitions, and disturbance recovery. A top speed of 6.5 m/s is able to be controlled by the approach presented. All running results are shown in a video supplement to the chapter. Section 5.6 ends with a summary.
5.2 The 3D Spring-Loaded Inverted Pendulum (SLIP) Model and Trajectory Optimization

This section describes the 3D Spring-Loaded Inverted Pendulum (3D-SLIP) model dynamics and introduces a principled trajectory optimization method to provide CoM trajectories for steady-state running. While a 3D-SLIP model is used here, two-dimensional SLIP models have been quite useful in the control and analysis of hopping monopods and bipeds in the sagittal plane. Poulakakis and Grizzle [106] formally embed an extension of the 2D-SLIP model into the dynamics of a hopping monopod with a geometric nonlinear control approach. Hutter et al. [53] studied a SLIP model with an operational-space controller for CoM tracking to regulate hop height and velocity in a simulated leg. Rutschmann et al. [114] applied nonlinear model predictive control to plan SLIP trajectories for uneven terrain footholds. Garofalo et al. [37] developed a 2D-walking controller based on a bipedal SLIP model. While humanoid running is largely dominated by sagittal plane dynamics, these controllers do not have the generality to control lateral sway in humanoid running, and do not provide insight into lateral footstep selection for disturbances. These cases are handled here.

Three-dimensional SLIP models have recently been proposed as a natural generalization of the planar SLIP model. Although these 3D-SLIP models have been the subject of analytical studies [13, 120], their application to trajectory generation and control of humanoid robots has yet to emerge. Seipel and Holmes [120] develop approximations to the 3D-SLIP step-to-step dynamics and show the inherent instability of periodic 3D-SLIP gaits. Carver [13] treats the 3D-SLIP model as a monopod in 3D and analyzes a number of control problems for 3D steering. Most notably, these previous studies have not addressed kinematic constraints on foot positions.
which need to be handled in order to retarget 3D-SLIP trajectories to the humanoid. Additionally, while these models exhibit an infinite number of periodic trajectories for any given forward speed, methods to select preferable template trajectories for a humanoid have not been proposed. The new approach in this chapter provides a general methodology to apply this template model to humanoid running control and finds optimal periodic trajectories which address these constraints.

### 5.2.1 The 3D-SLIP Model

The 3D spring-loaded inverted pendulum (3D-SLIP) model, shown in Figure 5.3, is a natural generalization of the common planar SLIP model [120]. The model consists of a point mass $m$ and leg that experiences phases of support and flight. The mass follows ballistic dynamics in flight wherein the massless leg is positioned for upcoming support. Following touchdown (TD), a Hookean spring with constant $k_s$ and rest length $r_0$ imparts forces onto the mass. The period of support ends at liftoff (LO) when the spring once again reaches its rest length. It is assumed that forward motion is in the positive $x$-direction throughout, as shown in Figure 5.3.

The evolution of the 3D-SLIP model can be described more precisely as a hybrid dynamic system. Since the dynamic evolution of the point mass is intended to describe the CoM of the humanoid, its position in inertial coordinates is denoted as $p_c \in \mathbb{R}^3$ with velocity $\dot{p}_c \in \mathbb{R}^3$. Flight dynamics follow $m \ddot{p}_c = mg$, where $g \in \mathbb{R}^3$ is the gravity vector. In flight, the foot position $p_f \in \mathbb{R}^3$ is adjusted for the upcoming support with touchdown angles $\theta$ and $\phi$, shown in Figure 5.4, as

$$p_f = p_c + p_{\text{hip}}(\sigma) + \ell_h \begin{bmatrix} \sin(\theta) \cos(\phi) \\ \sigma \sin(\theta) \sin(\phi) \\ -\cos(\theta) \end{bmatrix}. \quad (5.1)$$
where $\sigma = 1$ to represent a left foot touchdown, and $\sigma = -1$ to represent a right foot touchdown. As a result, the position of the SLIP leg is set through a rotation $\theta$ about the fixed $-y$ axis, and is followed by a rotation about the fixed $z$ axis by an angle $\sigma \phi$. Thus, for both legs, a positive $\phi$ provides an outward lateral positioning of the foot.

In Eq. 5.1 $\ell_h$ represents the length of the humanoid virtual leg (hip to foot center) at touchdown, and $p_{hip}$ is the position of the hip with respect to the CoM. Here it is assumed that the hip is offset takes the form

$$p_{hip}(\sigma) = \begin{bmatrix} 0 & \sigma y_{hip} & 0 \end{bmatrix}^T$$

(5.2)

where $y_{hip}$ nominally equals half the width of the pelvis. As an alternative to using touchdown angles $\theta$ and $\phi$ to specify the SLIP anchor relative to the CoM, this hip offset allows the touchdown angles to more closely correspond to angles of the humanoid virtual leg. This is a modification over previous work [13, 120] that enables...
Figure 5.4: Touchdown angle definitions. Hip displacement is shown for a right foot touchdown.

more direct application of the 3D-SLIP template to the humanoid and addresses the kinematic reachability of the SLIP footholds.

In support, the dynamics follow

\[ m \ddot{p}_c = k_s(r_0 - ||r||) \hat{r} + mg \]  \hspace{1cm} (5.3)

where \( r_0 \) is the rest length of the spring (computed as its length, from CoM to foot, at touchdown), \( r \in \mathbb{R}^3 \) is given by \( r = p_c - p_f \), and \( \hat{r} \) is the unit vector along \( r \). It is noted that while the 3D-SLIP includes a point foot model, the position \( p_f \) is intended to represent the center of pressure (CoP) below a support foot when retargeted to the humanoid. The model transitions to and from support when the 3D-SLIP state
intersects the TD and LO switching manifolds respectively:

\[ S_{TD} = \{(p_c, \dot{p}_c) \mid e_z^T p_c = \ell h \cos(\theta), e_z^T \dot{p}_c \leq 0\}, \quad (5.4) \]

\[ S_{LO} = \{(p_c, \dot{p}_c) \mid ||r|| = r_0, r^T \dot{p}_c \geq 0\}. \quad (5.5) \]

### 5.2.2 3D-SLIP Apex Return Map

We are interested in controlling the 3D-SLIP model from step-to-step by varying its touchdown angles and spring characteristics. Through selection of these quantities, this dissertation concentrates on regulation of the top-of-flight (TOF) velocity and height of the model. An apex state-of-interest \( x \) is thus constructed from the full SLIP state \( (p_c, \dot{p}_c) \) by

\[
x = \begin{bmatrix} e_x^T p_c \\ e_y^T \dot{p}_c \\ e_z^T p_c \end{bmatrix} = \begin{bmatrix} v_x \\ u_y \\ h \end{bmatrix}
\]

(5.6)

where \( e_x, e_y, \) and \( e_z \) are the unit vectors shown in Figure 5.3. Given touchdown angles and spring characteristics described by \( u_n \) for the \( n \)-th step, an apex return map can be formed

\[
x_{n+1} = f(x_n, u_n, \sigma_n)
\]

(5.7)

which maps the TOF state \( x_n \) to the subsequent TOF state. Letting \( E = \text{diag}(1, -1, 1) \), it follows that the return map satisfies the identity

\[
x_{n+1} = f(x_n, u_n, \sigma_n)
\]

\[
= E f(E x_n, u_n, -\sigma_n).
\]

(5.8)

(5.9)

which amounts to changing the sign convention on the \( e_y \) axis for simulation, and then reversing it following simulation to recover the intended result. As a result, the
return map can be completely specified by the return map when \( \sigma = 1 \). To simplify notation, when omitted, \( \sigma = 1 \) is assumed for the return map.

Since the 3D-SLIP is a passive model, active additions need to be considered to enable the 3D-SLIP to change speeds or to recover from disturbances. Here, it is assumed that the spring stiffness \( k_s \) can vary at the instant of maximum spring compression. Denoting these control variables before and after maximum compression as \( k_{s1} \) and \( k_{s2} \), the control decisions \( u \) for each step are collected as

\[
u = [\theta, \phi, k_{s1}, k_{s2}]^T.
\]  

(5.10)

Given a desired forward speed, Section 5.2.3 introduces a method to find an initial apex state \( x_0 \) and control \( u_0 \) that will lead to periodic 3D-SLIP dynamics. Since it is of interest to find 2-step periodic motions of the 3D-SLIP, search for 1-step motions is restricted to those with alternating lateral velocity, but constant height and forward velocity at each TOF. One-step motions are thus desired with

\[
E x_0 = f(x_0, u_0).
\]  

(5.11)

Section 5.3 then presents a method to stabilize these periodic trajectories by developing 3D-SLIP controllers for different speeds.

5.2.3 Finding Periodic 3D-SLIP Trajectories

For any given forward speed, the 3D-SLIP model exhibits an infinite number of periodic trajectories. For instance, by adjusting touchdown angles and leg stiffnesses, periodic gaits can be generated with different maximum heights, or different lateral sway characteristics. A method is introduced here which uses offline optimization to find periodic 3D-SLIP trajectories that approximately mimic human locomotion and are able to be retargeted to the humanoid model.
Human running data from Rowlands et al. [113] and Hoyt et al. [50] is used to specify target gait timings of the 3D-SLIP model. Studies have shown that human cadence \( c \) (steps per minute) increases [113] and support time \( t_s \) decreases [50] with increased speed. Based on the data in these studies, the following relationships were determined

\[
c = 2.55v_x^2 - 8.77v_x + 172.9 \quad (5.12)
\]

\[
\log_{10}(t_s) = -0.64 \log_{10}(v_x) - 0.2 \quad . \quad (5.13)
\]

To provide additional time for leg positioning in flight, cadence is unmodified from the previous experimental human data, while target support times are shortened to be governed by the following equation

\[
t_s = 10^{-0.2v_x^{-0.82}} \quad . \quad (5.14)
\]

These relationships can be used to determine desired TD and LO times

\[
\mathbf{T}_d(v_x) = [t_{TD,d}, t_{LO,d}]^T \quad (5.15)
\]

as a function of forward velocity. Given a TOF state-control pair \((x, u)\), dynamic simulation can be used to evaluate the actual TD and LO times. This evaluation is denoted by the mapping \( g \)

\[
[t_{TD}, t_{LO}]^T = g(x, u) \quad . \quad (5.16)
\]

Given a desired forward TOF velocity \( v_x \), a least-squares optimization problem can then be formulated to find a state-control pair for an upcoming left foot support \((\sigma = 1)\) which matches the periodicity constraint (Eq. 5.11) and achieves the desired
gait timings

\[
\min_{h_0,v_0,k_s,\theta} \, \| \mathbf{E} \mathbf{x}_0 - \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) \|^2 + \| T_d(v_x) - \mathbf{g}(\mathbf{x}_0, \mathbf{u}_0) \|^2 \tag{5.17}
\]

where \( \mathbf{x}_0 = [v_x, v_y, h_0]^T \)

\[
\mathbf{u}_0 = [\theta, 0, k_s, k_s]^T. \tag{5.18}
\]

Note that a touchdown angle of \( \phi = 0 \) has been fixed to ensure a gait with footstep locations directly in front of the hips. Additionally, the spring stiffnesses \( k_{s_1} \) and \( k_{s_2} \) are selected to be equal, since any change in stiffness would change the 3D-SLIP total energy and prevent satisfaction of Eq. 5.11.

Given a zero residual error for Eq. 5.17, the optimal state-control pair \((\mathbf{x}_0^*, \mathbf{u}_0^*)\) can be used to generate a 2-step periodic 3D-SLIP motion. To see this, the dependence of the return map on \( \sigma \) may be reintroduced, with \( \mathbf{x}_1^* \) and \( \mathbf{x}_2^* \) defined as

\[
\mathbf{x}_1^* = \mathbf{f}(\mathbf{x}_0^*, \mathbf{u}_0^*, \sigma_0) \tag{5.20}
\]

\[
\mathbf{x}_2^* = \mathbf{f}(\mathbf{x}_1^*, \mathbf{u}_0^*, \sigma_1), \tag{5.21}
\]

where \( \sigma_0 = 1 \) and \( \sigma_1 = -1 \). Yet, since \( \mathbf{x}_1^* = \mathbf{E} \mathbf{x}_0^* \), it follows that

\[
\mathbf{x}_2^* = \mathbf{f}(\mathbf{E} \mathbf{x}_0^*, \mathbf{u}_0^*, \sigma_1) \tag{5.22}
\]

\[
= \mathbf{E} \mathbf{f}(\mathbf{E}^2 \mathbf{x}_0^*, \mathbf{u}_0^*, -\sigma_1) \tag{5.23}
\]

\[
= \mathbf{E} \mathbf{f}(\mathbf{x}_0^*, \mathbf{u}_0^*, \sigma_0) \tag{5.24}
\]

\[
= \mathbf{E}^2 \mathbf{x}_0^* \tag{5.25}
\]

\[
= \mathbf{x}_0^*. \tag{5.26}
\]

through application of Eq. 5.9 and use of the property \( \mathbf{E}^2 = \mathbf{I}_{3\times3} \).

The optimization of Eqs. 5.17-5.19 was performed in MATLAB with the nonlinear least-squares function \texttt{lsqnonlin}. Despite the need to use dynamic simulation of the
3D-SLIP model in the evaluation of $f$ and $g$, the optimization is solved quickly in MATLAB. For instance, it takes approximately 20 seconds to generate 31 periodic 3D-SLIP gaits for forward speeds from 3.5 m/s to 6.5 m/s (at 0.1 m/s increments). Over this range of speeds, the optimal state-control pairs $(x_0^*, u_0^*)$ exhibit velocity-dependent touchdown angles and leg stiffnesses which increase with speed from $23.1^\circ$ to $26.5^\circ$ and 11.7 to 16.4 kN/m respectively. The optimized TOF heights decrease only slightly with speed, from 91.0 cm at 3.5 m/s to 88.0 cm at 6.5 m/s. Across this entire range of speeds, the methods proposed above ensure that the required footholds remain kinematically accessible by the humanoid, and the required matching of human gait timing enforces a natural preference among the many periodic 3D-SLIP gaits available.

5.3 3D-SLIP Deadbeat Control

Once periodic 3D-SLIP motions have been generated, a 3D-SLIP controller is desired to transition from nearby TOF states to a periodic trajectory. Deadbeat control laws can be developed to achieve this goal in a single step but often require online optimization [53] or prohibitively large knowledge bases [13]. Here a first-order approximation to a deadbeat controller is developed around the periodic apex state. This control law is easy to compute offline and can be applied online for real-time control of a humanoid.

Let $(x_0^*, u_0^*)$ be a state-control pair which satisfies Eq. 5.11 as computed in the previous section. A first-order approximation to the return map around $(x_0^*, u_0^*)$
provides:

\[ x_1 = f(x_0^* + \Delta x, u_0^* + \Delta u) \]  
\[ \approx E x_0^* + J_x \Delta x + J_u \Delta u \] (5.27)

where \( J_x = \frac{\partial f}{\partial x} \) and \( J_u = \frac{\partial f}{\partial u} \) are Jacobians of the return map evaluated at \((x_0^*, u_0^*)\). These Jacobians can be evaluated numerically with finite differences. For a given TOF error \( \Delta x \), the control objective of driving \( x_1 \) to \( E x_0^* \) can be achieved approximately by selecting \( \Delta u \) such that:

\[ J_u \Delta u = -J_x \Delta x. \] (5.29)

Numerical experiments have shown \( J_u \in \mathbb{R}^{3 \times 4} \) to be full rank, which provides redundancy to satisfy Eq. 5.29. To account for this redundancy, the change in spring constant \( \Delta k_s \) is chosen to be opposite that of \( \Delta k_s \). Thus, letting the reduced control \( \Delta \tilde{u} = [\Delta \theta, \Delta \phi, \Delta k_{s1}]^T \), with

\[ \Pi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}, \]

it follows that

\[ \Delta u = \Pi \Delta \tilde{u} \]

recovers the full control. Under this reduction, the selection of

\[ \Delta u = -\Pi (J_u \Pi)^{-1} J_x \Delta x \]

satisfies Eq. 5.29 and provides deadbeat control to the linearized return map. Thus, letting \( K = -\Pi (J_u \Pi)^{-1} J_x \) the resulting control law is given as

\[ u = u_0^* + K(x_0 - x_0^*). \] (5.30)
Similar to arguments by Carver [13], the implicit function theorem can be applied to show that Eq. 5.30 is in fact a first-order approximation to a deadbeat controller under the restriction $\Delta k_{s_1} = -\Delta k_{s_2}$. It is important to note that Eq. 5.30 is only a valid deadbeat control approximation approach for an upcoming left foot support ($\sigma = 1$). Similar symmetry arguments to in steady-state running provide the deadbeat approximation for an upcoming right foot support ($\sigma = -1$) as

$$u = u_0^* + K(Ex_0 - x_0^*) . \quad (5.31)$$

The value of $K$ to stabilize periodic running at 3.5 m/s is shown in Eq. 5.32 and displays many expected relationships. For instance, the first column shows which control actions should be taken if the system needs to change forward speed. A positive $\Delta v_x$ indicates that the system is moving too fast, which requires a larger touchdown angle $\theta$ and removal of spring energy $\Delta k_{s_2} < 0$ to correct the error. Similar expected relationships mainly modify $\phi$ to correct for lateral velocity error. Note that these gains are not tuned, but rather are provided automatically from solution of Eq. 5.29.

$$\begin{bmatrix} \Delta \theta \\ \Delta \phi \\ \Delta k_{s_1} \\ \Delta k_{s_2} \end{bmatrix} = \begin{bmatrix} 0.13 & -0.013 & -0.51 \\ -0.076 & 0.90 & -1.95 \\ 13.2 & 0.86 & 36.9 \\ -13.2 & -0.86 & -36.9 \end{bmatrix} \begin{bmatrix} \Delta v_x \\ \Delta v_y \\ \Delta h \end{bmatrix} . \quad (5.32)$$

Here all angles are measured in radians, distances are measured in meters, and spring constants have units kN/m. Separate feedback matrices $K$ are computed for each of the 31 periodic 3D-SLIP state-control pairs $(x_0^* , u_0^*)$, generated in Section 5.2.3, for speeds from 3.5-6.5 m/s.
Figure 5.5: State machine used for running control. State transition events are noted on the transition arrows. Local SLIP deadbeat control occurs at each liftoff to select touchdown angles and target CoM compliance characteristics for the upcoming step.

5.4 Humanoid Control

In order to embed the 3D-SLIP dynamics into the humanoid model, the Prioritized Task-Space Control approach of Chapter 4 is applied. However, the 3D-SLIP model does not provide all the necessary task-space commands that are required to achieve a high-speed run. A running state machine is used to assist in the specification of the remainder of the task-space commands. These commands need to be tailored to the current phase of motion, such as when each different foot is in support. This required structure makes a state machine a natural choice to sequence the humanoid controller through these phases.
5.4.1 Running State Machine

A running state machine, shown in Figure 5.5, is used to sequence the humanoid through phases of support and flight. The state machine is assumed to have access to the system state \((\mathbf{q}, \dot{\mathbf{q}})\) in order to formulate commanded task dynamics for the feet and CoM to track the 3D-SLIP template behavior. In addition, centroidal angular momentum control is applied in support due to its postulated role in the maintenance of balance [95], and a pose controller is applied to enable the specification of a desired system configuration. The commanded task dynamics are similar to the application of task-space control for a dynamic kick and jump in Chapter 4.

5.4.2 Prioritized Task-Space Control

A foot controller operates in all states to command angular and linear foot accelerations, \(\dot{\mathbf{\omega}}_c\) and \(\ddot{\mathbf{p}}_c\), for foot trajectory tracking. For support feet, this command is set to zero. When the foot is in the air, these rates are selected based on Eq. 4.33 for orientation and Eq. 4.34 for the position of the foot relative to the CoM.
A simple concatenation of three cubic spline trajectories relative to the CoM is
used to provide \((p_d, \dot{p}_d, \ddot{p}_d)\) in the law dictated by Eq. 4.34. A sample flight foot
trajectory relative to the CoM is shown in Figure 5.6. These three cubic splines serve
to lift, transfer, and plant the foot. Transfer and touchdown targets are adjusted
online based on the SLIP template touchdown angles. Estimated support and flight
times from the SLIP model are used to set the timing of these trajectories. Desired
orientations for each foot follow cubic splines on the pitch angle of the foot. Foot
pitch angle waypoints are manually specified and do not vary with speed. We note
that the foot trajectories end with zero velocity relative to the CoM, which induces
losses at impact. Early leg retraction as studied by Palmer and Orin [98] could be
added to improve touchdown in future work.

The use of centroidal momentum control is an important component to stabilize
the high-speed run. The centroidal momentum controller operates in the support
states only, and commands rates of change in system linear and angular momentum
to track the 3D-SLIP trajectories and promote balance. A rate of change in total
system linear momentum \(\dot{l}_G\) is commanded from PD control of the humanoid CoM
\((G)\) to position of the 3D-SLIP mass:

\[
\dot{l}_{G,c} = m[\ddot{p}_c + K_{D,\ell}(\dot{p}_c - \dot{p}_G) + K_{P,\ell}(p_c - p_G)]
\]  

(5.33)

where \(p_G\) is the CoM position and \(m\) is the total mass of the system. At each LO,
the state of the 3D-SLIP template is reset to coincide with that of the humanoid.
The apex state \(x\) is predicted, and the 3D-SLIP control law of Eq. 5.30 is applied.
This control provides target touchdown angles for the upcoming support to realize
a desired speed. The 3D-SLIP model is integrated forward by the state machine in
software to provide continuous target dynamics to the humanoid.

99
The application of centroidal angular momentum control is less straightforward than in standing balance. The commanded rate of change in centroidal angular momentum $\dot{k}_{G,c}$ again takes a simple form

$$
\dot{k}_{G,c} = -K_{D,k}k_G.
$$

(5.34)

to provide a dampening of any excess angular momentum. While the roll and yaw angular momentum are well regulated near zero in human running [45], the pitch angular momentum is not due to leg cycling. With this in mind, pitch angular momentum is ignored by the task-space controller that processes this command.

To achieve pose control, joint accelerations are commanded for actuated joints and the torso orientation as in Chapter 4. For all joints except the shoulder, the desired pose is fixed and has zero rate. Commanded joint accelerations follow Eq. 4.38 for revolute joints and Eq. 4.33 for spherical joints. Desired shoulder pitch angles and rates are commanded proportional to those of the opposite virtual leg. This promotes a swinging of the arm in phase with the opposite leg. This angular momentum canceling motion is further modified by the task-space controller which attempts to regulate the yaw angular momentum to zero.

For all running results, the foot positions and orientations are set as a first priority, with all other tasks as a secondary priority. Task weightings are set to encourage better tracking of certain tasks. Here, arm task weightings are tuned to provide upper-body motion freedom to the task-space controller. Task weightings and gains are summarized in Table 5.1. Precise tuning of these values is not required to produce stable running, but does affect the nuances of the motion (arm swing magnitude, for instance) due to task trade-offs.
Table 5.1: Weight and gain settings for the PTSC. Where omitted, all derivative gains are set for critical damping ($K_D = 2\sqrt{K_P}$).

### 5.5 Results

The use of a high-level 3D-SLIP controller coupled with a lower-level task-space controller is shown to enable high-speed humanoid running that is able to change speeds and recover from disturbances. This section presents running results at a fixed speed and then demonstrates the tracking capabilities of the controller. The same commanded task gains, task-space weightings, and task-space priorities are employed across all results.

#### 5.5.1 Steady-State Fixed-Speed Running

The capabilities of the controller are shown for running at a commanded speed of 3.5 m/s. A video attachment to the conference paper version of this chapter [144] shows representative running results and is visible at


The CoM velocity tracking of the task-space controller is shown in Figure 5.7. Despite the impact at TD, the controller is able to provide tracking of the CoM
Figure 5.7: CoM velocity tracking for running at a desired speed of 3.5 m/s. The velocities in the forward (x) and vertical (z) directions are 1-step periodic, while the lateral velocity (y) is 2-step periodic.

velocity to that of the SLIP model in all directions. This impulse, not captured in the 3D-SLIP model, represents a disturbance that is effectively handled by the PTSC. Note that the CoM tracking is not given explicit priority over other commanded task dynamics such as torso orientation or angular momentum. Although explicit CoM prioritization does lead to better tracking results for the CoM, it was found that the system is more robust to disturbances when CoM tracking is not prioritized. Still, the error in TOF forward velocity is approximately 2% for the results in the graph shown.

The angular momentum control applied has advantages to prevent the feet from slipping due to excess required yaw moments. Figure 5.8 shows the contribution of
Figure 5.8: Yaw angular momentum about the CoM as contributed by the legs and the upper body. The combination of a simple target arm motion and centroidal angular momentum control cause the upper body to cancel the majority of the yaw angular momentum generated by leg cycling.

the upper and lower body to the overall centroidal yaw angular momentum. The task-space controller results in upper-body motions that cancel the majority of the yaw angular momentum of the lower body. This role of the arms in the regulation of yaw angular momentum is a characteristic that is observed in human running [45]. Note that the derivative (slope) of the total yaw angular momentum curve is equal to the generated yaw moment at the foot. The application of yaw momentum control coupled with the arm swing heuristic has effectively decreased the required yaw moment at the feet during support.

5.5.2 Running Transitions

The 3D-SLIP controller provides the task-space controller with reference CoM dynamics to change speeds and recover from push disturbances. Figure 5.9 shows the
tracking of a commanded forward velocity profile. We note that for each commanded speed, a periodic 3D-SLIP solution (Eq. 5.17) and a 3D-SLIP control law (Eq. 5.30) have been computed offline. This amounts to storing a small amount of information, $(x_0, u_0, K)$, for each desired speed. The controller is able to accelerate at up to 0.2 m/s per step and decelerate at up to 0.4 m/s per step. The controller is unable to accelerate faster, as the approximate deadbeat controller does not take into account constraints on the touchdown angle which are required to limit the minimum touchdown angle $\theta$ based on the TOF height. Once again, the CoM is not explicitly prioritized, which prevents perfect tracking at TOF but improves robustness to disturbances.

### 5.5.3 Push Disturbances

Figure 5.10 shows the response of the system to a series of lateral disturbances. Pushes are either 1000 N or 750 N and are applied for 40 ms. The system is able to
maintain balance when the same pushes are applied in the sagittal plane as well. Lateral push recovery is detailed here, as its out-of-plane dynamics are a new complexity that is managed by the 3D-SLIP controller. All pushes occur during support. For instance, the first push occurs during a right foot support immediately before liftoff. The 3D-SLIP controller picks touchdown angles that modify the left foot touchdown to reject this additional $y$ velocity. A push to the left can also be rejected when the left foot is in support, by taking a cross step with the right foot. This is shown in the next series of 3 disturbances (each 30 N·s) that occur in succession. The final two 40 N·s pushes occur earlier in support, and can be partially rejected by the CoM controller in support. These require less extreme recovery footsteps, but result in different torso dynamics, as shown in the video attachment. Although steady-state running is largely dominated by sagittal plane dynamics, these out-of-plane disturbance recovery results are unable to be described by planar SLIP models. This result showcases a major advantage of applying the 3D-SLIP.

5.6 Summary

This chapter has presented a whole-body control approach for high-speed running in a simulated humanoid robot. Local deadbeat control applied to a 3D-SLIP template model provides appropriate CoM dynamics for the system to change speeds and recover from large disturbances. While the local deadbeat control is only approximate, the simple form of the control law enables use online. When coupled with a task-space controller that operates at real-time rates, these corrective reference CoM dynamics are accurately reproduced by the simulated humanoid and are an effective method for the system to maintain dynamic balance. As a key feature, this approach
Figure 5.10: Footstep selection for lateral disturbance rejection. Shown are the actual touchdown locations of the humanoid foot. Target footsteps are selected from local deadbeat control of the 3D-SLIP model.

provides footstep planning that is automatically adjusted with changing speeds and disturbances without any necessary retuning of the control parameters.

These results represent the first application of the 3D-SLIP model to the control of a whole-body humanoid system and encourage its further use. The following chapters expand upon the use of this model to enable new high-speed capabilities for the humanoid. Chapter 6 describes modifications to enable humanoid turning, and Chapter 7 describes a 3D-SLIP model to produce a running long jump.
Chapter 6

High-Speed Humanoid Turns

6.1 Introduction

This chapter develops new methods to control humanoid turns during high-speed running through adaptation of the 3D-SLIP running approach given in Chapter 5. As opposed to previous 3D-SLIP studies, which have focused on steering with a monopod model, motion optimization for the SLIP here enforces leg separation. This leg separation gives rise to body sway in forward running, and allows the template to capture the unique roles that the inside and outside legs each play during a high-speed turn. Much like the approach in Chapter 5, modifications away from a monopod model of running are required in order to make the template useful for a two-legged humanoid model. The techniques proposed allow the humanoid to change its turn rate and direction from step-to-step, as shown in Figure 6.1, and enables execution of a high-speed turn with a radius that is one fourth that of a standard 400m track.

3D-SLIP steering control has been addressed for single leg models by a variety of other authors. Recent work by Wu and Geyer [151] has studied 3D-SLIP steering and discovered time-based deadbeat control laws that provide terrain robustness to the template. These laws extend those by Ernst et al. [22] for 2D-SLIP hopping with
Figure 6.1: This chapter describes methods for online control of humanoid turning. The use of a 3D-SLIP template model with steering control allows a knowledge base to be developed for humanoid running control at different speeds and turning rates. CoM trajectories generated from this knowledge base are tracked online with a whole-body task-space controller which uses torque control at the humanoid joints.

time-based deadbeat control. While the methods in this chapter are not presently robust to terrain variation, a synergy with the ideas in [151] could be an interesting avenue to provide future terrain robustness. Carver [13] studied a variety of 3D-SLIP steering problems for a single leg model, but also did not focus on trajectories that could be retargeted to a humanoid.

The challenge to generate 3D-SLIP trajectories for a humanoid turn are shown in Figure 6.2. Figure 6.2(a) shows the 3D-SLIP trajectories for a continuous monopod turn. For this case, the foot is always placed towards the outside of the turn, which would create kinematic difficulty for the inside leg of a humanoid to reach the desired foothold. In order for the inside and outside legs to have different foot placements that are both reachable by the humanoid, each leg must play a unique role to accomplish
Figure 6.2: CoM trajectories for 3D-SLIP steering with monopod and biped turning. In monopod turning, the foot is always placed toward the outside of the turn. This creates difficulty in the humanoid for the inner leg to reach the desired foothold. During a biped turn, hip separation is addressed, and each leg employs a different positioning relative to the center of mass. Despite the presence of CoM sway, the methods proposed here are shown to still follow a nominal circular path.

Overall, very little work has been done on executing dynamic turns for humanoid robots. An exception is the recent work by Miura et al. [87] which has focused on controlling an in-place turn wherein the feet slip rotationally with respect to the ground. The running turn executed here faces challenges that are largely distinct from an in-place turn. Due to its non-zero turn radius, a significant centripetal force is required.
to execute the turn. In simulation, Hodgins et al. [47] developed running turns for human characters through a series of heuristically designed controllers. Palmer and Orin [100] studied fuzzy control methods for a turn in a quadruped trot. Krasny and Orin [62] developed a turn during a quadruped gallop using computationally intensive offline genetic algorithms. Mordatch et al. [89] used online optimization with a novel template to generate turns during a humanoid jog. The use of the 3D-SLIP to execute a humanoid running turn here is a new feature in comparison to previous work, allowing higher speed locomotion in comparison to Mordatch et al. [89] and requiring less intensive heuristic design in comparison to Hodgins et al. [47]. Unlike either of these previous works, the system is capable to maintain balance under push disturbances at high speeds.

The remainder of this chapter is organized as follows. Section 6.2 introduces the 3D-SLIP model and a new notion of its principal heading direction. Section 6.3 then presents an optimization-based approach to generate 3D-SLIP trajectories for a turn. Methods to track the resultant trajectories are presented in Section 6.5 with many details similar to that in Chapter 5. Figure 6.3 provides a summary of the main changes to the control architecture from Chapter 5 which enable humanoid turns here. The results in Section 6.6 describe how the new methods allow the inside and outside legs to play specialized roles during the turn. Namely, the outside leg produces the majority of the centripetal force to execute the turn, while the inside leg largely only supports the weight of the humanoid. These roles are consistent with the kinematic ability of each leg to generate centripetal force, and are a consequence of the 3D-SLIP optimization proposed. This result is the high-level contribution of the extensions in
Figure 6.3: Block diagram for the humanoid control system with 3D-SLIP CoM steering. The structure of this controller is similar to that in Chapter 5, with the main changes for turning underlined in the diagram. A step controller based on the 3D-SLIP model is used to find touchdown angles and SLIP-spring parameters for the next step. These parameters are used to drive a 3D-SLIP simulation which provides CoM trajectories online. The turning state machine here uses the principal heading direction of the 3D-SLIP model to shape its commands to the task-space controller.

This chapter, as previous 3D-SLIP steering approaches have not addressed the ability to realize their trajectories in a two-leg morphology.

6.2 3D-SLIP Model for Biped Turns

In this chapter, the 3D-SLIP model is again adopted to generate target CoM dynamics for the humanoid. This section presents new definitions for its continuous
and discrete dynamics before developing a trajectory optimization strategy in Section 6.3.

6.2.1 Principal Heading Angle

The 3D-SLIP model used here is in many parts the same as that used in straight-ahead running. As a main difference, a supplementary coordinate system is included in the template to capture the principal heading direction of its mass during a turn. This supplementary coordinate system \( e'_x, e'_y, e'_z \) is shown in Figure 6.4 and rotates, about the inertial \( e_z \) axis, as the 3D-SLIP mass changes its principal direction of travel. It is important to note that due to effects of lateral sway, the \( e'_x \) forward heading coordinate is not generally aligned with the velocity of the SLIP mass. The orientation of this prime coordinate system is given with respect to the inertial coordinate system (ICS) \( e_x, e_y, e_z \) by the rotation matrix:

\[
R_s(\gamma) = \begin{bmatrix}
\cos(\gamma) & -\sin(\gamma) & 0 \\
\sin(\gamma) & \cos(\gamma) & 0 \\
0 & 0 & 1
\end{bmatrix},
\]

(6.1)

where \( \gamma \) represents the principal heading angle.

The 3D-SLIP model follows the same alternating periods of ballistic flight followed by support as in Chapter 5. As a main difference, the foot position \( p_f \in \mathbb{R}^3 \) is adjusted in flight for the upcoming support with touchdown angles \( \theta \) and \( \phi \) as

\[
p_f = p_c + R_s \cdot \left( p_{hip}(\sigma) + \ell_h \begin{bmatrix}
\sin(\theta) \cos(\phi) \\
\sigma \sin(\theta) \sin(\phi) \\
-\cos(\theta)
\end{bmatrix} \right),
\]

(6.2)

where \( \sigma \in \{1, -1\} \) encodes a left or right leg, respectively. As denoted previously, \( \ell_h \) represents the length of the humanoid virtual leg (hip to foot center) at touchdown, and \( p_{hip} \) is the position of the hip with respect to the CoM (Eq. 5.2). The quantity \( \theta \)
controls the angle between the humanoid virtual leg and the $e'_z$ axis, while $\phi$ rotates the leg laterally outward about the $e'_z$ axis. These touchdown angle definitions represent a difference from Eq. 5.1, where instead touchdown angles encoded rotations about fixed inertial axes. With the new touchdown angle definitions, in the special case when $\phi = 0$, the foot is placed directly in front of the hip along the $e'_x$ direction. Flight ends at touchdown (TD) when the foot intersects the ground plane.

In support, the dynamics again follow

$$m \ddot{p}_c = k_s(r_0 - \|r\|) \hat{r} + mg \quad (6.3)$$

where $r_0$ is the rest length of the spring (computed as its length at touchdown). Support ends at liftoff (LO), when the leg has returned to its rest length.
6.2.2 Extended Return Map

While the continuous dynamics of the 3D-SLIP model are needed to generate continuous CoM trajectories online, the discrete step-to-step dynamics of this model are useful to generate step controllers. Again, discrete control is limited to a slice of the SLIP state $x$ at Top of Flight (ToF) which includes the CoM velocity and height:

$$x = \begin{bmatrix} e^T \dot{p}_c \\ e^T \dot{p}_c \\ e^T p_c \end{bmatrix}.$$  \hfill (6.4)

Starting from a ToF state $x_n$ with assumed heading $R_s = \Sigma_n$, the touchdown angles $\theta, \phi$ and leg spring constant $k_s$ can be selected to steer the model to a desired state $x_{n+1}$ at the following ToF. As in Chapter 5, the spring constant is allowed to change at midstance, with spring constants $k_{s1}$ and $k_{s2}$ denoting the values before and after midstance, respectively. Given a selection of control variables $u_n = [\theta, \phi, k_{s1}, k_{s2}]^T$, forward simulation of the continuous dynamics can be used to determine the next ToF state, which is described by a redefined return map $f$ as

$$x_{n+1} = f(x_n, \Sigma_n, u_n, \sigma_n).$$  \hfill (6.5)

Due to the structure of $R_s$, this return map obeys:

$$f(x_n, \Sigma_n, u_n, \sigma_n) = \Sigma_n f(\Sigma_n^T x_n, 1, u_n, \sigma_n).$$  \hfill (6.6)

This equation amounts to performing simulation in ToF local coordinates, and then rotating the result back into the ICS. Similarly, the return map satisfies a generalization of Eq. 5.9

$$f(x_n, 1, u_n, \sigma_n) = Ef(E^T x_n, 1, u_n, -\sigma_n)$$  \hfill (6.7)

114
where $E = \text{diag}(1, -1, 1)$. Combined with Eq. 6.6, this provides the relationship

$$f(x_n, \Sigma_n, u_n, \sigma_n) = \Sigma_n E f(E \Sigma_n^T x_n, 1, u_n, -\sigma_n).$$ (6.8)

Therefore the return map can always be evaluated through use of an associated evaluation with $\sigma = 1$. Thus, again, when omitted, $\sigma = 1$ is assumed to simplify notation.

### 6.3 3D-SLIP Steering Optimization

This section introduces a principled optimization-based approach to find 3D-SLIP controls $u$ for running at a given speed and turning rate. As opposed to forward running, separate controls are required for inside and outside legs during a turn. The unique role played by the inside and outside legs is a new property of the 3D-SLIP optimization proposed here. Optimized trajectories for a given forward speed are designed to maintain continuity when transitioning to and from any turn rate. This design enables online humanoid steering with step-to-step modifications in the turn rate. Previous results for straight-ahead running are first summarized, which allows for consideration of the special case of transitions from straight-ahead running into a turn.

#### 6.3.1 Use of 3D-SLIP Optimization for Straight-Ahead Running

Given a desired forward speed $v_x$, the 3D-SLIP optimization approaches for straight-ahead running (Eqs. 5.17 - 5.19) provide a ToF state $x^* = [v_x, v_y^*, h^*]^T$ and control variables $u^* = [\theta^*, \phi^* = 0, k_s^*, k_s^*]^T$ for periodic 3D-SLIP running. These parameters allow the humanoid to run with its feet placed in front of its hips when trajectories are retargeted. It is key to note that this footstep separation gives rise
to lateral CoM sway with velocity $v_y^*$ at ToF. The ToF state-control pair is optimized to satisfy:

$$ E \mathbf{x}^* = f(\mathbf{x}^*, 1, \mathbf{u}^*) \quad (6.9) $$

which enforces 1-step periodicity on the ToF height $h$ and forward velocity $v_x$, but 2-step periodicity on the ToF lateral velocity $v_y$. As a matter of convention, $\mathbf{x}^*$ is selected to represent the ToF state prior to a period of left foot support, while the ToF state before a period of right foot support is given by $E \mathbf{x}^*$.

### 6.3.2 Transitions from Straight-Ahead Running into a Turn

As a special case, consider the situation of starting from a periodic run with forward speed $v_x$. Given the state-control pair for straight-ahead running $(\mathbf{x}^*, \mathbf{u}^*)$, new control inputs must be employed to start from $\mathbf{x}_0 = \mathbf{x}^*$ and execute a turn. Here, as one possible approach, the rate of turning is characterized by a change in heading angle $\Delta \gamma$ per step. The method proposed here enforces this turn angle by selecting left foot and right foot parameters $\mathbf{u}_L$ and $\mathbf{u}_R$ such that

$$ R_s(\Delta \gamma) E \mathbf{x}_0 = f(\mathbf{x}_0, 1, \mathbf{u}_L, \sigma_L = 1), \quad (6.10) $$

$$ R_s(2\Delta \gamma) \mathbf{x}_0 = f( R_s(\Delta \gamma) E \mathbf{x}_0, R_s(\Delta \gamma), \mathbf{u}_R, \sigma_R = -1). \quad (6.11) $$

While the turn angle $\Delta \gamma$ is only enforced on velocity, it will be shown in Section 6.4 that the resultant CoM positions have a sweep angle of $2\Delta \gamma$ from ToF to ToF following a left and right foot cycle.

Intuitively, if the condition in Eq. 6.10 is satisfied, the next ToF velocity with respect to local coordinates will be numerically equal to the ToF velocity with respect to the ICS during forward running. This property allows the resultant CoM trajectory to be smoothly composed with that for any new turn rate in the following
step. However, this condition also enforces alternating lateral sway velocities (in local
coordinates) from step-to-step during the turn. While this may seem like a drawback,
because it allows sway to the outside of the turn on every other step, the results in
Section 6.6 show that this design decision allows each leg to make proper use of its
kinematic availability to generate centripetal force during the turn.

To find such a $u_L$ which satisfies Eq. 6.10, the following optimization problem is
formulated:

$$\min_{\theta, \phi} \| R_s(\Delta \gamma) E x_0 - f(x_0, 1, u) \|^2$$

subject to

$$u = [\theta, \phi, k_s, k_s^*]$$

(6.13)

$$\underline{\theta} \leq \theta \leq \overline{\theta}$$

(6.14)

$$\underline{\phi} \leq \phi \leq \overline{\phi}$$

(6.15)

where the $\underline{\ }$ and $\overline{\ }$ symbols represent bounds summarized in Table 6.1. The selection
of a leg stiffness equal to that in forward running ($k_s^*$) has been found to simplify
optimization and result in comparable gait timings to forward running. This opti-
mization problem can be solved quickly (usually under a second with an appropriate
initial guess) in MATLAB using the nonlinear least squares function $\text{lsqnonlin}$. Turn
angles $\Delta \gamma$ are determined to be valid in the case that the objective function is zero
at the optimum. This process is repeated offline for a range of forward speeds and
turn angles.

Controls for the right leg with a turn angle of $\Delta \gamma$ can be found from corresponding
controls for the left leg with a turn angle of $-\Delta \gamma$. Due to property Eq. 6.8, Eq. 6.11
can be simplified to

$$R_s(2\Delta \gamma) x_0 = R_s(\Delta \gamma) E f(x_0, 1, u_R, 1).$$

(6.16)
<table>
<thead>
<tr>
<th>Variable</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta$</td>
<td>0</td>
<td>$\pi/4$</td>
</tr>
<tr>
<td>$\phi$</td>
<td>$-\pi/2$</td>
<td>$\pi/2$</td>
</tr>
</tbody>
</table>

Table 6.1: Kinematic bounds placed on the humanoid virtual leg angles for 3D-SLIP optimization.

Premultiplication of both sides by $\mathbf{E} \mathbf{R}_s(\Delta \gamma)^T$ provides

$$
\mathbf{E} \mathbf{R}_s(\Delta \gamma) \mathbf{x}_0 = f(\mathbf{x}_0, 1, \mathbf{u}_R, 1).
$$

(6.17)

Yet, since the form of $\mathbf{R}_s$ satisfies

$$
\mathbf{E} \mathbf{R}_s(\Delta \gamma) \mathbf{E} = \mathbf{R}_s(-\Delta \gamma)
$$

(6.18)

it follows that Eq. 6.17 is equivalent to

$$
\mathbf{R}_s(-\Delta \gamma) \mathbf{E} \mathbf{x}_0 = f(\mathbf{x}_0, 1, \mathbf{u}_R, 1).
$$

(6.19)

Comparing this equation to Eq. 6.10, it follows that the left leg controls for turn angle $-\Delta \gamma$, are the same as the right leg controls for turn angle $\Delta \gamma$.

6.3.3 Control of 3D-SLIP Steering

As in Chapter 5, 3D-SLIP feedback controllers can be constructed to stabilize the CoM dynamics of the humanoid when they deviate from the 3D-SLIP trajectories. In this manner, the state of the 3D-SLIP can be periodically reset to match the CoM state of the humanoid, while SLIP-based trajectories are generated online to return to the desired CoM motion.

In straight-ahead running, a single deadbeat controller was required for a given forward speed, and was appropriate for both legs. Theoretically, to achieve local deadbeat control for turning, separate linear feedback matrices $\mathbf{K}$ could be constructed to
satisfy Eq. 5.29 for each forward speed and turning rate. In practice, however, it has been found that the SLIP feedback matrices $K$ for straight-ahead running provide asymptotically stable ToF control in the SLIP model across a wide range of turn rates.

More specifically, the controls for running are modified as follows. Given a ToF state $x_n$ with principal heading $\Sigma_n$, left foot controls are modified away from their optimized value $u^*_L$ by

$$u_L = u^*_L + K(\Sigma_n^T x_n - x^*)$$

while right foot controls are modified away from their optimized value $u^*_R$ by

$$u_R = u^*_R + K(E \Sigma_n^T x_n - x^*).$$

These control laws can be seen to generalize those applied in straight-ahead running (Eq. 5.30 and Eq. 5.31).

### 6.4 Characterization of Optimized Trajectories

The previous section has presented a 3D-SLIP turn optimization that enforces a change in velocity direction from step-to-step and maintains the alternating lateral velocity at each ToF as in forward running. This design provides periodicity of the velocity in local coordinates for continuous turning. However, it may be surprising that the global position of the 3D-SLIP model is periodic for special choices of $\Delta \gamma$. This section characterizes the position trajectories of the 3D-SLIP model when nominal controls $u_L$ and $u_R$ for periodic turning from the previous section are employed.
6.4.1 The Sweep Angle of the 3D-SLIP Trajectory Over Each Gait Cycle

Given a desired turn angle $\Delta \gamma$ per step, and forward velocity $v_x$, let $x^*$ be the associated ToF state for forward running, with turning controls $u_L$ and $u_R$ obtained through Eq. 6.12. Without loss of generality, let us suppose that the 3D-SLIP model starts at ToF when $t = 0$ in state

$$\mathbf{p}_c(0) = [0, 0, h^*]^T \quad \text{and}$$

$$\mathbf{p}_c(0) = [v_x, v_y^*, 0]^T.$$  \hfill (6.22)

Under application of these controls, let $T$ be the time for one full gait cycle. One gait cycle is defined from ToF through both left and right foot steps, and ending at the subsequent ToF. Regardless of the path that the model takes from $t = 0$ to $t = T$, $\dot{\mathbf{p}}_c(T)$ is related to $\dot{\mathbf{p}}_c(0)$ as

$$\dot{\mathbf{p}}_c(T) = R_s(2\Delta \gamma) \dot{\mathbf{p}}_c(0)$$  \hfill (6.24)

as ensured by Eq. 6.11. This relationship is shown in Figure 6.5(a).

As shown in Figure 6.5(b) however, the trajectory can be related to the sweep of an angle $2\Delta \gamma$ about a circle. The circle can be constructed as follows, namely by requiring $\Delta \mathbf{p}_c = \mathbf{p}_c(T) - \mathbf{p}_c(0)$ to be a chord of a circle, with chord angle $2\Delta \gamma$. This requires the circle to have radius $r$ which satisfies

$$\|\Delta \mathbf{p}_c\| = 2r \sin(\Delta \gamma).$$  \hfill (6.25)

By requiring the circle to pass through both $\mathbf{p}_c(0)$ and $\mathbf{p}_c(T)$, the center of the circle can be determined. Geometrically, there are two such circles that satisfy these conditions, where the turn direction can be used to select the proper turn center.
Thus, despite enforcing a turn angle on velocity during turn optimization, there is always a circle of radius $r$ (which is different for each forward speed and turn rate) wherein the optimized trajectories sweep an angle of $2\Delta \gamma$ through the circle in each gait cycle.

### 6.4.2 ToF Positions For a Continuous Turn

The arguments in the previous subsection can be combined with properties of the velocities during continuous turning in order to characterize the ToF positions. When $u_L$ and $u_R$ are applied repeatedly, it follows from their optimization that

$$\dot{p}_c(t + kT) = R_a(2k\Delta \gamma) \dot{p}_c(t), \quad \forall t \geq 0, k \in \{0, 1, 2, \ldots\}.$$
Figure 6.6: Evolution of ToF positions after each gait cycle through continuous turning.

The position of the SLIP model after \( N \) full gait cycles can be obtained by integrating the velocity as:

\[
p_c(NT) = p_c(0) + \int_0^{NT} \dot{p}_c(t) \, dt.
\] (6.26)

However, due to the periodicity of the velocity in local coordinates:

\[
p_c(NT) = p_c(0) + \sum_{k=0}^{N-1} \int_0^T R_s(2k\Delta\gamma) \dot{p}_c(t) \, dt
\] (6.27)

\[
= p_c(0) + \sum_{k=0}^{N-1} R_s(2k\Delta\gamma) \left( \int_0^T \dot{p}_c(t) \, dt \right)
\] (6.28)

\[
= p_c(0) + \sum_{k=0}^{N-1} R_s(2k\Delta\gamma) \Delta p_c.
\] (6.29)

This evolution is shown graphically in Figure 6.6. Through this construction, it is clear when \( 2N\Delta\gamma = 2\pi \) for some positive integer \( N \), the ToF positions after each gait cycle will correspond to the vertices of a regular polygon, and thus the ToF positions will be periodic. To see this mathematically, it is noted that when \( 2N\Delta\gamma = 2\pi \) the sum in Eq. 6.29 can be simplified to
\[ \sum_{k=0}^{N-1} R_z(2k\Delta \gamma) = \sum_{k=0}^{N-1} \begin{pmatrix} \cos \left( \frac{2\pi k}{N} \right) & -\sin \left( \frac{2\pi k}{N} \right) & 0 \\ \sin \left( \frac{2\pi k}{N} \right) & \cos \left( \frac{2\pi k}{N} \right) & 0 \\ 0 & 0 & 1 \end{pmatrix} \]

\[ = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \]

which follows from the fact that the \( N \) complex \( N \)-th roots of unity sum to 0.\(^{14}\) Thus, due to this simplification, \( \mathbf{p}_c(NT) = \mathbf{p}_c(0) \), and \( \mathbf{p}_c(t) \) is \( NT \) periodic. More generally, it also holds that for a positive integer \( N \), \( \mathbf{p}_c(t) \) is \( NT \) periodic if there exists a positive integer \( M \) such that \( 2N\Delta \gamma = 2M\pi \).

### 6.5 Application for Humanoid Control

Control of the humanoid model is again accomplished with the combination of a State Machine coupled with a Task-Space controller. The main characteristics of the Turning State Machine are similar to straight-ahead running. Namely, CoM and centroidal angular momentum control are applied during support to control balance, while foot position and orientation control is applied throughout to track hand-designed foot trajectories. The main design change here is that a number of the control targets are judiciously selected to occur w.r.t. the \( e'_x \), \( e'_y \), \( e'_z \) coordinate system. These control targets are then realized through a prioritized task-space controller that selects joint torques and ground reaction forces.

\(^{14}\)The \( k \)-th of the complex \( N \)-th roots of unity is given as \( z_k = e^{2\pi k/N} \), where \( i = \sqrt{-1} \) is the complex unit. The sum of the \( N \) complex \( N \)-th roots of unity are given by \( z_0^N + z_1^N + \ldots + z_N^{N-1} \). These form a geometric progression with sum given by \( \frac{1-z_N^N}{1-z_1^N} \), which is zero since \( z_1^N = 1 \). Expanding the real and imaginary parts of the sum of the roots provides corresponding zero sum identities for \( \sin(2\pi k/N) \) and \( \cos(2\pi k/N) \).
6.5.1 Control During Support

Task control during each support state is comprised of CoM, centroidal angular momentum, foot, and pose control as in Chapter 5. Flight foot trajectories are generated online using cubic splines for the position of the foot relative to the CoM. By applying this control with respect to the rotating auxiliary SLIP coordinate system, leg trajectories for forward running are able to be applied to the case of the turn.

CoM and centroidal angular momentum control is again applied to maintain balance. CoM control is carried out with respect to the ICS, while centroidal angular momentum is controlled about the auxiliary SLIP coordinates. Centroidal angular momentum control is applied with a setpoint,

$$k_{G,d} = [0, 0, I_{zz} \omega]^T,$$

where $I_{zz}$ is the system’s net moment of inertia about the $e'_z$ axis, and $\omega$ is computed from $\Delta \gamma$ and the gait period. Since the pitch angular momentum is not well regulated during a run, angular momentum control is only applied with respect to the $e'_x$ and $e'_z$ coordinate axes.

Finally, pose control is applied with a low task weight to promote a natural configuration of the system. As the main difference during the turn, a rolled orientation set point of the torso relative to $e'_x$ is selected to provide a banked turn. The banked roll angle is chosen based on the roll angle of the humanoid virtual leg found from 3D-SLIP optimization. Proper selection of this bank angle keeps the hip angles close to kinematically centered, which provides maximum kinematic availability to respond to any disturbances.
6.5.2 Control During Flight

Task control during flight is largely the same as in support with two main differences. First, CoM and centroidal angular momentum control are disabled. Secondly, foot trajectory servos relative to the CoM in local coordinates are applied for both legs in flight. The discrete nature of the 3D-SLIP ToF to ToF dynamics requires one additional modification. As a practical implementation issue, the SLIP simulator is augmented with a continuous heading trajectory generator based on a cubic spline. This cubic spline is initialized to match the angle of the discrete rotations of the $e'_x$, $e'_y$, $e'_z$ system that change at ToF. The use of this spline prevents discrete changes in the control objectives at ToF.

6.6 Results

6.6.1 Continuous Turning

Continuous turning with a forward speed of 3.5 m/s and a turn rate of $\omega = 0.4$ rad/s is highlighted here and in the video attachment to the conference paper version of this chapter [141]. The video is also available online at:


The foot positions applied by the humanoid for this turn are shown in Figure 6.7. During a straight-ahead portion before the turn, the feet are placed directly in front of the hips. When the turn begins, touchdown angles pre-optimized as $u_L$ and $u_R$ for this turn rate are used to execute the turn. Both feet are placed towards the outside of the turn in order to generate additional centripetal force in comparison to forward running. It can be seen that curvature of the CoM trajectory during a right foot
support is much higher than during a left foot support. In this sense, the right leg is doing more work to accomplish the turn than the left leg.

To understand this effect further, the lateral ground reaction forces (GRFs) were compared between straight-ahead running and during the turn. These GRFs are plotted in Figures 6.8 and 6.9. In Figure 6.8 for straight-ahead running, the lateral force alternates from positive in one step to negative in the next. This pattern gives rise to lateral CoM sway and enables foot placement in front of the hips. When transitioning to the turn, with forces in Figure 6.9, the force patterns in the forward direction are largely unchanged. However, in the lateral direction, both feet provide additional force towards the center of the turn in comparison to forward running. Although the left foot produces much less centripetal force to execute the turn, it is kinematically unavailable to produce the same forces as the right foot. That is, to execute a left turn, centripetal forces are generated by placing the foot to the right of the CoM. Since it is kinematically difficult for the left foot to be placed to the right of the CoM, about the best that the left leg can do is to get its foot underneath the CoM to support the weight of the humanoid. These different roles that the inside and outside legs play are not captured in studies of monopod steering which would employ the same foot positioning relative to the CoM at each step. The footstep locations for such a turn would not be kinematically reachable by the humanoid.

The plots in Figure 6.10 show how this foot placement varies as speed and turn rate are increased. Simple physics dictates that the centripetal force required for the turn will scale proportional to the product $v_x \omega$. These trends are observed in the foot placements as speed or turn rate is increased. With higher speed or turn rate, the touchdown angles are optimized to place the feet further toward the outside of
Figure 6.7: Foot positioning used by the humanoid for a running turn at 3.5 m/s with a turning rate of $\omega=0.4$ rad/s. The nominal path for the turn is a circle with a radius that is approximately 1/4 that of the inside-most lane of a 400m track. Both feet are placed towards the outside of the turn to generate the inward radial force required to execute the turn. However, radial force production is different for each leg, as lateral sway about the nominal path enables all footholds to remain kinematically reachable.

6.6.2 Online Steering with Different Turn Rates

The design approach taken in Section 6.3 allows the humanoid turn rate to be modified online from step-to-step without any additional required 3D-SLIP optimization. To demonstrate the ability to compose trajectories of different turn rates, the extreme case of switching turn direction in a single step is illustrated in the video supplement (http://www.go.osu.edu/Wensing_Orin_IROS2014).
Figure 6.8: Force profiles for 3.5 m/s running with the 3D-SLIP template. The two-step periodic nature of the lateral force provides a two-step period to the lateral CoM dynamics. The CoM enters the left step with a lateral velocity toward the left foot (in the $e'_y$ direction). As a result of the left foot force in support, this velocity is reversed in sign by the following LO.

In this demonstration, the humanoid runs at 4.0 m/s with a turn rate of $\omega = 0.3$ rad/s. Following the execution of a partial turn, the commanded turn rate is changed in sign, and a new target CoM trajectory is generated online based on this new command. Actual rate commands to the 3D-SLIP are delayed by one gait cycle in order to allow the banked torso roll to prepare for the upcoming change in leg angles. The tracking of the torso roll command is shown in Figure 6.11 and successfully re-aligns the torso to the new leg angles within a few steps. The use of a soft PD controller on the torso prevents rapid upper body movements which would require excessive forces and potentially destabilize the system.

While it is not a main focus of this chapter, the video also showcases the ability of the 3D-SLIP controller to generate recovery trajectories online in response to push
disturbances. These disturbances can occur in any direction (in or out of the sagittal plane), although disturbances directed towards the center of the turn are more easily handled than those to the outside of the turn. This is an intuitive result, as proper execution of the turn depends largely on the production of centripetal force, so any disturbance towards the inside makes the turn easier to execute.

### 6.7 Summary

This chapter has presented new methods which enable online high-speed humanoid turning. By starting from top of flight (ToF) states from straight-ahead running with leg separation, the trajectories for turning allow each leg to play a unique role during the turn. While this leads to asymmetric radial force production by the inner
Figure 6.10: Foot positioning comparison for different turn rates and running speeds. The required inward radial force to execute the turn is proportional to the product $v_x \omega$. As a result, the lateral foot placement becomes more extreme for higher turning rates and running speeds.
Figure 6.11: Torso roll during a left turn to right turn transition. A new torso orientation is commanded one full gait cycle before the turn rate change to prepare the torso for the upcoming footstep repositioning.

and outer leg, the centripetal force required for each leg is consistent with its kinematic ability to produce force in that direction. Without these new features of SLIP trajectories, as enforced by the SLIP optimization proposed here, the previous work on SLIP steering would not be capable for integration with a whole-body humanoid controller.

With a few judicious modifications to the whole-body control architecture for straight-ahead running, the 3D-SLIP trajectories here can be followed by the humanoid. The design choices applied allow the trajectories to be smoothly combined, which gives the humanoid the ability to change turn rates online from step-to-step,
and to recover from push disturbances. The 3D-SLIP model has allowed for straight-ahead running motions to be composed with running turns within a common framework. The following chapter will describe further application of this principle to produce a human running long jump through the introduction of an extended 3D-SLIP model that is specific to this new motion.
Chapter 7

Extensions to the 3D-SLIP for Running Long Jumps

7.1 Introduction

This chapter describes a series of extensions to the 3D-SLIP model which make it an appropriate template model for a running long jump. It is shown that trajectory optimization with this template, along with the whole-body task-space control, enables the humanoid to perform a running long jump. The running long jump is able to be performed with a dynamically-balanced landing transition back into high-speed running. Snapshots from the resultant motion are shown in Figure 7.1. This result represents the first time that a running long jump has been demonstrated in simulation without the use of time-intensive off-line whole-body trajectory optimization techniques.

While starting from a high-speed run, the running long jump includes a more complex thrust step in comparison to the support step during forward running. In order to produce a maximum span running long jump, the humanoid must coordinate leg forces to produce significant vertical velocity, often at the expense of slowing down its forward motion. This trade-off is further complicated by leg kinematics which limit the stroke length of the thrust step, requiring an intricate synchronization of
the leg kinematic movement with the leg’s propulsion of the Center of Mass (CoM). Additionally, a significant underactuated ballistic flight period places importance on the state of the system at liftoff.

In robotics research, very little work has addressed the performance of a running long jump. Krasny and Orin [64] studied the optimization of a running long jump for galloping quadrupeds through the use of a genetic algorithm (GA). The GA evaluated the performance of potential controllers through whole-body dynamics simulation, and required hours of computing time to find a suitable solution. Dellin and Srinivasa [19] presented a framework for whole-body dynamic locomotion which included CoM planning based on motion capture template trajectories along with time intensive whole-body trajectory optimization. The method described here is the first which does not require whole-body optimization off line by relying on a biologically inspired template model to plan optimal CoM trajectories. Optimization with this
template model addresses the complex leg synchronization required for performance of an optimal running long jump.

The remainder of this chapter is organized as follows. Section 7.2 develops a 3D long jump template model which is based on a simpler planar model studied in biomechanics [124]. While the dynamics of takeoff and landing in a running jump share elements in common with high-speed running, the CoM dynamics experience a notable difference at the beginning of support, as shown in Figure 7.2. In this period, a large force peak contributes an initial vertical impulse to assist in the production of a maximum jump span. Section 7.3 presents a principled optimization-based approach to generate trajectories with a new 3D model that captures this effect. The results share many key characteristics with human long jump data. The whole-body humanoid control approach is summarized in Section 7.4 which enables the performance of a running long jump in 3D dynamic simulation. Concluding remarks are provided in Section 7.5.

7.2 Extended 3D-SLIP Template for Long Jump Optimization

While the standard 3D-SLIP model of locomotion can be used to describe the CoM dynamics of running with good fidelity, it lacks the capability to describe important characteristics of a running long jump. Studies by Seyfarth et al. [124] have shown that human long jumpers employ a lengthened leg at takeoff in comparison to touchdown. In addition, human long jump data displays a passive ground reaction force (GRF) peak at the beginning of support. Two modifications to the 3D-SLIP model are described below which provide the ability to capture these characteristics.
Figure 7.2: Experimentally observed vertical ground reaction forces and leg stiffness for a human long jump, from [124]. The passive peak at the beginning is attributed to a forceful foot plant which decelerates the leg mass.

7.2.1 Leg Extension 3D-SLIP Model

In order to capture the leg lengthening effects found in human long jumping, active lengthening may be added to the 3D-SLIP model used in Chapters 5 and 6. Here, this lengthening is modeled through the incorporation of a linear actuator in series with the SLIP spring. This modification is shown in Figure 7.3(a). The dynamics of this model are similar to the 3D-SLIP model, and also undergo phases of support and flight.

The dynamic evolution of the point mass in this extended model is still intended to describe the CoM of a larger system, while the anchor point represents the Center of Pressure (CoP) below the support foot. Thus, $p_c \in \mathbb{R}^3$ continues to denote position of the point mass, with the anchor point of the model denoted as $p_f \in \mathbb{R}^3$. Letting $r = \|p_c - p_f\|$ and $r_0$ its value at TD, the spring imparts forces along the leg with
Figure 7.3: (a) Leg extension 3D-SLIP (b) and long jump 3D-SLIP models to capture the salient requirements of producing a maximum-span running long jump. Following touchdown (TD) both models include a linear spring with stiffness $k_s$ that is driven by a linear actuator. Similar to the traditional 3D-SLIP model, ground contact terminates at liftoff (LO) when the spring reaches its rest length. The long jump 3D-SLIP model employs a main mass $m_1$ and small mass $m_2$ which together equal the total mass $m$. Mass 2 is attached to the spring leg at fixed radius $r_2$ via a nonlinear spring-damper. This mass is initialized along the leg at touchdown, but drifts off the leg throughout support.
magnitude:

\[ f_s = k_s \left( r_0 + \ell_a - r \right) , \]  

(7.1)

where \( \ell_a \) is the length of the linear actuator, assumed to be zero at TD. When \( \ell_a \) is identically 0, the leg extension 3D-SLIP model is reduced to the standard 3D-SLIP model. The spring force in Eq. 7.1 provides the following support dynamics for the mass

\[ \ddot{p}_c = \frac{f_s}{m} \dot{r} + g \]  

(7.2)

where \( \dot{r} \) is the unit vector along the SLIP leg and \( g \) is the gravity vector. Throughout this chapter, a simple law is assumed for the actuator. Motivated by human data in the sagittal plane [124], its length is chosen proportional to the angular excursion of the SLIP leg projected onto the sagittal \((x, z)\) plane. The angle of the SLIP leg from the vertical in the \((x, z)\) plane is given by:

\[ \theta_c = \text{atan2} \left( r \hat{x}^T \hat{r}, r \hat{z}^T \hat{r} \right) , \]  

(7.3)

with \( \theta_{c,0} \) defined to be its value at touchdown. From this definition, the actuator length is assumed to take the form

\[ \ell_a = b_a \left( \theta_c - \theta_{c,0} \right) \]  

(7.4)

where \( b_a \) is a positive constant of proportionality to be tuned through optimization in Section 7.3. Forward running jumps are dominated by motion in the positive \( x \)-direction, and thus \( \theta_c \) and \( \ell_a \) are monotonically increasing.

### 7.2.2 Long Jump 3D-SLIP Model

In addition to active lengthening, human long jumpers plant their support foot more forcefully prior to takeoff, resulting in a large force peak that decelerates the
support leg mass. This force peak injects additional vertical impulse, which is a key requirement of jumping for distance. The foot plant behavior can be introduced through the addition of a second mass to the Leg Extension SLIP model, shown in Figure 7.3(b). In this model, the mass \( m_2 \) represents the support leg mass, and is connected to the SLIP leg at a fixed distance \( r_2 \) through a nonlinear spring-damper. The large mass \( m_1 \) is positioned at radius \( r_1 \) and is still driven by the SLIP spring. Similar to before, the spring produces forces

\[
\mathbf{f}_s = k_s (r_{1,0} + \ell_a - r_1) \tag{7.5}
\]

directed along \( \hat{r} \). These forces act only on \( m_1 \). To describe the position of mass \( m_2 \), coordinate vectors \( \hat{x}_2 \) and \( \hat{y}_2 \) are attached to and rotate with the SLIP leg. Combined with \( \hat{r} \) these provide a coordinate system with rotation matrix

\[
\mathbf{R} = \begin{bmatrix} \hat{x}_2 & \hat{y}_2 & \hat{r} \end{bmatrix} \tag{7.6}
\]

Letting \( \mathbf{p}'_2 \) be the position of mass \( m_2 \) relative to this frame, its position in inertial coordinates is given as

\[
\mathbf{p}_2 = r_2 \hat{r} + \mathbf{R} \mathbf{p}'_2. \tag{7.7}
\]

Again motivated by the planar case, a nonlinear spring-damper force is applied to mass \( m_2 \) in order to return it to position \( r_2 \hat{r} \) in inertial coordinates. A locally coordinate-decoupled force is assumed given by

\[
\mathbf{f}_2 = \mathbf{R} \begin{bmatrix} -(k_2 p'_{2,x} + d_2 \dot{p}'_{2,x}) \cdot |p'_{2,x}|^c_2 \\ -(k_2 p'_{2,y} + d_2 \dot{p}'_{2,y}) \cdot |p'_{2,y}|^c_2 \\ -(k_2 p'_{2,z} + d_2 \dot{p}'_{2,z}) \cdot |p'_{2,z}|^c_2 \end{bmatrix}, \tag{7.8}
\]

where \( k_2, d_2 \) and \( c_2 \) are constants. The inclusion of nonlinear effects is justified by considering the dynamics that the additional mass is intended to capture; when
planted forcefully, the foot experiences ground contact forces which can be described well with nonlinear contact models \cite{79}. Fixed selections for $r_2$, $k_2$, $d_2$, and $c_2$, are given in Table 7.1. The value of $r_2$ in this table is chosen to be approximately one quarter of the extended virtual leg length (0.975 m) of the humanoid considered in Section 7.4. Masses $m_1$ and $m_2$ represent 80% and 20% of the mass of the humanoid.

Spring-damper constants for a planar long jump model running at 8.2 m/s \cite{124} were tuned to produce values for $k_2$, $d_2$, and $c_2$ which resulted in similar passive force peaks for speeds in the range of 4 m/s to 6 m/s with this model. While fixed here, further adaptation of these parameters could provide benefits to jump span. In the following section, an optimization-based approach is presented to select the rest of the parameters for the Long Jump 3D-SLIP model.

### Kinematics and Dynamics of the Long Jump 3D-SLIP Model

During support, simulation of the Long Jump 3D-SLIP model is complicated by the spring-damper forces on the second mass. Due to its nonlinear effects, the force $f_2$ may in fact create moments around the SLIP leg that must be countered by constraint moments at the anchor point. This characteristic introduces additional complexities in order to model the 2 DoF joint at $p_f$ between the ground and the SLIP leg.

<table>
<thead>
<tr>
<th>$r_2$</th>
<th>$k_2$</th>
<th>$d_2$</th>
<th>$c_2$</th>
<th>$m_1$</th>
<th>$m_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25 m</td>
<td>14.6 kN/m</td>
<td>$10^{8.5}$ N·s / m</td>
<td>2.8</td>
<td>58.1 kg</td>
<td>14.5 kg</td>
</tr>
</tbody>
</table>

Table 7.1: Fixed parameter values for Long Jump 3D-SLIP dynamics.
Turning first to kinematics, the configuration of the Long Jump 3D-SLIP Model can be specified uniquely by $r_1$, $p'_2$, and $R$ (which includes $\hat{r}$). The rotational configuration $R$ can be stored using a unit quaternion $\epsilon$

$$\epsilon = [\epsilon_0 \quad \epsilon_1 \quad \epsilon_2 \quad \epsilon_3]^T,$$

with $\|\epsilon\| = 1$, where a common formula [136, Table 1.1] relates $\epsilon$ to $R$. The velocity of the system can be similarly specified by $\dot{r}_1$, $\dot{p}'_2$, and $\omega$. $\omega$ provides the angular velocity of $R$ in inertial coordinates

$$\dot{R} = \omega \times R,$$

and can be converted to quaternion rates through the standard equation [136]

$$\dot{\epsilon} = \frac{1}{2} \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \epsilon_0 & \epsilon_3 & -\epsilon_2 \\ -\epsilon_3 & \epsilon_0 & \epsilon_1 \\ \epsilon_2 & -\epsilon_1 & \epsilon_0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}.$$  

(7.11)

The 2 DoF rotational joint at $p_f$ places a restriction on the admissible values of $\omega$. To model this joint, it is again noted that $p_f$ is intended to represent the center of pressure (CoP) below the foot of the humanoid. Due to this requirement, any constraint moments at the base of the SLIP leg should exist only in the direction normal to the ground plane. Yet, since the constraint moments can do no work under virtual displacements consistent with the free modes of motion, it follows that the angular velocity $\omega$ must take the form $\omega = [\omega_x, \omega_y, 0]^T$.

One might suspect that the configuration parameterization of $R$ with quaternions could be replaced with a simpler 2-tuple parameterization, such as a pair of Euler angles. However, the restriction $\omega_z = 0$ imposes a non-holonomic constraint on the system, which precludes the use of any 2-element parameterization for $R$. To see this,
with $\omega_z = 0$ the quaternion rates in Eq. 7.11 take the restricted form
\begin{equation}
\dot{\epsilon} = \frac{1}{2} \begin{bmatrix}
-\epsilon_1 \\
\epsilon_0 \\
-\epsilon_3 \\
\epsilon_2 \\
\end{bmatrix} \omega_x + \frac{1}{2} \begin{bmatrix}
-\epsilon_2 \\
\epsilon_3 \\
\epsilon_0 \\
-\epsilon_1 \\
\end{bmatrix} \omega_y. \quad (7.12)
\end{equation}

Thus, vector fields $g_1(\epsilon)$ and $g_2(\epsilon)$ can be defined such that
\begin{equation}
\dot{\epsilon} = g_1 \omega_x + g_2 \omega_y. \quad (7.13)
\end{equation}

It follows from Frobenius’ theorem [91] that the constraint $\omega_z = 0$ is holonomic if and only if the Lie bracket of $g_1$ and $g_2$, given as
\begin{equation}
[g_1, g_2](\epsilon) = \frac{\partial g_2}{\partial \epsilon} g_1(\epsilon) - \frac{\partial g_1}{\partial \epsilon} g_2(\epsilon), \quad (7.14)
\end{equation}
satisfies $[g_1, g_2] \in \text{span} \{g_1, g_2\}$. Forming
\begin{equation}
[g_1, g_2] = \frac{1}{2} \begin{bmatrix}
\epsilon_3 \\
\epsilon_2 \\
-\epsilon_1 \\
-\epsilon_0 \\
\end{bmatrix} \quad (7.15)
\end{equation}
and noting the similarity of $[g_1, g_2]$ to the third column of the rate matrix in Eq. 7.11, it is verified that $[g_1, g_2] \notin \text{span} \{g_1, g_2\}$, and thus the constraint $\omega_z = 0$ is nonholonomic. Therefore, despite the presence of only 2 DoFs at $p_f$, a full parameterization of $SO(3)$ is required to store $R$, and is accomplished through quaternions here.

Following these conventions, standard Newton and Euler equations can be used to derive the dynamics of the system, which are summarized in Table 7.2. The equation for $[\dot{\omega}_x, \dot{\omega}_y]^T$ arises from equating the net moment on the SLIP leg (about $p_f$) to its rate of change in angular momentum about $p_f$. The equations for $\dot{\dot{r}}_1$ and $\dot{\dot{p}}'_2$ can be derived through a second differentiation of the kinematic equations and use of Newton’s second law.
**Kinematics**

\[
p_1 = r_1 \hat{r} \quad \dot{p}_1 = \dot{r}_1 \hat{r} + \omega \times p_1 \\
p_2 = r_2 \hat{r} + R \dot{p}_2' \quad \dot{p}_2 = \omega \times p_2 + R \dot{p}_2'
\]

**Dynamics**

\[
\beta := m_1 p_1 \times g - p_2 \times f_2 - m_1 p_1 \times (2 \dot{r}_1 \omega \times \hat{r} + \omega \times \omega \times p_1) \\
I := m_1 S(p_1) S(p_1)^T \\
\begin{bmatrix}
\dot{\omega}_x \\
\dot{\omega}_y
\end{bmatrix} = \begin{bmatrix} I_{xx} & I_{xy} \\
I_{yx} & I_{yy} \end{bmatrix}^{-1} \begin{bmatrix} \beta_x \\
\beta_y \end{bmatrix} \\
\ddot{r}_1 = \frac{f_x}{m_1} + \dot{r}_1^T g - \dot{r}_1^T (\omega \times \omega \times p_1) \\
\ddot{p}_2' = R^T \left( \frac{f_2}{m_2} + g \right) - R^T (\dot{\omega} \times p_2 + \omega \times \omega \times p_2 + 2 \omega \times R \dot{p}_2')
\]

Table 7.2: Equations of motion for the Long Jump 3D-SLIP model.

### 7.3 Long Jump Optimization

Given an initial state of the CoM, this section provides an optimization-based method to produce a reference CoM trajectory for a maximum-span long jump. As shown in Figure 7.4, the span \(s\) of a long jump is considered from footprint to footprint, across the thrust and landing steps. The total span is comprised of three components, two on the ground and a main component in flight. The following subsections present a staged optimization process which first generates a thrust step, and then subsequently generates a landing step.
Figure 7.4: The span $s$ of the long jump is broken into three components: a) the takeoff span, measured from the center of the support foot to the horizontal position of the CoM at liftoff, b) the flight span, measured from the CoM at liftoff to the CoM at the subsequent touchdown, and c) the touchdown span, again measured from CoM to the center of the foot.

7.3.1 Thrust Step Optimization

Starting from a steady-state run, the thrust step of the long jump serves to generate a large vertical CoM velocity before takeoff. Generation of this vertical velocity requires proper selection of the leg’s touchdown configuration as well as proper coordination of the leg’s stroke during support. These details are first coordinated on the Long Jump 3D-SLIP model before being mapped to the humanoid as described in Section 7.4. The touchdown span (© in Figure 7.4) does not vary greatly across long jumps, and thus was considered fixed when optimizing the thrust step.

The optimization approach used here applies constrained nonlinear optimization methods, where dynamic simulation of the Long Jump 3D-SLIP is applied to evaluate the objective (jump span) and constraints. The optimization variables and constraints
are summarized in Table 7.3 with further details provided below. Simulation of the Long Jump 3D-SLIP model is carried out starting at the top-of-flight (TOF) before thrust, continuing through support, and ending at the subsequent TD. Assuming an approach speed of $v_x$ in the forward direction, the TOF CoM State is given from steady-state forward running as optimized in Chapter 5. For a fixed approach speed the long jump optimization problem is given in general form as

$$\begin{align*}
\max_x & \quad s(x) \\
\text{s.t.} & \quad g(x) \leq g(x) \leq \bar{g} \\
& \quad \underline{x} \leq x \leq \bar{x}
\end{align*}$$

(7.16)

(7.17)

(7.18)

where $x = [\theta, \phi, \ell_{TD}, k_s, b_a]^T$ is the vector of optimization variables, $s(x)$ provides the simulated span, and $g(x) \in \mathbb{R}^4$ contains simulated results for the constrained quantities in Table 7.3. The markings $\cdot$ and $\cdot$ represent upper and lower bounds. Leg touchdown angles $\theta$ and $\phi$, and a touchdown leg length $\ell_{TD}$ specify the SLIP foot
position in flight

\[
p_f = p_c + p_{\text{hip}} + \ell_{TD} \begin{bmatrix}
\sin(\theta) \cos(\phi) \\
-\sin(\theta) \sin(\phi) \\
-\cos(\theta)
\end{bmatrix}
\] (7.19)

where \( p_{\text{hip}} \) approximates the vector between the CoM and the hip of the next support leg. As in Chapter 5, the TD angles \( \theta, \phi \) follow a fixed \(-y, z\) Euler angle convention, and roughly control the forward and lateral leg angles respectively. Given selections for \( \theta, \phi, \) and \( \ell_{TD} \), the state of the CoM at TD can be computed from ballistic physics without simulation. At TD, the CoM state is then mapped to the initial state of the Long Jump 3D-SLIP with mass \( m_2 \) along the leg.\(^{15}\)

During support, the behavior of the Long Jump 3D-SLIP model is influenced by its leg stiffness \( k_s \), and its actuator lengthening \( b_a \). Improper selection of \( k_s \) or \( b_a \) often results in trajectories that are not transferable to the humanoid. For instance, selection of a small spring stiffness may result in a soft landing with too low of a CoM height, while a high spring stiffness may result in GRFs that are infeasible to be generated by the humanoid. Similarly, a high leg lengthening constant \( b_a \) may result in a CoM position that is beyond the kinematic limits of the humanoid. A limit on \( \|p_c - p_f\| \) at LO is set to approximate this constraint. As a final constraint, the lateral velocity at LO is constrained to be zero, in order to prevent lateral motion in flight.

Following support, ballistic motion is assumed for the CoM and allows for the flight span (\( \Box \) in Figure 7.4) to be computed. A fixed touchdown CoM height, taken from steady-state running, is considered across all simulations. MATLAB’s

\(^{15}\)The assumption that mass \( m_2 \) is along the leg in flight allows only two parameters \( \theta, \phi \) to describe the orientation of the leg. At touchdown, the leg orientation is converted to quaternions for simulation of the support leg orientation dynamics which evolve on \( SO(3) \).
Figure 7.5: Simple model trajectories in the sagittal plane for an optimized long jump following a 6 m/s run. Dashed trajectories for $p_c$ indicate flight. The model begins with a steady-state run, experiences an explosive thrust step, and then employs a single recovery step to return to periodic motion.

$fmincon$ with its active-set algorithm was used to solve Eqs. 7.16-7.18 and generate the maximum-span reference motion.

### 7.3.2 Landing Step Optimization

The landing step optimization develops a trajectory of the Long Jump 3D-SLIP model that starts in flight, after the optimized thrust step, and returns to steady-state running at the subsequent TOF. Once complete, a continuous reference CoM trajectory is available, starting from and ending at the CoM motion for running as shown in Figure 7.5.

The optimization process to generate a landing step largely mirrors that of the thrust step, with the main exception being its objective. Simulation occurs from the TOF following thrust, continues through the landing step, and terminates at the subsequent TOF. Landing optimization employs the same variables and bounds in
Table 7.4: Optimized Long Jump 3D-SLIP parameters for a 6 m/s approach

<table>
<thead>
<tr>
<th>Step</th>
<th>$\theta$</th>
<th>$\phi$</th>
<th>$\ell_{TD}$</th>
<th>$k_s$</th>
<th>$b_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thrust</td>
<td>28.8°</td>
<td>−12.3°</td>
<td>0.89 m</td>
<td>16.5 kN/m</td>
<td>0.16 m/rad</td>
</tr>
<tr>
<td>Landing</td>
<td>24.8°</td>
<td>−14.4°</td>
<td>0.97 m</td>
<td>14.4 kN/m</td>
<td>−0.01 m/rad</td>
</tr>
</tbody>
</table>

Table 7.3. The constraint on $\|\mathbf{p}_c - \mathbf{p}_f\|$ is replaced with a lower bound of 0.85 m to prevent an excessively shortened leg at TD. The lateral LO velocity is unconstrained, and all other constraints remain the same. Rather than maximizing span, the landing step optimization minimizes the squared error between the achieved CoM TOF state following landing and that of steady-state running. Again, MATLAB’s `fmincon` was used to optimize the SLIP touchdown and stroke parameters. With this approach, optimized trajectories for landing were able to perfectly match the TOF CoM state for steady-state running. Optimized variables $\mathbf{x}$ for the thrust and landing steps are shown in Table 7.4, and result in a 4.12 m long jump at a 6 m/s approach.

### 7.3.3 Optimization Results

The long jump trajectories produced through this process exhibit characteristics common to human long jumpers. The CoM velocity for an optimized long jump with a 6 m/s approach speed is shown in Figure 7.6. As noted on the figure, the thrust step results in a net loss in forward velocity. This slight decrease in forward velocity from 0 – 170 ms is accompanied by a large increase in vertical velocity. This exchange of velocity components has been conceptualized as a “pivot” [40] in the biomechanics literature and is inherent to the SLIP-based model used here. At higher speeds, human long jumpers may lose over 1.0 m/s in forward velocity in order to achieve additional span [43].
Figure 7.6: CoM velocity for an optimized long jump following a 6 m/s run. During the thrust period before the long jump, the system loses forward velocity ($v_x$) through a pole-vault action over its support leg that generates vertical velocity ($v_z$). Dashed segments indicate periods of ballistic flight.

All of the constraints placed on the Long Jump 3D-SLIP model are limiting to its performance, and provide insight into the key characteristics of jumping for distance. Figure 7.7 shows trajectories of the CoM height, approximate virtual leg length $\|p_c - p_f\|$, and vertical GRF for the optimum. Constraints on each of these quantities are active. Given the constraint on the vertical GRF, vertical impulse may only be increased through additional time in support. For the optimum, a maximum time in support is achieved by coordinating stiffness and lengthening of the leg to utilize the full allowable range of its stroke. A sensitivity analysis of the optimum reveals the importance of this principle, as the optimum span increases by 9.7 cm per 1 cm of additional length, and by 4.8 cm per 1 cm of additional allowable vertical excursion downward.
Figure 7.7: Active constraints on the optimum long jump. The optimum extends its time on the ground by (a) traveling as low as allowed to allow for maximum stroke and (b) coordinating its leg thrust to achieve maximum leg extension at takeoff. (c) The optimum also applies the maximum allowable vertical GRF, after the initial passive peak, to impart the largest vertical impulse.

Aside from proper leg coordination for maximum stroke, it is important to select proper touchdown angles to maximize the benefit of the “pivoting” or pole-vaulting effect over the support leg. To understand this effect, the long jump optimization was repeated with an additional constraint placed on the angle of the CoM velocity at takeoff relative to the horizontal. As shown in Figure 7.8, this takeoff-velocity angle $\gamma_{LO}$ negatively affects span when too high or too low. Low takeoff angles are obtained by entering the thrust step with the foot largely underneath the body, minimizing
Figure 7.8: Effect of takeoff-velocity angle $\gamma_{LO}$ on jump span. At takeoff angles below the optimum, insufficient vertical velocity is created in support. At takeoff angles higher than the optimum, horizontal velocity is sacrificed for vertical velocity without benefit to span.
any vaulting effect. High takeoff angles are obtained with the foot further in front at
TD, but can lose too much forward velocity.

To illustrate this tradeoff further, takeoff-velocity magnitude is plotted against
takeoff-velocity angle in Figure 7.9. Approximating the flight as having equal takeoff
and landing heights, the flight span for each takeoff state can be computed as $\frac{2}{g} v_x v_y$.
The contours on Figure 7.9 represent takeoff states that result in the same estimated
flight span. Despite this simplification, the graphical optimum for the estimated flight
span is within a degree of the optimum for total span. Note that the maximum at-
tainable takeoff-velocity diminishes with increased takeoff-velocity angle. This effect
is not present in the traditional ballistic shooting problem, where the takeoff-velocity
magnitude is fixed and the optimum takeoff-velocity angle is $45^\circ$. Due to the dimin-
ishing takeoff-velocity effect for the long jump, the optimum takeoff-velocity angle is
much lower. While the importance of takeoff angle has been shown for human long
jumping [72], the optimization constraints imposed here enable similar properties to
emerge from our model. Notice that at lower speeds, the optimal takeoff-velocity
angle shifts to higher values, as additional vertical velocity may be generated with
less detriment to forward speed. This trend agrees with human studies [10].

7.4 Prioritized Task-Space Control of a Humanoid Running
Long Jump

Performance of the running long jump by the humanoid can be accomplished with
an extension to the state-machine-based task-space control approach applied in Chapter 5. The reference CoM motions developed here fit within this framework without
difficulty and allow for the previous task-space approach for running to be applied.
Even for the running long jump, it is emphasized that the task-space controller is
Figure 7.9: Maximal takeoff-velocity versus takeoff-velocity angle for a 6 m/s and 4 m/s running approach. Contours indicate magnitude and angle combinations which produce equivalent estimated flight spans $\frac{2}{g}v_xv_y$. The point with optimal total span for a 6 m/s approach is shown and corresponds to a near optimum of estimated flight span.

It is highlighted that in running, the pitch angular momentum is not well regulated due to the presence of leg cycling in the sagittal plane. Although this angular momentum is not well regulated at the end of a normal running step, its regulation is very important to the proper performance of a running long jump. As a result, during the
thrust step of the running long jump, pitch angular momentum control is activated to dampen any angular momentum and prevent excess body rotation in flight.

7.5 Summary

This chapter has presented new methods to develop a running long jump for humanoid robots. The new Long Jump 3D-SLIP model has shown to produce optimal CoM trajectories with properties that are consistent with human long jumpers. With the constraints imposed on the model, the takeoff-velocity angle has been shown to be a key characteristic for achieving a maximum jump distance. The constrained nonlinear optimization formulation discussed allows for optimal leg coordination and takeoff-velocity angle to be produced automatically for any given CoM approach speed. This strategy allows for the extension of a controller for high-speed running to be applied to control both the thrust and landing steps. While this approach for a running long jump integrates cleanly into the previously applied framework, it is the first time that a running long jump has been performed in simulation without requiring any motion capture or whole-body offline optimization.
Chapter 8

Summary and Conclusions

8.1 Summary

The dynamic capabilities of legged animals have largely remained beyond those of legged machines. In particular, methods to maintain dynamic balance in legged robots have continued to be an elusive goal. The objective of this dissertation was to make a significant contribution towards the development of these dynamic capabilities for humanoid robots. While there are parallel challenges facing the sensing and actuation for these machines, this work has provided a control framework which is general to produce a standing long jump, high-speed run, running turn, and running long jump in 3D dynamic simulation.

The control framework has made broad contributions in the areas of task-space control and simple-model-based control for humanoid robots. At a high-level, the simple-model-based control approach has been shown to be an effective method to manage the most important dynamics of the humanoid, namely its footstep locations and Center of Mass (CoM) trajectories. This approach provides the humanoid with a method to predict the consequences of its footsteps as they pertain to its ability to maintain dynamic balance and continue locomotion. The combination of this
approach with a whole-body task-space controller has been shown to provide appropriate whole-body movements at runtime, without requiring whole-body trajectory optimization. Throughout, control of centroidal momentum has been applied as a key component of task-space control, resulting in emergent upper-body motions to maintain balance in many of the examples studied. New connections between centroidal dynamics and the joint-space dynamics of humanoid robots, provided in Chapter 2, should accelerate the adoption of this approach in future systems.

More specifically, Chapters 3 and 4 have provided task-space control developments that are important for real-time control. The Extended Force Propagator Algorithm given in Chapter 3 has the lowest order of any available algorithm to compute the operational-space (task-space) inertia matrix (OSIM). Through the new computation and application of extended force propagators, which describe the dynamic effects of forces across spans of links, the EFPA is the fastest recursive algorithm available for the OSIM, and outperforms sparsity-based algorithms for many topologies.

In order to solve the task-space control problem, a new conic optimization formulation of prioritized task-space control in Chapter 4 has been shown to outperform a previous quadratic programming formulation in terms of available solution speed. As an additional advantage, the new formulation retains friction cone constraints on the ground reaction forces in their full complexity without sacrificing computational performance. The new formulation has enabled the solution of the prioritized task-space control problem at rates of 200 Hz in this dissertation, and is a main contributor to the overall control system running at real-time rates. The application of task-space control to a standing jump and dynamic kick have shown the utility of performing
centroidal momentum control within this framework, and demonstrated the potential for future extension of this controller to operate on uneven terrain.

The combination of the task-space controller with simple-model-based control has been shown to provide new capabilities for dynamic humanoid locomotion. The use of the 3D-SLIP model for gait generation, introduced in Chapter 5, is a unique feature of this work and provides a stark departure from traditional ZMP-based gait generation methods. The use of the 3D-SLIP model provides automatic footstep placement which allows the humanoid to run at a wide range of speeds, change speeds quickly, and respond to push disturbances. In addition, the use of this simple-model allows the humanoid to make quick decisions about foot placement within real-time constraints. Olympic speed running has been demonstrated, along with a set of extensions which enable high-speed turns. The trajectory generation methods for turning in Chapter 6 naturally capture the unique roles of the inside and outside leg that are required to successfully execute a turn. This feature of the resultant motions has not been studied in previous theoretical work on 3D-SLIP steering.

The generality of the framework to other movements has been shown for an aperiodic running long jump. Through extensions to the 3D-SLIP model to capture important characteristics of a long jump, the work in Chapter 7 shows the ability to clear a 4m gap following a 6 m/s approach speed. The trajectory optimization results for this model display many common techniques with human long jumpers to optimize jump span. Specifically, in order to produce the vertical impulse necessary for maximum span, the system must sacrifice takeoff-velocity magnitude. As a result, the optimum takeoff-velocity angle for the running long jump does not agree with the ballistic optimum of 45° which is intuitively expected. The use and study of the
SLIP model to perform other movements provides rich potential for future humanoid systems.

8.2 Future Work

The progress made in this dissertation has generated a wide range of future directions in which to continue the advance of dynamic humanoid capabilities. A number of potential extensions and improvements to the approaches used here, are described below.

- Within the context of task-space control, recent work by Kuindersma et al. [66] and Escande et al. [23] has shown the potential of warm-started active set methods to outperform conic optimization methods. However, these approaches have yet to be tested for highly-dynamic movements such as a run. These new approaches formulate the task-space control problem as a quadratic program, linearizing the friction cone constraints. The active set solution approaches are able then to obtain increased computational speed when the active set of constraints does not change from one control step to the next. While this approach is effective for quasi-static locomotion, its utility in highly dynamic scenarios, where forces and torques are more rapidly changing, remains unstudied.

- Although the task-space controller has been shown to be applicable for uneven terrain scenarios, the application of simple models for uneven terrain remains an interesting future direction. Some theoretical work has studied the application of leg repositioning strategies for terrain-robust deadbeat control of the 3D-SLIP [151]. The integration of these results with a task-space control framework could provide an initial avenue of approach.
In this work, very simple setpoints (normally zero valued) were employed for centroidal angular momentum control. It remains an unsolved problem as to how more complex desired angular momentum setpoints or trajectories can be used to improve balance control. For instance, following a push that pitches the torso during a running stride, some small angular momentum may be desired during flight to reorient the torso. Principled methods to form desired angular momentum trajectories could also have application to produce motions with rapid turning, such as in the performance of a flip.

In the 3D-SLIP model, the ground reaction force (GRF) intersects the CoM at every instant of support, giving rise to constant centroidal angular momentum for the template model. Recently, biomechanics studies have shown that the GRF does not intersect the CoM during human locomotion. Instead GRFs often intersect at a different point, called the Virtual Pivot Point (VPP) [81] which is offset from the CoM. Templates described as Trunk SLIP (TSLIP) models [125] have been studied to capture this phenomena in 2D. Both extensions of this model to 3D and its application as a template for the humanoid could provide more principled setpoints for pitch angular momentum or torso posture, and could facilitate the replication of heel to toe center of pressure movement which is observed in human running.

In the current work, leg trajectories were designed heuristically with a series of cubic splines. Although these trajectories were general to produce the results in this dissertation, more principled optimization of these trajectories could provide benefits to the humanoid. For instance, leg trajectories could be optimized...
to require the least amount of torque and allow the natural dynamics of leg swing to partially carry the leg through its transfer. In addition, leg trajectory optimization could be structured to enforce early retraction of the leg, which would reduce impact losses. Previous work by Palmer and Orin [98] has shown that early retraction can be an important factor in the ability to traverse irregular terrain. Karssen et al. [59] have shown that that early swing-leg retraction improves gait stability and disturbance rejection without effect on the energetic efficiency of the gait.

- High-velocity leg swing contributes the most significant dynamics which are driven by hand-designed trajectories in this work. Still, the desired dynamics for other tasks do affect the resultant motion. Future work could study a general fusion of trajectory optimization approaches with the task-space control approaches applied here. By also optimizing desired torso trajectories and arm trajectories offline (with the task-space controller in the evaluation loop), it is likely that overall system actuation requirements could be reduced, while the continued application of simple-model-based CoM control could maintain the ability to provide dynamic balance online at real-time rates. Alternately, this fusion could be studied through including simple-model embeddings as constraints within a trajectory optimization algorithm. This approach could have potential to make whole-body trajectory optimization less sensitive to initial seeds. A nesting of this approach with trajectory optimization on successively more complex template models, prior to whole-body optimization, could further boost this benefit.
The current generation of a high-speed turn enforces a specific pattern on the evolution of the top-of-flight (ToF) velocity from step to step. This approach has been shown to be effective, and provides kinematically viable footstep locations when retargeted to the humanoid. Still, the outside leg during the turn has been shown to provide the majority of the required centripetal force. The use of torso reorientation to enable additional centripetal force production for the inside leg could be studied in order to more closely equalize the contributions of each leg to the turn.

The incorporation of a flip into a running long jump could be studied to determine its impact on jump span. During a running long jump, the main goal is to generate vertical impulse to increase the jump span. Excess vertical impulse in the second half of stance, however, would create an undesired moment on the system, and result in excessive forward torso pitch. In the performance of a running flip, however, this forward pitch is desired and could enable additional vertical impulse production as compared to the jump here, providing additional jump span.

This work has shown the robustness of the control approach to push disturbances. A more thorough analysis of the robustness characteristics of the approach, to factors such as control delays, sensing inaccuracies, and model discrepancies, would need to be carried out before application on physical hardware.

The running control approaches developed here have been limited to the application of deadbeat control of the 3D-SLIP model. Although deadbeat control
is desirable in the sense that it eliminates tracking error quickly, it does so at the expense of large control inputs. Preliminary results show that more sophisticated Model Predictive Control approaches for the 3D-SLIP model have the potential to improve the regulation capabilities of the 3D-SLIP controller (in terms of its basin of attraction), while employing smaller control inputs.

- Finally, the application of the 3D-SLIP model to the the range of motions studied here encourages it further use to develop a wider vocabulary of motion. For instance, this model could be extended to perform static-to-dynamic transition movements such as a fast start or fast stop.

8.3 Conclusions

In summary, this dissertation has developed a framework to optimize and control high-speed running and jumping movements in a humanoid model, largely through the new application of 3D-SLIP models within the humanoid domain. This template model only describes the most important characteristics of locomotion, namely the foot step locations and center of mass trajectories. Task-space control with centroidal angular momentum as a key task has been shown to enable the remainder of the humanoid motion to be managed online at real-time rates. The high-speed and disturbance-robust capabilities generated through this approach have previously not been demonstrated for a humanoid in simulation or hardware. These gait generation approaches have provided a departure from traditional ZMP-based methods that often employ the linear inverted pendulum (LIP) model.

Looking ahead, parallel advances in system sensing and high-power compliant actuation have the potential to ultimately realize physical humanoids with dynamic
capabilities. Combined with the control framework in this dissertation, future humanoids could have the capacity for dynamic balance; preventing falls by predicting and managing the long-term important dynamics of a given motion. The strides made in this dissertation will hopefully enable forward progress, particularly, at faster than quasi-static rates, on the journey to realize this vision.
Bibliography


Appendix A

Detailed Humanoid Model Description

This appendix describes the humanoid model used here in further detail. Figure A.1 provides the coordinate system definitions for the left half of the humanoid. These local coordinate systems are used in this Appendix to described the rigid body parameters of each link.

Each of the revolute joints shown, including the individual ankle joints, can be described with the Denavit-Hartenberg (DH) convention [136], which enables additional efficiency savings during forward simulation [83]. These joints are special in that for a DH joint $i$, frame $i$ is related to its predecessor frame $p(i)$ through a screw transformation about the $\hat{x}_{p(i)}$ axis, followed by a screw transformation about the $\hat{z}_i$ axis. In all cases here, the second screw transform is simply a rotational transform.

This property causes the associated spatial transformation $^iX_{p(i)}$ to take the form

\[
^iX_{p(i)} = \begin{bmatrix} R_z(q_i)^T & 0 \\ 0 & R_z(q_i)^T \end{bmatrix} \begin{bmatrix} R_x(\alpha_i)^T & 0 \\ R_x(\alpha_i)^T S(p(i)p_i)^T & R_x(\alpha_i)^T \end{bmatrix}
\]  \hspace{1cm} (A.1)

where $R_x(\alpha_i)$ is a fixed rotation of magnitude $\alpha_i$ about the $\hat{x}_{p(i)}$ axis

\[
R_x(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) \end{bmatrix}.
\]  \hspace{1cm} (A.2)
Figure A.1: Coordinate system definitions for the left half of the humanoid. The coordinate systems for the right half of the humanoid are parallel to those shown. That is, for a symmetric configuration of the right half of the humanoid, the \( \hat{z} \) axis of all arm joints points to the right of the humanoid, while the \( \hat{z} \) axis for all hip, knee, and initial ankle joints points to the left.
and \( R_z(q_i) \) is a joint angle \( q_i \) dependent rotation given as

\[
R_z(q_i) = \begin{bmatrix}
\cos(q_i) & -\sin(q_i) & 0 \\
\sin(q_i) & \cos(q_i) & 0 \\
0 & 0 & 1
\end{bmatrix}.
\] (A.3)

Use of the DH convention for these joints also allows \( p^{(i)}_i \) to take the simplified form

\[
p^{(i)}_i = \begin{bmatrix} a_i \\ 0 \\ 0 \end{bmatrix}
\] (A.4)

for a fixed scalar \( a_i \). With these conventions, and those outlined in Chapter 2, the following sections detail the kinematic and inertia parameters for each of the major links in the system. Parameters are provided for the bodies in the left half of the humanoid only, as they are the same for bodies in the right half.

**Torso Body**

The torso body, Body 1, is modeled as a single rigid body that contains the lumped mass and inertia of the waist, body, neck, and head. Table A.1 provides the dynamic parameters for this body which are required to form its spatial inertia tensor as described in Eq. 2.8. All quantities are given with respect to the origin of frame 1, \((\hat{x}_1, \hat{y}_1, \hat{z}_1)\) shown in Figure A.1, which is centered between the hips.

**Thigh Bodies**

The thigh bodies, Bodies 2 and 6, rotate relative to the torso through a ball and socket joint. Table A.2 provides the dynamic parameters for the left thigh, and gives the vector \( ^1p_2 \in \mathbb{R}^3 \) which locates the origin of frame 2 with respect to frame 1.

**Shank Bodies**

The shank bodies, Bodies 3 and 7, rotate relative to each thigh through a single degree of freedom revolute joint at the knee. Table A.3 provides the dynamic
<table>
<thead>
<tr>
<th>Mass</th>
<th>$m_1 = 41.948$ kg</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoM Vector</td>
<td>$c_1 = \begin{bmatrix} 0 \ 0 \ 0.4056 \end{bmatrix}$ m</td>
</tr>
<tr>
<td>Inertia Matrix</td>
<td>$\bar{I}_1 = \begin{bmatrix} 7.4050 &amp; 0 &amp; 0 \ 0 &amp; 8.6238 &amp; 0 \ 0 &amp; 0 &amp; 1.4210 \end{bmatrix}$ kg·m²</td>
</tr>
</tbody>
</table>

Table A.1: Dynamic parameters for the torso body.

<table>
<thead>
<tr>
<th>Mass</th>
<th>$m_2 = 7.258$ kg</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoM Vector</td>
<td>$c_2 = \begin{bmatrix} 0.209 \ 0 \ 0 \end{bmatrix}$ m</td>
</tr>
<tr>
<td>Inertia Matrix</td>
<td>$\bar{I}_2 = \begin{bmatrix} 0.0229 &amp; 0 &amp; 0 \ 0 &amp; 0.4681 &amp; 0 \ 0 &amp; 0 &amp; 0.4711 \end{bmatrix}$ kg·m²</td>
</tr>
<tr>
<td>Frame Origin</td>
<td>$^1p_2 = \begin{bmatrix} 0 \ 0.127 \ 0.053 \end{bmatrix}$ m</td>
</tr>
</tbody>
</table>

Table A.2: Dynamic and kinematic parameters for the left thigh body. The parameters for the right thigh body are the same except for $^1p_6 = [0, -0.127, 0.053]^T$ m.
Mass:

\[ m_3 = 3.375 \text{ kg} \]

CoM Vector:

\[ c_3 = \begin{bmatrix} 0.189 \\ 0 \\ 0 \end{bmatrix} \text{ m} \]

Inertia Matrix:

\[ I_3 = \begin{bmatrix} 0.0063 & 0 & 0 \\ 0 & 0.1778 & 0 \\ 0 & 0 & 0.1778 \end{bmatrix} \text{ kg} \cdot \text{m}^2 \]

DH Parameters:

\[ \alpha_3 = 0 \]
\[ a_3 = 0.483 \text{ m} \]

Table A.3: Dynamic and kinematic parameters for the left shank body.

Parameters for the left shank and describes its kinematic parameters using the DH convention.

**Ankle and Foot Bodies**

Each foot, Body 5 or 9, is attached to its shank through a series of two revolute joints, which gives the equivalent kinematics of a universal joint. This modeling is implemented through the insertion of a zero-mass intermediate ankle body, Body 4 or 8, between the shank and the foot. Table A.4 provides the dynamic parameters for the left foot link, and describes the kinematic parameters of the foot and its intermediate ankle body. Contacts are modeled at the corners of the feet, with contact positions relative to each foot frame at

\[
\begin{bmatrix} 0.0549 \\ 0.0805 \\ -0.0695 \end{bmatrix} \text{ m}, \quad \begin{bmatrix} 0.0549 \\ -0.0805 \\ -0.0695 \end{bmatrix} \text{ m}, \quad \begin{bmatrix} 0.0549 \\ 0.0805 \end{bmatrix} \text{ m}, \text{ and } \begin{bmatrix} 0.0549 \\ -0.0805 \end{bmatrix} \text{ m}.
\]

(A.5)
Table A.4: Dynamic and kinematic parameters for the left foot body and kinematic parameters for its intermediate ankle body.

**Upper Arm Bodies**

Each upper arm, Body 10 or 12, is attached to the torso through a ball and socket joint shoulder joint. Table A.5 provides the dynamic and kinematic parameters for the left upper arm.

**Forearm Bodies**

Each forearm, Body 11 or 13, is attached to the upper arm through a revolute joint at the elbow. The forearm body includes the mass an inertia of the forearm and hand. Table A.6 provides the dynamic and kinematic parameters for the left forearm.
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mass</strong></td>
<td>$m_{10} = 2.0321 \text{ kg}$</td>
</tr>
<tr>
<td><strong>CoM Vector</strong></td>
<td>$\mathbf{c}_{10} = \begin{bmatrix} 0.1180 \ 0 \ 0 \end{bmatrix} \text{ m}$</td>
</tr>
<tr>
<td><strong>Inertia Matrix</strong></td>
<td>$\mathbf{\bar{I}}_{10} = \begin{bmatrix} 0.0036 &amp; 0 &amp; 0 \ 0 &amp; 0.0430 &amp; 0 \ 0 &amp; 0 &amp; 0.0420 \end{bmatrix} \text{ kg\cdot m}^2$</td>
</tr>
<tr>
<td><strong>Frame Origin</strong></td>
<td>$^{1}\mathbf{p}_{10} = \begin{bmatrix} 0 \ 0.1875 \ 0.5870 \end{bmatrix} \text{ m}$</td>
</tr>
</tbody>
</table>

Table A.5: Dynamic and kinematic parameters for the left upper arm body. The parameters for the right upper arm body are the same except for $^{1}\mathbf{p}_{12} = [0, -0.1875, 0.5870]^T \text{ m}$.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mass</strong></td>
<td>$m_{11} = 1.5966 \text{ kg}$</td>
</tr>
<tr>
<td><strong>CoM Vector</strong></td>
<td>$\mathbf{c}_{11} = \begin{bmatrix} 0.2993 \ 0 \ 0 \end{bmatrix} \text{ m}$</td>
</tr>
<tr>
<td><strong>Inertia Matrix</strong></td>
<td>$\mathbf{\bar{I}}_{11} = \begin{bmatrix} 0.0018 &amp; 0 &amp; 0 \ 0 &amp; 0.1697 &amp; 0 \ 0 &amp; 0 &amp; 0.1695 \end{bmatrix} \text{ kg\cdot m}^2$</td>
</tr>
<tr>
<td><strong>Frame Origin</strong></td>
<td>$^{10}\mathbf{p}_{11} = \begin{bmatrix} 0.2707 \ 0 \ 0 \end{bmatrix} \text{ m}$</td>
</tr>
</tbody>
</table>

Table A.6: Dynamic and kinematic parameters for the left forearm body.
Appendix B

Computational Cost Details for the Operational-Space Inertia Algorithms

This appendix details the computational costs computations for the EFPA and RJKA presented in Chapter 3. A detailed version of the RJK algorithm, in the notation of this dissertation, is given with its backward pass in Table B.1 and forward pass in Table B.2. As mentioned in Chapter 3, the inward recursion of the RJKA is much simpler in comparison to the EFPA. However, the additional computation on the inward pass of the EFPA greatly simplifies its outward pass. Table B.3 provides a notational correspondence between the notation here and the original notation of the RJK algorithm [31, 112].

The RJKA arrives at the block of $\Lambda^{-1}$ recursively by considering intermediate quantities $\Lambda_{ij}^{-1}$ which relate forces on Body $j$ to accelerations at Body $i$. In their algorithm, neither of these intermediate bodies are end-effector bodies. This is in contrast to the EFPA that uses intermediate quantities $\Lambda_{ik}^{-1}$ which relate forces at end-effector $k$ to accelerations at Body $i$. The RJKA requires only a subset out of all possible $\Lambda_{ij}^{-1}$ quantities, as encoded in a boolean array $\textit{need}$. An algorithm to compute the boolean array $\textit{need}$ for the RJKA is given in [31].
Initialize: $I^A_i = I_i$

for $i = N$ to $1$ do
  $D_i = (\Phi_i^T I^A_i \Phi_i)^{-1}$
  $K_i = \Phi_i D_i \Phi_i^T$
  $L_i = 1_{6 \times 6} - K_i I^A_i$
  if $p(i) \neq 0$ then
    $I^A_{p(i)} = I^A_{p(i)} + iX^T_{p(i)} I^A_i L_i iX_{p(i)}$
  end
end for

Table B.1: Inward recursion of the RJK Algorithm

The computational cost of each of the algorithm operations is given in Table B.4. These cost assignments largely follow those presented in [31, App. B]. Each of the spatial operations in the algorithms employ various efficiency tricks that are enabled by each recursive step taking place in local coordinates. Most of these tricks relate to exploiting structure in the spatial transformation matrices $iX_{p(i)}$. At any non-terminal link, the Denavit-Hartenberg convention can be used to describe the connectivity to a single child. For such children, this enables the coordinate transformations $iX_{p(i)}$ to represent 2 screw transformations, one about the $x$-axis, followed by one about the once-transformed $z$-axis. Further details on these cost savings are given by Featherstone [30] and McMillan and Orin [83]. Another optimization, as noted by McMillan et al. [84], is that the floating-base coordinate system can be located such that it is related by a pure rotation about the $z$-axis to a privileged child. This results in dramatic cost savings for this single privileged link in the system. In all other cases, spatial transformations require 3 screw transformations.
for $i = 1$ to $N$ do
  if need$(i, i)$ then
    $\Lambda_{ii}^{-1} = K_i$
    if $p(i) > 0$ then
      $\Lambda_{ii}^{-1} = \Lambda_{ii}^{-1} + L_i \cdot X_{p(i)} \cdot \Lambda_{p(i)p(i)}^{-1} \cdot X_{p(i)}^T \cdot L_i^T$
    end
  end
end for $i$

for $i = 1$ to $N - 1$ do
  for $j = i + 1$ to $N$ do
    if need$(i, j)$ then
      if $\text{ancest}(i, j) = i$ then
        $\Lambda_{ij}^{-1} = \Lambda_{ip(j)}^{-1} \cdot jX_{p(j)}^T \cdot L_{j}^T$
      elseif $\text{ancest}(i, j) > 0$ then
        $\Lambda_{ij}^{-1} = L_i \cdot X_{p(i)} \cdot \Lambda_{p(i)p(j)}^{-1} \cdot jX_{p(j)}^T \cdot L_{j}^T$
      else
        $\Lambda_{ij}^{-1} = 0$
      end
    end
  end for $j$
end for $i$

for $k_1 = N + 1$ to $N + m$ do
  for $k_2 = k_1$ to $N + m$ do
    $\Lambda_{k_1k_2}^{-1} = k_1 X_{p(k_1)} \cdot \Lambda_{p(k_1)p(k_2)}^{-1} \cdot k_2 X_{p(k_2)}^T$
  end for $k_2$
end for $k_1$

$\Lambda^{-1} = \begin{bmatrix}
  \Lambda_{N+1,N+1}^{-1} & \cdots & \Lambda_{N+1,N+m}^{-1} \\
  \vdots & \ddots & \vdots \\
  \Lambda_{N+1,N+m}^{-1} & \cdots & \Lambda_{N+m,N+m}^{-1}
\end{bmatrix}$

Table B.2: Outward recursion of the RJK Algorithm
Table B.3: Notational correspondence between [31, Table 4] and the RJKA here.

<table>
<thead>
<tr>
<th>RJKA [31, Table 4]</th>
<th>RJKA here</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_i^{-1}$</td>
<td>$D_i$</td>
</tr>
<tr>
<td>$\tau_i$</td>
<td>$L_i^T$</td>
</tr>
<tr>
<td>$\phi_i$</td>
<td>$iX_{p(i)}^T$</td>
</tr>
<tr>
<td>$P_i$</td>
<td>$I_i^A$</td>
</tr>
<tr>
<td>$\Omega_{ij}$</td>
<td>$\Lambda_{ij}^{-1}$</td>
</tr>
</tbody>
</table>

The most costly operations listed in Table B.4 are those which relate to the application of extended force and acceleration propagators. That is, multiplications:

$$\Lambda_{k_1 k_2}^{-1} = k_1 \mathcal{X}_i \Lambda_{i k_2}^{-1} \quad \text{and} \quad \Lambda_{1 k}^{-1} = K_1 \mathcal{X}_1^T$$

require dense matrix multiplications with cost $216m + 180a$ in general. (This cost is reduced by $36m+36a$ for the second operation if $k$ is supported by the privileged child, since $\mathcal{X}_1^T$ has a zero row in this case.) Despite this high cost, application of $\mathcal{X}_i$ is still cheaper than repeated application of local transforms, even for relatively short chains. This is the fundamental principle that enables the computational benefit of the EFPA over the recursive approach of RJK.
<table>
<thead>
<tr>
<th>Calculation</th>
<th>Algorithm</th>
<th>Case</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_i$</td>
<td>Both</td>
<td>Floating-Base</td>
<td>$36, d + 105, m + 90, a$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Revolute</td>
<td>1, d</td>
</tr>
<tr>
<td>$L_i$</td>
<td>Both</td>
<td>Floating-Base</td>
<td>0, m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Revolute</td>
<td>5, m</td>
</tr>
<tr>
<td>$I^A_i L_i$</td>
<td>Both</td>
<td></td>
<td>15, m + 15, a</td>
</tr>
<tr>
<td>$iX^T_{p(i)} (\ldots) iX_{p(i)}$</td>
<td>Both</td>
<td>z rotation</td>
<td>22, m + 25, a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 screws</td>
<td>60, m + 62, a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 screws</td>
<td>98, m + 102, a</td>
</tr>
<tr>
<td>$I^A_{p(i)} + (\ldots)$</td>
<td>Both</td>
<td>z rotation</td>
<td>15, a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 or 3 screws</td>
<td>21, a</td>
</tr>
<tr>
<td>$iX_{p(i)} A^{-1}_{p(i)k}$</td>
<td>Both</td>
<td>z rotation</td>
<td>48, m + 24, a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 screws</td>
<td>120, m + 72, a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 screws</td>
<td>144, m + 108, a</td>
</tr>
<tr>
<td>$L_i(\ldots)$</td>
<td>Both</td>
<td></td>
<td>30, m + 24, a</td>
</tr>
<tr>
<td>$iX_{p(i)} A^{-1}<em>{p(i)p(i)} iX^T</em>{p(i)}$</td>
<td>RJKA</td>
<td>z rotation</td>
<td>63, m + 54, a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 screws</td>
<td>76, m + 80, a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 screws</td>
<td>114, m + 120, a</td>
</tr>
<tr>
<td>$L_i (\ldots) L^T_i$</td>
<td>RJKA</td>
<td></td>
<td>35, m + 28, a</td>
</tr>
<tr>
<td>$A^{-1}_{ii} + (\ldots)$</td>
<td>RJKA</td>
<td></td>
<td>1, a</td>
</tr>
<tr>
<td>$K_i^k X_i^T$</td>
<td>EFPA</td>
<td>Floating-Base</td>
<td>216, m + 180, a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Revolute</td>
<td>6, m</td>
</tr>
<tr>
<td>$k_i X_i A^{-1}_{i k_2}$</td>
<td>EFPA</td>
<td></td>
<td>216, m + 180, a</td>
</tr>
</tbody>
</table>

Table B.4: Computational costs for spatial operations within the EFPA and RJKA. The notation (\ldots) indicates that the operation is performed on the result of the line above it.