The Dissertation Committee for Donghyun Kim
certifies that this is the approved version of the following dissertation:

Sensor-Based Robust Whole-Body Control of Highly Dynamic Legged Humanoid Robots

Committee:

Luis Sentis, Supervisor

Benito R. Fernandez

Peter Stone

James Sulzer

Dongmei Chen

Sensor-Based Robust Whole-Body Control of Highly Dynamic Legged Humanoid Robots

by

Donghyun Kim

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2018

Acknowledgments

I wish to thank the multitudes of people who helped me. My most profound gratitude should go to Dr. Sentis who has been supporting my entire Ph.D. and providing the most vital trust for my research. I am sure that I could not achieve a single piece of progress without his incredible supervising and supports. I also need to thank for all committees about their invaluable comments to make my dissertation mature. Dr. Fernandez has always been generous to my random questions, and share his wisdom. Dr. Stone gave me a chance to test my algorithm in an NAO robot, which might be the first moment to implement our WBC on a full humanoid robot. The study caused me to contemplate the transferability of whole-body control. I need to appreciate Dr. Chen and Dr. Sulzer about the generous comments and essential inputs for my research and help to clarify the direction of this dissertation.

I also need to mention all our lab colleagues who are the smartest robotics engineers in the world. The cooperation with you all is one of the happiest memories of my life. Especially, Kwansuk is the most prominent utility man answered all issues I faced during experimental studies. It is lucky to meet Junhyeok who has stayed all nights with me and has gone through every step to build experimental setup. I would like to mention Steven who has always been supportive of my studies and willing to help me. I surely remember all great cooperation with Orion, Nick, and Uday. I should say, Gray and Ye, co-authors of our first journal paper during my Ph.D. Travis, I cannot describe how lucky we are to have you in the lab.

My Ph.D. life cannot be healthy without my best Korean friends, Kyungwoo, Youngbum, and Sangok. My ex-roommate, Taewoo, did all required works to settle my life down in the U.S. instead of me. All members of our basketball club are crucial people in my Ph.D. life.

I must appreciate and be willing to give all achievements for my family. My father, mother, sister, and brother are invaluable meaning in my life. Parents in law have been providing me unconditional support and trust, which I have never received from any other person. The last but the most important person is my wife, Euddeum, who is the key to all my achievements and future accomplishments. It is the best luck in my life to have you as my life partner.

Thank you all.

Sensor-Based Robust Whole-Body Control of Highly Dynamic Legged Humanoid Robots

Publication No. _____

Donghyun Kim, Ph.D.
The University of Texas at Austin, 2018

Supervisor: Luis Sentis

Industrial robots significantly improve the productivity of manufacturing operations performing various tasks rapidly, accurately, and repeatedly. It would be hard to imagine factories without robotic arms. At the same time, it is difficult to imagine human-centered robots maintaining infrastructure and providing care as they are not yet versatile enough. One important obstacle to the adoption of human-centered robots is their limited mobility. Legged humanoid robots represent an embodiment of a highly dexterous system which could provide human-like capabilities to boost automated services in human environments. Therefore this thesis is dedicated to investigate the sensor-based control of legged humanoids robots such that they can achieve versatile and high task performance.

To tackle agility and robustness in legged humanoid robots, I have studied the dynamic whole-body motion control of these kind of robots, with special focus on dynamic locomotion in coordination with whole-body task capabilities. One of the unique aspects of this study is the enhancement of locomotion capabilities without compromising the robot's dexterity. Currently, existing locomotion techniques for legged systems are highly specialized and not adaptable to generic robotic structures with manipulation requirements. Here, we explore the robot's legged mobility without compromising its dexterity by utilizing a general-purpose whole-body controller (WBC), i.e. a control algorithm which can find a dynamically-consistent mapping from operational space tasks to joint torques. The use of a WBC is appealing due to its ability to coordinate multiple tasks for highly redundant robotic systems. As such, WBCs have been deployed recently for controlling humanoid robots. However, the use of WBCs for achieving highly

dynamic sensor-based motions has been lacking, and our work addresses the technical problems of such and endeavor.

Our research primarily focuses on employing WBCs for dynamic motion control of legged robots. The dynamic control of real robots requires both algorithmic developments and comprehensive system analyses for real-time deployment, which covers a broad spectrum of components from motor-level control to high-level planners. Therefore, my studies include the algorithmic enhancement of WBCs, the development of locomotion planners, the analysis of real-time controllers, and the integration of state-estimators. The algorithmic theory and methods are verified in both simulation and various real systems.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

During DARPA's Robotics Challenge held in 2013 and 2015, various humanoid robots were required to execute many practical tasks [34]. Most of these robots implemented a certain class of controllers called whole-body controllers (WBC), which are easily reconfigurable to accomplish multiple control objectives under environmental constraints. A perceived limitation of these robots was that their motions were much slower than their human counterparts. To remedy this issue, future WBC methods must significantly improve the speed and accuracy of the executed motions without losing their task processing capability. We have investigated the dynamic motion control of multi-limb systems using WBC to address the fundamental difficulties in high speed and accurate task execution. Especially, our research focuses on biped locomotion, which is the most challenging problem in humanoid control. In this chapter, we describe our motivation, the state-of-the-art of humanoid control, and our approaches for the issues in dynamic control of biped robots.

1.1 Motivation and Background

Biped robots are an alternative technology over wheeled systems due to their maneuverability in various terrains, their small footprint for operations in tight spaces, and their omnidirectional movements. To utilize this promising topology, we need to build a robust control architecture to make biped robots be able to perform human-level tasks in dynamic environments. This dissertation will explore control issues in dynamic biped locomotion and methods for the real-time controls, aiming to extend the frontiers of robotics research and related fields.

1.1.1 Whole Body Control

WBC [86] is a family of multi and prioritized task-space trajectory controllers for humanoid robots that rely on floating-base dynamic and computed torque commands as inputs to the plant. It yields asymptotically stable control policies for multiple tasks with simultaneous control of operational forces when needed. Priorities address resource allocation when two or more

task trajectories cannot physically be tracked by the robotic system. It naturally integrates equality constraints such as biarticular transmission constraints [89]. Other groups have explored richer versions of WBC with inequality constraints such as joint limit avoidance [21, 56, 57], collision avoidance [39], and singularity avoidance [60]. Several groups used evolved and more practical versions of WBC such as controllers used in the DARPA Robotics Challenge of 2013 and 2014. For instance, [36, 55] incorporate reaction forces as inequality constraints based on solving a quadratic programming optimization problem with desired center of mass trajectories. Treatment of reaction forces as inequality constraints in the WBC communities dates back to the work by [96]. And it showcases one of the weaknesses of our group's formulation of WBC. In early versions [87] we treated reaction forces as equality constraints. Such treatment corresponds to bilateral contact constraints, i.e. assuming that the floor contacts are actually rigid anchors. This is obviously an inaccurate model. One of our new WBC formulations introduced in Chapter. 3 uses a realistic unilateral contact model for WBC while maintaining one of its main strengths, efficient prioritized control.

Bipedal and quadrupedal walking capabilities have been devised using WBC. [32] demonstrated locomotion of a quadrupedal robot by utilizing hierarchical tasks based on least-square problems. The integration of the versatile capture point (CP) as an operational space of WBC was proposed and controlled either as a constraint or a task for bipedal humanoid robots [80]. The robot's Center of Gravity (CoG) has been used as a task controller for a while, such as in [59]. Walking pattern generators have been incorporated into WBC in multiple instances such as in [13]. During the DARPA robotics challenge, several top participants incorporated WBC's into their strategy for achieving mobile dexterous capabilities. For instance, high-level trajectory optimization and low-level optimization with inverse dynamics were integrated into the framework by [20].

Many experimental approaches for WBC rely on impedance-based techniques or inverse dynamics and optimization algorithms to solve for constraints and contacts. One of them is the pioneering work by [33] which formulates task space impedance controllers to fulfill contact and task constraints; the concept of impedance control was first presented in [30]. [95] represents the first implementation that we know of to achieve full dynamic model based task control with contact constraints on a humanoid robot. In [27] the implementation of hierarchical inverse dynamic algorithms using a quadratic program is presented and demonstrated on a Sarcos biped robot. Experiments include balancing while withstanding external forces, balancing on a moving platform, and standing on a single foot. In [5] a torque-based WBC is presented for controlling the Atlas and Valkyrie humanoid robots. A quadratic program solver is used to minimize

the error with respect to the desired momentum rate, contact forces, and task accelerations. In [84] WBC with inequality constraints via inverse dynamics and a quadratic program solver is proposed for the humanoid robot HRP-2. The algorithm is used offline to generate trajectories that are then tracked by a real-time controller. In [24] a torque-based WBC focused on optimizing multi-contact wrenches derived from desired center of mass movements is presented. These studies aim at controlling humanoid robots with actuated ankles, and thus do not consider the underactuation nature and fast locomotion dynamics of point-foot biped robots.

Many of the previous WBCs formulate structures for controlling contact forces. There are many styles for regulating those forces. In [33], desired center of mass accelerations and task wrenches are projected and summed up yielding a total center of mass wrench. When there are multiple contacts, an optimization algorithm is applied to optimally distribute the contact wrenches before projecting them to the center of mass. A center of mass Jacobian transpose projection is used to achieve the desired global wrench. One of the problems with this method is that the forward kinematics model used for controlling the center of mass does not account for contact constraints which could result in less accurate motion tracking and force regulation. Nonetheless, the method allows the Sarcos humanoid robot to balance fairly well but with visibly slow center of mass motions.

One great controller is the full body controller by [96] focusing on two stages consisting on first solving for the contact forces then followed by solving the full-body constrained inverse dynamics problem. To deal with multiple contacts, a quadratic program is posed based on a simplified relation between center of mass and task wrenches to solved for the contact forces that comply with friction constraints. Another successful WBC is the momentum-based controller by [6]. Variations of this controller were used to obtain the second place for the DARPA Robotics Challenge. The controller has a similar two-stage nature than the previous full body controller. But it has multiple advantages. First, for the contact solver, it uses the centroidal momentum matrix [26,64] which correctly describes the relationship between external wrenches, i.e. contact wrenches and gravitation forces, and joint accelerations. Then it solves a quadratic problem that includes motion task constraints. In turn, the output reaction forces are dynamically correct with respect to center of mass behavior and motion tasks. Second, using the centroidal momentum matrix it solves for joint accelerations, which is a more generic output than reaction forces alone. In its second stage, it solves for the constrained inverse dynamics based on the output joint accelerations and contact wrenches.

In our WBOSC in Section. 2.1, internal forces are the control input for regulating contact forces. Because internal forces are fully controllable, WBOSC uniquely formulates a sensor-based

feedback controller to precisely regulate internal forces. In comparison, previous controllers rely on feedforward control of the internal forces which require high-bandwidth joint torque control. However, the high-bandwidth joint torque control generally impedes achieving high-bandwidth task control as discussed in Section 5.1.1.

It is necessary for us to convert desired contact forces to internal forces. Fortunately, this is realizable by using the simple projection operators from contact forces to internal forces described in Equation (2.16). As for determining the desired contact forces that comply with frictional constraints, we described in [88] a search based method to do so. In this dissertation, we describe a simple multi-contact model to solve for the contact forces that comply with frictional constraints. It would also be possible to use the contact solvers mentioned in [6, 96] as a first stage, then proceed to solve the inverse task dynamics problem using WBOSC as the second stage of the controller. The takeaway message is that we can use any contact solver of our choice, convert contact forces to internal forces using simple projections as described in Equation (2.16), then proceed to solve for the whole-body torques using WBOSC. Note that controlling task accelerations plus internal forces is equivalent to controlling reaction forces. For instance, the momentum-based controller previously reviewed solves for the joint accelerations and reaction forces that comply with task accelerations and friction constraints. In our case, we directly solve for the task accelerations and compute the internal forces that comply with friction constraints. The two cases are functionally equivalent. However, WBOSC allows for feedback control of internal forces which other controllers do not (Section. 5.1.2).

1.1.2 Control of Robots with Point Feet

Point-foot biped robots similar to ours have often been studied [9, 14, 25, 74, 106, 108] due to their mechanical simplicity and fast locomotion capabilities. Only a few have achieved dynamical balancing without a constraint mechanism, the three most notable ones being the hydraulically actuated hopper from [78] and the bipeds from [9] and [82]. These last two biped robots are based on the same mechanical architecture but differ in the type of controller.

One of the most successful approaches to point-foot locomotion comes from [93], which stabilizes the Hybrid Zero Dynamics of a high dimensional multi-body model of the robot. However, the algorithm is designed for robots supported by a planarizing mechanism. Recently, the same group [9] has combined this full-body locomotion algorithm with the SIMBICON method [109] to achieve unsupported point-foot locomotion. Like ours, the robot accomplishes numerous steps before falling.

In the works [81, 82], an unsupported point-foot robot accomplishes continuous walking by using only simple rule-based algorithms to stabilize the walk. Our locomotion algorithm is different in that it searches adequate foot positions to stabilize the robot around a balance point. The balance point can stay in place or track a desired trajectory for locomotion. In terms of walking, we cannot think of advantages or disadvantages compared to that line of work. The higher performance of their experimental locomotion results is due to a number of factors which we believe are not related to the choice of locomotion planner. First, the mechanical structure of their robot is more rigid than ours providing significantly higher foot positioning accuracy. Second, they use a high performance IMU which contributes to precision and tracking stability. Third, their robot uses small passive feet that prevent it from drifting on the yaw direction. In contrast, our robot does not have passive feet. One key difference of WBOSC is that it allows biped robots to use internal forces for balancing on highly irregular terrains, such as our example on balancing Hume on a steep disjointed terrain.

Another successful locomotion approach also based on hybrid zero dynamics, utilizes human-inspired trajectories to generate stable periodic locomotion and even handle some roughness on the terrain [25,110]. However, the locomotion experiments use a mechanical planarizing stage to constraint the robot to a plane. Frameworks such as the Capture Point [75] and the Divergent Component of Motion [17] based on the linear point mass pendulum model are very practical and widely used for controlling humanoid robots with actuated feet. However, they have not been used to control point-foot robots to date.

1.1.3 Series Elastic Actuator Control

Series Elastic Actuators (SEAs) are designed with an elastic element and provide two important benefits over their rigid, or directly connected, counterparts: they offer improved force control accuracy [71, 103], and a lower output inertia. This high fidelity torque tracking is critical to contact scenarios where the internal multi-contact forces must be maintained within a certain range, as is the case when a robot balances on steep terrain. Some robots, for example the robots of [93] and [22], use SEAs only for energy storage, and do not take advantage of their force control capabilities. Hume, on the other hand uses SEAs primarily for their force control and inertial benefits to handle collisions. Yet the inclusion of SEAs comes at a cost to performance. Controllers that have been designed to approximate the dynamics of a perfect position source, a perfect torque source [67, 71, 103], or a second order mass-spring-damper system all face the same problem in their final closed loop system: the dynamics of the reference plant are impossible to achieve at the highest frequencies.

The use of SEAs for biped locomotion was pioneered in [72], which suggests a straightforward PD torque control strategy with some model-based feedforward terms. However, the work does not compensate for static errors. The same torque controller was used in the SEA actuated robot [75] to study push recovery. However, such an application does not require accurate force tracking so much as position control, and thus it is not possible to asses the torque performance of these actuators. Trajectory following accuracy did appear to be limited by the SEA compliance; In the author's own words they state: "Due to the use of SEAs with very compliant springs, we have had difficulty to quickly and accurately swing the leg," A sentiment echoed by our own observations. Recent studies describe potential solutions for this type of problem with SEA actuators [62, 92, 103, 112], with passivity based controllers emerging as a solid, if conservative, approach. [103] adds an inner motor velocity loop and incorporates integral torque action to the controller. More recently, [62] compared the stability and performance of various active impedance control approaches based on cascaded SEA controllers.

1.1.4 Reinforcement Learning based Locomotion Planner

One of the main challenges for learning robust dynamic locomotion policies is handling the high number of continuous variables describing the motion and force interactions of full humanoid robots. To deal with the dimensionality problem, we review previous work that has greatly inspired us. [61], solves a periodic locomotion generation problem via RL on a planar biped robot. We advance upon this work by solving the learning problem for 3D robots, avoiding reliance on human walking trajectories, and generating policies for non-periodic gaits. Other important works employ learning as an optimization problem over known locomotion trajectories. In [99], a periodic locomotion problems is solved by optimizations of a known stable central patter generated (CPG) walking trajectories using RL. However, no focus is given to dealing with large external disturbances. In addition, our focus is on the generation of trajectories from scratch without prior stable locomotion patterns.

In [58], robust walking trajectories to external pushes are achieved based on capture point trajectory optimization via gradient based learning updates. In this work, the capture point method is used as an analytic controller to initiate the learning process with information about foot placement, step-timing, and ZMP controls. Although the authors also show learning of push recovery strategies without previous capture point generated trajectories, our focus is stronger on autonomously learning the locomotion process without reliance on already stable walking gaits. As such, we belief our algorithm is able to learn from scratch recovery strategies in a more generic sense, for instance to recover from pushes in any direction while walking.

Like ours, autonomous learning of periodic gaits has been explored before in passive dynamic walkers [102]. Once more, our focus is on gait generators that can produce non-periodic gaits and tolerate large push disturbances in all directions of motion.

The dynamic locomotion community has previously used online optimization methods instead of RL, such as model predictive control (MPC). The main problem of these approaches is the high computational cost. To mitigate this problem, researchers have made significant efforts to develop efficient computational processes. [18] used the gradient of a cost function to solve the MPC problem efficiently. [42] linearized the planning problem by optimizing over one step ahead of time. Our approach relying on learned neural networks replaces the need for complex online computations, enabling the generation of hundreds of steps in an instant compared to the stepping time scales. [107] proposed a controller for a 12 DoF biped system by using dynamic programming and a lookup table that was obtained offline based on simple models. The multiple policies achieved from each simple model were combined to control the target system. In contrast, our work relies on the generic inverted pendulum locomotion model, and a versatile full-humanoid body controller.

1.2 Chapter Summary

The dissertation largely consists of four components: algorithmic aspects of WBC, locomotion planner, real-time feedback controllers, and state estimators. Each chapter includes simulation and experimental results to validate the proposed algorithms and methods. I made individual sections for biped balancing experiments to emphasize the importance of the experimental achievements and to ensure that details of the implementations are well illustrated. The dissertation is organized as follows. Chapter 2 explains the algorithmic enhancements we made in the WBC formulation. Chapter 3 describes our new WBC formulation incorporating inequality constraints, task hierarchy, and slip. In Chapter 4, we introduce our two new locomotion planners using a phase space planning method. Chapter 5 addresses issues raised in real-time feedback control and explains our methods to tackle the difficulties. Chapter 6 describes the formulations of our estimators and empirical verification of them. Chapter 7 presents experimental results of point-foot biped stabilization and details of the tests.

Chapter 2

Whole Body Control

"Whole-Body Control" is a torque-level, prioritized, projection-based, multi-task, operational space, feedback controller for floating based robots originally outlined in [85] and extended to include internal force control in [88]. It uses null-space projections to accomplish task prioritization and computes contact and constraint consistent reduced Jacobian matrices to project multiple task impedance controllers into joint torques. WBC also controls internal forces using the internal force matrix, which resides in the orthogonal space to joint accelerations and by extension to operational space motion tasks.

Although WBC has many benefits for humanoid motion control, the previous formulation shows limitations to manage contact switching and inequality constraints. In this chapter, we illustrate our contact transition method and our WBC formulation addressing inequality constraints. We also present two new task formulations, capture point (CP) and centroidal momentum (CM) tasks. Additionally, the analytical formulations of a time derivative of point Jacobian and $\dot{\boldsymbol{J}}\dot{\mathbf{q}}$ of centroidal angular momentum (CAM) are explained in Section 2.2.5. In the last section, we introduce our new WBC formulation, dubbed whole-body locomotion control.

This chapter contains material from the following publications:
[46] Donghyun Kim, Steven Jens Jorgensen, Peter Stone, and Luis Sentis. Dynamic behaviors on the NAO robot with closed-loop whole body operational space control. In 16th International Conference on Humanoid Robots (Humanoids), pages 11211128. IEEE, 2016.
[47] Donghyun Kim, Jaemin Lee, and Luis Sentis. Robust Dynamic Locomotion via Rein- forcement Learning and Novel Whole Body Controller. arXiv.org, 2017.
[49] Donghyun Kim, Ye Zhao, Gray Thomas, Benito R Fernandez, and Luis Sentis. Stabilizing Series-Elastic Point-Foot Bipeds Using Whole-Body Operational Space Control. IEEE Transactions on Robotics, 32(6):13621379, 2016.
My work covers the algorithm development, programming, experiment, and the major portion of paper writing.

2.1 Review of Whole-Body Operational Space Control

Whole-body operational space control (WBOSC) was first laid out in [85] with a further exploration of internal forces that was published later, in [88]. Interested readers should refer to these sources for a description of the theoretical background complete with proofs for the concepts below, and I provide a cursory overview of WBOSC in this dissertation.

Modeling robots entails representing not only the state of each joint $q_1, \ldots, q_{n_{\text{joints}}}$, but also the states of the robot as an object in space. We parameterize the robot's base as a 6-dimensional floating joint (three prismatic and one ball joints) between the world and the robot's hip coordinate system with state vector $\dot{\mathbf{q}}_b \in \mathbb{R}^6$. We use quaternion to present orientation of the floating base and our convention is explained in Appendix. A. Combining the robot and base states into a single vector, we arrive at $\dot{\mathbf{q}} \in \mathbb{R}^6 \oplus \mathbb{R}^{n_{\text{joints}}} = \mathbb{R}^{n_{\text{dofs}}}$, the generalized joint vector. The joint torques can only directly affect the joints themselves, and not the floating base dynamics, so we define an underactuated matrix $\boldsymbol{U} \in \mathbb{R}^{\left(n_{\text{dofs}}-6\right)\times n_{\text{dofs}}}$ that maps the global joint vector to the subspace of actuated joints.

$$\mathbf{q}_{\text{act}} = \boldsymbol{U}\ \mathbf{q}, \tag{2.1}$$

with $\mathbf{q}_{\text{act}} \in \mathbb{R}^{n_{\text{joints}}}$ being the robot's actuated joints. The dynamics of the robot's generalized joints can be described by a single, linear, second order differential equation

$$\boldsymbol{A}\ddot{\mathbf{q}} + \mathbf{b} + \mathbf{g} = \boldsymbol{U}^\top \boldsymbol{\tau}_{\text{control}}, \tag{2.2}$$

where $\{\boldsymbol{A}, \mathbf{b}, \mathbf{g}\}$ represent the inertia matrix, Coriolis/centrifugal forces, and gravitational forces, respectively while $\boldsymbol{\tau}_{\text{control}}$ is the desired control command applied to the output joints of the robot. Without considering contacts, or the subtle non-holonomic effect of angular momentum, joint torques would have no effect on the geometrically uncontrollable six-dimensional subspace of the generalized joints: $\{\mathbf{z} \in \mathbb{R}^{n_{\text{dofs}}} : \mathbf{z}^\top \boldsymbol{A}^{-1}\boldsymbol{U}^\top = \mathbf{0}\}$. However, we can sometimes gain the ability to control more of this space due to contact constraints.

We consider two contact cases for point-foot biped robots: single support in which one foot is in contact, and double support where the robot is supported by both feet. In the single support phase we describe the contact via a support Jacobian $\boldsymbol{J}_s \in \mathbb{R}^{3\times n_{\text{dofs}}}$, which maps from generalized joint velocity to the velocity of the constrained foot point in Cartesian space. When considering double contact, our support Jacobian represents twice as many constraints. Since this generalized point, either the single foot point in $\mathbb{R}^3$ or the double foot point in $\mathbb{R}^6$, is constrained, we know its acceleration must be zero. Substituting the constraint $\boldsymbol{J}_s\ddot{\mathbf{q}} + \dot{\boldsymbol{J}}_s\dot{\mathbf{q}} =$

$\ddot{\mathbf{x}}_{\text{foot(or feet)}} = 0$ and adding the associated co-state of constraint space reaction forces $\mathbf{F}_r$, the dynamics become

$$\boldsymbol{A}\ddot{\mathbf{q}} + \mathbf{b} + \mathbf{g} + \boldsymbol{J}_s^\top \mathbf{F}_r = \boldsymbol{U}^\top \boldsymbol{\tau}_{\text{control}}, \tag{2.3}$$

and we can find $\ddot{\mathbf{q}}$ and $\mathbf{F}_r$ by solving the matrix equation

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{J}_s^\top \\ \boldsymbol{J}_s & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \mathbf{F}_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}^\top \boldsymbol{\tau}_{\text{control}} - \mathbf{b} - \mathbf{g} \\ -\dot{\boldsymbol{J}}_s \dot{\mathbf{q}}. \end{bmatrix}, \tag{2.4}$$

Converting to upper diagonal form via Gaussian elimination we find

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{J}_s^\top \\ \mathbf{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \mathbf{F}_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}^\top \boldsymbol{\tau}_{\text{control}} - \mathbf{b} - \mathbf{g} \\ \overline{\boldsymbol{J}}_s^\top \left[\boldsymbol{U}^\top \boldsymbol{\tau}_{\text{control}} - \mathbf{b} - \mathbf{g} \right] + \boldsymbol{\Lambda}_s \dot{\boldsymbol{J}}_s \dot{\mathbf{q}} \end{bmatrix}, \tag{2.5}$$

with $\boldsymbol{\Lambda}_s \triangleq (\boldsymbol{J}_s \boldsymbol{A}^{-1} \boldsymbol{J}_s^\top)^{-1}$ and $\overline{\boldsymbol{J}}_s \triangleq \boldsymbol{A}^{-1} \boldsymbol{J}_s^\top \boldsymbol{\Lambda}_s$. Substituting $\boldsymbol{N}_s^\top \triangleq \boldsymbol{I} - \boldsymbol{J}_s^\top \boldsymbol{\Lambda}_s \boldsymbol{J}_s \boldsymbol{A}^{-1}$ to more conveniently express the resulting constrained dynamic equation,

$$\boldsymbol{A}\, \ddot{\mathbf{q}} + \boldsymbol{N}_s^\top (\mathbf{b} + \mathbf{g}) + \boldsymbol{J}_s^\top \boldsymbol{\Lambda}_s \dot{\boldsymbol{J}}_s \dot{\mathbf{q}} = (\boldsymbol{U}\boldsymbol{N}_s)^\top \boldsymbol{\tau}_{\text{control}}. \tag{2.6}$$

This can be viewed as constraining the dynamics to the dynamically consistent null-space of the constraint by defining the dynamically consistent pseudo inversion operator

$$\overline{\boldsymbol{X}} \triangleq \boldsymbol{A}^{-1} \boldsymbol{X}^\top [\boldsymbol{X} \boldsymbol{A}^{-1} \boldsymbol{X}^\top]^{-1} \tag{2.7}$$

and observing that

$$\boldsymbol{N}_s = \boldsymbol{I} - \overline{\boldsymbol{J}}_s \boldsymbol{J}_s \tag{2.8}$$

is the null space projector of $\boldsymbol{J}_s$ under dynamically consistent inversion such that $\boldsymbol{J}_s \boldsymbol{N}_s = \mathbf{0}$, $\boldsymbol{N}_s \boldsymbol{A}^{-1} \boldsymbol{J}_s^\top = \boldsymbol{A}^{-1} \boldsymbol{N}_s^\top \boldsymbol{J}_s^\top = \mathbf{0}$, and $\boldsymbol{N}_s \boldsymbol{N}_s = \boldsymbol{N}_s$.

WBOSC for an operational task representation, $\mathbf{p}_{\text{task}}$, is defined by the differential kinematic equation

$$\dot{\mathbf{p}}_{\text{task}} = \boldsymbol{J}^*_{\text{task}}\, \dot{\mathbf{q}}_{\text{act}}, \tag{2.9}$$

where $\boldsymbol{J}^*_{\text{task}} \triangleq \boldsymbol{J}_{\text{task}} \overline{\boldsymbol{U}\boldsymbol{N}}_s \in \mathbb{R}^{n_{\text{task}} \times n_{\text{act}}}$ is the contact consistent task Jacobian and $\boldsymbol{J}_{\text{task}} \in \mathbb{R}^{n_{\text{task}} \times n_{\text{dofs}}}$ is the unconstrained task Jacobian. The basic control structure for the single support phase of the biped is thus

$$\boldsymbol{\tau}_{\text{control}} = \boldsymbol{J}^{*T}_{\text{task}} \mathbf{F}_{\text{task}}, \tag{2.10}$$

with $\mathbf{F}_{\text{task}}$ being the entry point for feedback control laws to govern trajectories, applied forces, or combinations of the two in the operational space. For instance, when controlling an operational space position trajectory, we use the model-based control law

$$\mathbf{F}_{\text{task}} = \boldsymbol{\Lambda}^*_{\text{task}} \mathbf{u}_{\text{task}} + \boldsymbol{\mu}^*_{\text{task}} + \mathbf{p}^*_{\text{task}}, \tag{2.11}$$

with $\mathbf{u}_{\text{task}}$ being a desired acceleration for the operational reference. $\{\boldsymbol{\Lambda}^*_{\text{task}}, \boldsymbol{\mu}^*_{\text{task}}, \mathbf{p}^*_{\text{task}}\}$ are inertia, velocity-based forces, and gravity based forces in the operational space that can be found in the previous references.

In the case of double support, there appear closed loop effects between the legs of the robot in contact with the environment. Our previous work has thoroughly addressed this problem by creating structures to control the internal forces. Internal forces are defined as those that are orthogonal to joint accelerations. As such, internal forces do not produce any movement and only contribute to force generation within the closed-loop formed by the contacts. Analyzing the right hand side of Equation (2.6), those forces correspond to the manifold

$$\left(\boldsymbol{U}\boldsymbol{N}_s\right)^\top \boldsymbol{\tau}_{\text{control}} = \mathbf{0}, \tag{2.12}$$

which reflect the cancellation of acceleration effects. Therefore, the torques that fulfill the above constraint belong to the null space of $(\boldsymbol{U}\boldsymbol{N}_s)$, which is defined by the projection

$$\boldsymbol{L}^* \triangleq \left(\boldsymbol{I} - \boldsymbol{U}\boldsymbol{N}_s \, \overline{\boldsymbol{U}\boldsymbol{N}_s}\right). \tag{2.13}$$

The torques associated with internal forces are those that do not contribute to net movement, i.e.,

$$\boldsymbol{\tau}'_{\text{int}} = \boldsymbol{L}^{*\top} \boldsymbol{\tau}_{\text{int}}, \tag{2.14}$$

where $\boldsymbol{\tau}_{\text{int}}$ is the set-point for the internal forces. Thus, when simultaneously controlling operational space tasks and internal forces we superimpose the orthogonal structures of Equation (2.10) and (2.14) yielding the WBOSC command

$$\boldsymbol{\tau}_{\text{control}} = \boldsymbol{J}^{*\top}_{\text{task}} \mathbf{F}_{\text{task}} + \boldsymbol{L}^{*\top} \boldsymbol{\tau}_{\text{int}}. \tag{2.15}$$

Internal forces can be physically defined as linear forces and mutually canceling reaction moments between pairs of supporting contacts. As explained in our previous works, such forces are expressed via the equation,

$$\mathbf{F}_{\text{int}} = \boldsymbol{W}_{\text{int}} \, \mathbf{F}_r, \tag{2.16}$$

where $\mathbf{F}_r$ is the set of all reaction forces on the environment and $\boldsymbol{W}_{\text{int}}$ is a matrix containing geometric transformations and selection criteria to extract the internal forces. Using this mapping, we demonstrated that the dynamics of internal forces correspond to

$$\mathbf{F}_{\text{int}} \;=\; \left(\overline{\boldsymbol{J}}^{*}_{i|l}\right)^{\top} \boldsymbol{\tau}_{\text{int}} + \mathbf{F}_{\text{int},\{t\}} - \boldsymbol{\mu}^{*}_{i} - \mathbf{p}^{*}_{i}, \tag{2.17}$$

where $\overline{\boldsymbol{J}}^{*}_{i|l} \triangleq \left(\boldsymbol{L}^{*}\boldsymbol{U}\overline{\boldsymbol{J}}_{s}\boldsymbol{W}^{\top}_{\text{int}}\right)$. Additionally, $\mathbf{F}_{\text{int},\{t\}}$ are internal forces induced by task behavior, i.e. $\mathbf{F}_{\text{int},\{t\}} \triangleq \boldsymbol{W}_{\text{int}}\overline{\boldsymbol{J}}^{\top}_{s}\boldsymbol{J}^{*\top}_{\text{task}}\mathbf{F}_{\text{task}}$, and $\boldsymbol{\mu}^{*}_{i}$ and $\mathbf{p}^{*}_{i}$ are Coriolis/ centrifugal and gravitational effects on the internal forces. Inverting the above equation, we derive the torques needed to accomplish a desired internal force,

$$\boldsymbol{\tau}_{\text{int}} \;=\; \boldsymbol{J}^{*\top}_{i|l}\Big(\mathbf{F}_{\text{int,ref}} - \mathbf{F}_{\text{int},\{t\}} + \boldsymbol{\mu}^{*}_{i} + \mathbf{p}^{*}_{i}\Big), \tag{2.18}$$

where $\boldsymbol{J}^{*}_{i|l}$ is the Moore-Penrose left-pseudoinverse of $\overline{\boldsymbol{J}}^{*}_{i|l}$, also referred to as the reduced Jacobian of internal forces acting on the contact closed loops, and $\mathbf{F}_{\text{int,ref}}$ is the vector of desired internal forces which we use as an entry point to control internal forces.

2.2 Functional Extensions to WBC

The extensions introduced in this chapter are general formulations applicable to any form of WBCs except contact switching transitions. Our contact switching transition method is designed for algorithms using projection-based contact constraint. All the others are general formulations for any WBCs. Here, we devise new formulations of the centroidal momentum task and the capture point task, which are crucial to enhancing the balancing capability of legged systems. And we found time derivative terms of point jacobians and the CM task jacobian.

2.2.1 Contact Switching Transitions

To reduce the high-speed behavior caused by a sudden change in joint torques, we devise a strategy that smoothens out the torque commands when the robot transitions between single and double support. The sudden change in torque command is due to the instantaneous switching between constraint sets within WBOSC. When the controller uses a single contact constraint, it returns $\boldsymbol{\tau}_{\text{single}}$. When it uses a double contact constraint, it returns $\boldsymbol{\tau}_{\text{double}}$. To ensure a smooth torque trajectory, we employ $\boldsymbol{\tau}_{\text{trans}}$, which is a unique instance of WBOSC that uses a single contact constraint but adds an artificial transition force associated with the contact behavior of the transitioning foot. The transitioning foot corresponds to the foot that just landed on

the ground or the foot that is about to lift up from the ground. In either case, we do not instantaneously switch the controller between single and double contact phases, but instead transition smoothly from single contact to double contact or vice versa. And the manner that we transition is by adding the artificial transition force to the single contact controller and slowly increasing or decreasing its value until it matches the final torques of the next phase. That is, for the single contact transitioning controller, we adjust Equation (2.11) to include an artificial transitional force, f_{trans}, as follows:

$$\mathbf{F}_{\text{task}} = \boldsymbol{\Lambda}^*_{\text{task}}\mathbf{u}_{\text{task}} + \boldsymbol{\mu}^*_{\text{task}} + \mathbf{p}^*_{\text{task}} + \mathbf{f}_{\text{trans}}. \tag{2.19}$$

Based on the double support hypothesis, $\boldsymbol{\tau}_{\text{double}}$, we find the reaction force that would be generated on the foot that just landed on the ground or that is generated on foot about to take off. Using the second row of the matrix Equation (2.5), and based on the forces corresponding to the task set in single contact phase resulting from the double contact phase reaction forces, i.e. $\mathbf{f}_{\text{double}\to\text{single}} \triangleq \boldsymbol{S}_{\text{double}\to\text{single}}\mathbf{F}_r$, we get

$$\mathbf{f}_{\text{double}\to\text{single}} = \boldsymbol{S}_{\text{double}\to\text{single}}\left(\overline{\boldsymbol{J}}_s^{\top}\left[\boldsymbol{U}^{\top}\boldsymbol{\tau}_{\text{double}} - \mathbf{b} - \mathbf{g}\right] + \boldsymbol{\Lambda}_s\dot{\boldsymbol{J}}_s\dot{\mathbf{q}}\right), \tag{2.20}$$

where $\boldsymbol{S}_{\text{double}\to\text{single}}$ is a projection operator that transforms the double contact reaction forces to equivalent task forces of the single contact controller. Let us take the case of the swinging foot that just landed on the ground. Rather than switching instantaneously to the double contact controller, we continue using the single contact controller with the artificial transition force. That force smoothly increases until it matches the double contact reaction forces projected to the single contact controller. Therefore, as the controller transition approaches to double support we increase the artificial force from $(\mathbf{f}_{\text{trans}} = \mathbf{0}) \to (\mathbf{f}_{\text{trans}} = \mathbf{f}_{\text{double}\to\text{single}})$. Conversely, after the controller switches from double support phase to single support phase prior to lifting the foot from the ground, we smoothen the transition by gradually removing the artificial force, $(\mathbf{f}_{\text{trans}} = \mathbf{f}_{\text{double}\to\text{single}}) \to (\mathbf{f}_{\text{trans}} = \mathbf{0})$. To achieve this gradouble transition, we define $\mathbf{f}_{\text{trans}} \triangleq w \cdot \mathbf{f}_{\text{double}\to\text{single}}$, $w \in [0, 1]$, where w varies linearly with time from one to zero or from zero to one over the course of the transition as appropriate.

To avoid running the controller twice every servo cycle during transitions, we reuse previous values for $\boldsymbol{\tau}_{\text{double}}$ and $\mathbf{f}_{\text{double}\to\text{single}}$. When a foot lifts, the old value of $\boldsymbol{\tau}_{\text{double}}$ is simply the last controller action before the start of the transition, and $\mathbf{f}_{\text{double}\to\text{single}}$ is calculated once based on that controller state. When a foot lands, we run the double support controller once at the start of the transition for the sole purpose of acquiring $\mathbf{f}_{\text{double}\to\text{single}}$. This process is similar to [32]

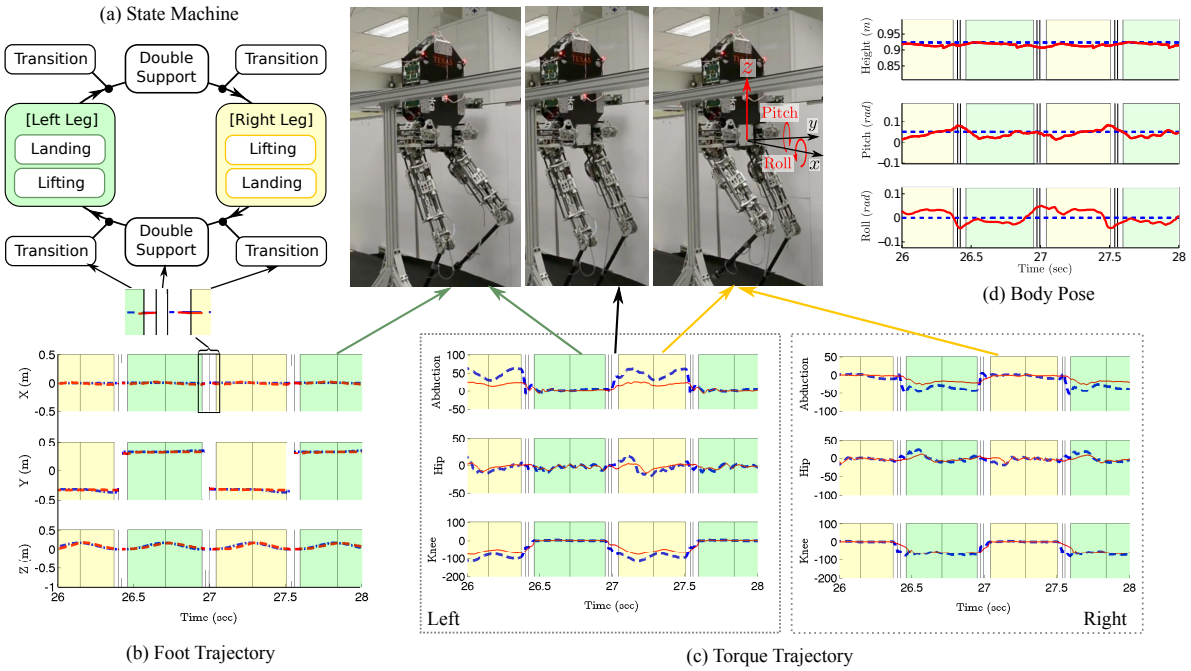


Figure 2.1: **State Machine and Stepping Motion Test.** (a) Motion within the stepping experiment is divided into three states: dual support, swing leg lift, and swing leg landing. Additionally, a transition state exists at the beginning of the lifting phase and the end of the landing phase in order to avoid the jerk caused by a sudden change of effective mass. Desired foot trajectories, blue, and sensor data, red, exhibit good tracking performance in Subfigure (b). In Subfigure (c), desired torque trajectories, blue, are plotted alongside sensor measured torques, in red. Background color indicates the phase of walking, with green corresponding to the left leg swing phase, yellow to the right leg swing phase, and white to the dual support phase. In Subfigure (d), position and orientation task tracking is plotted over a representative portion of the stepping experiment, with sensor data in red and desired values, dotted, in blue. Height refers to CoM height.

except that they apply it to a quadratic programming based controller whereas we apply it to a WBOSC-based controller.

Fig. 2.1 shows the stepping experiment offering a preliminary look at the performance of the control system with regard to achieving dynamic walking. It allows us to evaluate the performance of the transition strategy. In this experimental setup, Hume's x direction of motion was restricted by locking the sliding degree of freedom within the planarizing linkage. The robot's feet, placed roughly beneath the center of mass of the system, were lifted one at a time as though the robot were marching in place. Rather than using the planner, desired footstep location was held constant for each foot. This stepping in place motion followed a time scripted state machine, shown in Fig. 2.1(a). Since the states are symmetric with respect to the supporting leg being either right or left, states are categorized in two different task sets with left and right single support having symmetric structures. The WBOSC task set differs between dual support and other phases of the stepping state machine.

As shown in Fig. 2.1(d), the robot's tasks were to control CoM height, body pitch and roll angles in dual support. Therefore, in dual support, $\mathbf{x}^d_{\text{task}}$ is $[\text{CoM}_z,\ q_{Ry},\ q_{Rx}]^T$. In single support, we control foot position as well, therefore it changes to $[\text{CoM}_z,\ q_{Ry},\ q_{Rx},\ foot_x,\ foot_y,\ foot_z]^T$. Transitions are used, with $\mathbf{F}^d_{\text{task}} = [0,\ 0,\ 0, wf_{r,x},\ wf_{r,y},\ wf_{r,z}]^T$, where $w \in [0, 1]$ is the scaling factor and $f_{r,-}$ is the expected reaction force in the dual support case.The gain scheduling algorithm described in Section 5.1.1 is also used. With a positional CoM error bounded within 2 cm and an orientation error within 0.15 rad, as can be verified in Fig. 2.1(d), we can conclude that the controller has the potential to achieve closed loop standing through repeated stepping. Torque tracking performance for the low level controllers is displayed in Fig. 2.1(b). The results show larger torque tracking errors in the single supporting leg than in the swing leg. When the gain scheduling algorithm enables the integral gain on torque tracking, during swing, this error becomes far smaller, which is necessary in order to overcome the friction internal to the actuator when moving the lightweight shank link of the swing leg. Fig. 2.1(c) shows that the torque transitions gently change between each step, as a result of the contact transitioning procedure. Even when the command changes from 100 N to 0 N, this transition is handled gracefully.

2.2.2 Formulation of Centroidal Momentum Task

The centroidal momentum (CM) of a robot is the aggregate momentum on the robot's CoM. The robot's CM vector, $\mathbf{h}_G$, is related to its joint velocity vector as $\mathbf{h}_G = \boldsymbol{A}_G\dot{\mathbf{q}}$. Alternatively,

Position Gain in Dual Support

	K_x	I_x	D_x
CoM_z	450.0	55.0	5.0
q_{Ry}	400.0	55.0	5.0
q_{Rx}	80.0	10.0	5.0

Position Gain in Single Support

	K_x	I_x	D_x
CoM_z	470.0	60.0	10.0
q_{Ry}	300.0	60.0	5.0
q_{Rx}	100.0	10.0	5.0
$foot_x$	900.0	100.0	100.0
$foot_y$	900.0	100.0	50.0
$foot_z$	905.0	100.0	50.0

Default Torque Gain

	$K_{P,\tau}$	$K_{I,\tau}$
Every Joint	65	0

Torque Gain of Swing Leg

	$K_{P,\tau}$	$K_{I,\tau}$
Hip right	200	22
Hip left	180	15
Knee right	260	35
Knee left	255	30

Table 2.1: **Gain Set for the Stepping Test.**

this can be expressed as the product of the robot's centroidal inertia, $\mathrm{I_G}$, and its average velocity $\mathbf{v}_G$:

$$\mathbf{h}_G = \boldsymbol{I}_G \mathbf{v}_G = \boldsymbol{A}_G \dot{\mathbf{q}}. \tag{2.21}$$

Using an adjoint operator to change the reference coordinate, and defining the inertia of each link as I_i, I_G and A_G can be expressed as

$$\boldsymbol{I}_G = \sum_i \mathrm{Ad}^*_{T_i^{-1}} \boldsymbol{I}_i \mathrm{Ad}_{T_i^{-1}}, \tag{2.22}$$

$$\boldsymbol{A}_G = \sum_i \mathrm{Ad}^*_{T_i^{-1}} \boldsymbol{I}_i \boldsymbol{J}_i, \tag{2.23}$$

where T_i is the $SE(3)$ of the local frame of each link seen from the center of mass frame, consisting of a rotational representation ($\mathrm{R_i}$) and linear position ($\mathbf{p}_i$),

$$T_i = \begin{bmatrix} \boldsymbol{R}_i & \mathbf{p}_i \\ 0 & 1 \end{bmatrix}, \tag{2.24}$$

and Ad_{T_i} and $Ad^*_{T_i}$ are the adjoint and dual adjoint mapping, defined as

$$\mathrm{Ad}_{T_i} = \begin{bmatrix} \boldsymbol{R}_i & 0 \\ [\mathbf{p}_i]^\times \boldsymbol{R}_i & \boldsymbol{R}_i \end{bmatrix}, \tag{2.25}$$

$$\mathrm{Ad}^*_{T_i} = \begin{bmatrix} \boldsymbol{R}_i & 0 \\ [\mathbf{p}_i]^\times \boldsymbol{R}_i & \boldsymbol{R}_i \end{bmatrix}^\top, \tag{2.26}$$

where $[p]^\times$ is the function which changes a vector to a skew symmetric matrix. From the above equations, we can easily identify the Jacobian with the following equation,

$$\boldsymbol{J}_{\mathrm{CM}} = \boldsymbol{I}_G^{-1} \boldsymbol{A}_G. \tag{2.27}$$

Eq (2.27) represents the velocity mapping between the CM space (operational space) and configuration space. We newly incorporate this mapping for creating CM tasks in our WBC framework.

2.2.3 Formulation of Capture Point Task

In [73], capture point (CP) is defined as follows: For a biped in state x, a capture point, p, is a point on the ground where if a biped either covers p with its stance foot or steps onto p and then maintains its center of pressure (CoP) on p, then there is a safe feasible trajectory that the robot will end in a captured state. Therefore, the first objective of the balance controller

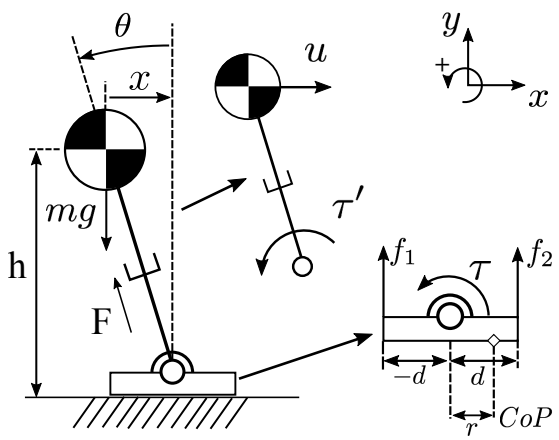


Figure 2.2: **CoM and thin foot Model** A single supported legged system can be modeled as a flat foot connected with rotational joint (ankle) and prismatic joint (leg).

is to move the CoP to the CP of a given state and then add feedback to move the CP to the desired CP location. [16] and [97] explain how to design a biped's balance controller by manipulating the CoP (or equivalently, ZMP). However, it is not trivial to define the Jacobian for a CoP control task since there is no direct velocity relationship between the CoP space and the configuration space. Additionally, the CoP can only be changed by moving the CoM. Therefore, we propose the simplified model (Fig. 2.2) to design a CP controller in our WBC framework. Intuitively speaking, the following process finds the relationship between the CoM acceleration (WBC input) and the CoP dynamics, which will be used to manipulate the CP.

The dynamics of the robot's CoM are

$$mg = F\cos\theta, \tag{2.28}$$

$$m\ddot{x} = -\frac{\tau'}{h} - F\sin\theta. \tag{2.29}$$

Dividing the two equations and substituting $\tan\theta = -\frac{x}{h}$ results into

$$\frac{m\ddot{x} + \frac{\tau'}{h}}{mg} = \frac{x}{h}, \tag{2.30}$$

$$\frac{g}{h}x - \frac{\tau'}{mh} = \ddot{x}. \tag{2.31}$$

Ignoring horizontal forces on the foot, the force/moment balance equations on the supporting

foot are

$$f_1 + f_2 = mg, \tag{2.32}$$

$$-f_1 d + \tau + f_2 d = 0, \tag{2.33}$$

$$-f_1(d + r) + f_2(d - r) = 0. \tag{2.34}$$

From those equations, we can identify the relationship between torque and CoP position (r).

$$\frac{(f_2 - f_1)d}{f_1 + f_2} = r, \tag{2.35}$$

$$-\frac{\tau}{mg} = r. \tag{2.36}$$

Because $\tau' = -\tau$, plugging Eq (2.36) into Eq (2.31) results in a linear ODE

$$\ddot{x} = \frac{g}{h}(x - r). \tag{2.37}$$

This equation implies that the ankle torque does not change the linear inverted pendulum dynamics above and influences only the CoP position. From the original CP formulation, $x_{\mathrm{CP}} = x + \sqrt{\frac{h}{g}}\dot{x}$. The CP dynamics are defined as

$$\dot{x}_{\mathrm{CP}} = \sqrt{\frac{g}{h}}(x_{\mathrm{CP}} - r). \tag{2.38}$$

Then, a suitable control law of the CoP to let the current CP converge to desired CP is

$$r = x_{\mathrm{CP}} + K(x_{\mathrm{CP}} - x^d_{\mathrm{CP}}), \tag{2.39}$$

where x^d_{CP} is the desired CP location and K is the gain. Inserting the above equation (2.39) into Eq. (2.37) and considering an operational task associated with the robot's center of mass, $\dot{\mathbf{x}} = \boldsymbol{J}_{\mathrm{CoM}}\dot{\mathbf{q}}$, the desired CoM acceleration input to control the CP in WBC is

$$\ddot{x} = u = \frac{g}{h}(x - x_{\mathrm{CP}}) + K'(x^d_{\mathrm{CP}} - x_{\mathrm{CP}}) \tag{2.40}$$

$$= \frac{g}{h}(x - x - \sqrt{\frac{h}{g}}\dot{x}) + K'(x^d_{\mathrm{CP}} - x_{\mathrm{CP}}) \tag{2.41}$$

$$= -\sqrt{\frac{g}{h}}\dot{x} + K'(x^d_{\mathrm{CP}} - x_{\mathrm{CP}}), \tag{2.42}$$

where $K' = K\frac{g}{h}$ and J_{CoM} is the Jacobian of the robot's Center of Mass. y directional capture point control is done in the same way.

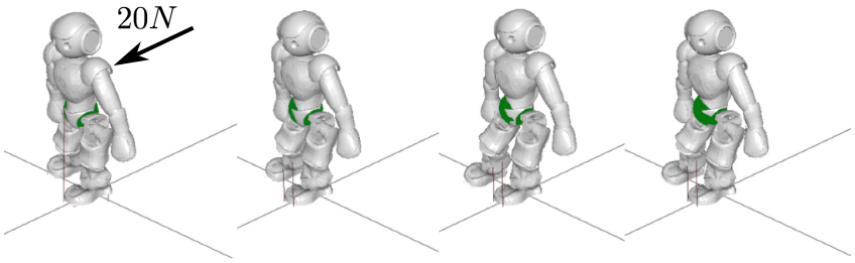


Figure 2.3: **Balance Control with Capture Point Task.** NAO's balance is disturbed by a constant 20N push force for 0.1 s. By manipulating the capture point with whole-body motion, NAO recovers and continues to balance on its left foot

2.2.4 Balance Control With Capture Point and Centroidal Angular Momentum Tasks

In the simulation, we assume that the NAO robot is a fully torque controllable humanoid. This assumption provides an environment to test highly dynamic motions and to verify that our task extensions to WBC function as intended. Namely, the CP task balances the robot, and the addition of CAM task further enhances the balancing capability. To test the balance controller in the simulation, we induce a disturbance by pushing the NAO robot on its torso with a constant force for 0.1 s while NAO balances on its left foot. First, we test a CP-based balance controller which only uses the CP criteria. This test is performed to verify the proposed control law in Eq (2.42). As seen in Fig. 2.3, we push the torso with 20N of force for 0.1 s. The simulation showed that the NAO robot successfully maintains its balance on its left foot. Second, we test how much the CAM task helps with maintaining balance. In this test, we add 15N of constant force perpendicular to the first force. As before, these two forces are induced for 0.1 s. Fig. 2.4 shows two tests being performed: (1) with CP + CAM Control, and (2) with CP Control only. Note that when performing the combined CP + CAM balance controller, we place the CP criteria as a higher priority task than the CAM task. This is because the CAM task acts as a supplement, which attempts to minimize the centroidal angular velocity of the robot. As Fig. 2.4 shows, when the combined CP + CAM Controller is used, the NAO robot uses its limbs more actively and successfully balances. Otherwise, using only a CP-based controller under these forces fail. Thus, the inclusion of CAM control contributes significantly to balance control.

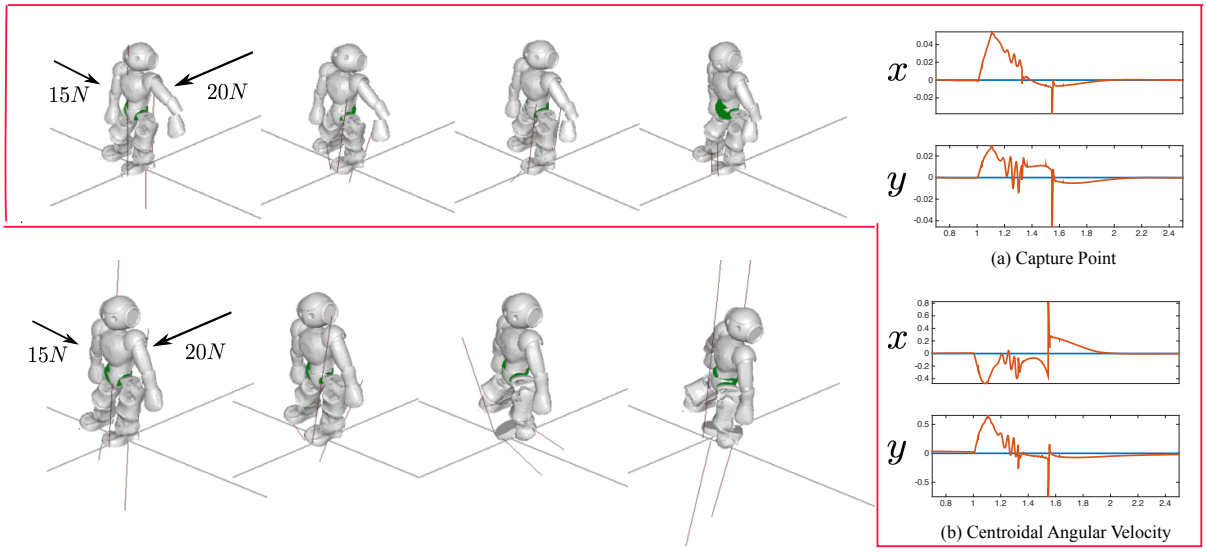


Figure 2.4: **Comparison of Balance Controllers.** In these two trials, we push the NAO with 15N and 20N on the lateral and sagittal direction respectively for 0.1 s. The first row is the combined CP + CAM balance controller and the second row only uses a CP-based balance controller. The images boxed in red shows the tracking performance of the actual (orange line) CP and CAM trajectory to the desired (blue line) trajectories for the CP + CAM balance control test. The simulation shows that under these forces, the CP + CAM balance controller successfully balances the robot, but fails with only a CP-based controller.

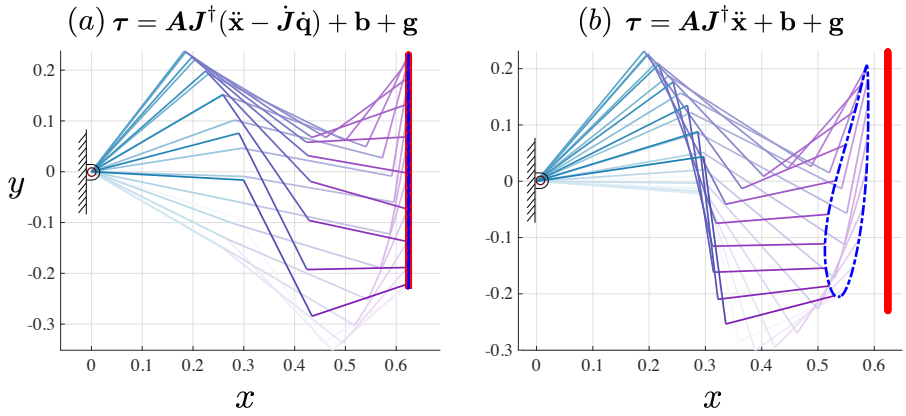


Figure 2.5: **Tracking performance comparison with and without the term $\dot{\boldsymbol{J}}\dot{\mathbf{q}}$.** A three-DoF planar manipulator is used to control its end effector to follow a vertical line (red lines) with 2 Hz frequency (blue dashed lines are the end-effector path). The tracking results demonstrte that the (a) controller, which accounts for $\dot{\boldsymbol{J}}\dot{\mathbf{q}}$, outperforms the (b) controller.

2.2.5 Time Derivative of Jacobian

The ability to compute efficiently the time derivative of Jacobian operators for fast operational space control has been overlooked. However it plays an important role on robustifying fast movements. Fig.2.5 shows that the tracking performance of a simple serial manipulator is significantly enhanced by using the term $\dot{\boldsymbol{J}}\dot{\mathbf{q}}$ via operational space control (OPC), where $\boldsymbol{J}$ is the Jacobian of the end effector and $\dot{\mathbf{q}}$ is the vector of joint velocities. The commanded task is to follow a vertical line defined by the function, $\mathbf{x}^d = [\ 0.62,\ 0.23\sin(4\pi t)\ ]^{\top}$. The OPC input is

$$\ddot{\mathbf{x}} = \begin{bmatrix} 0 \\ -36.32\sin(4\pi t) \end{bmatrix} + K_p(\mathbf{x}^d - \mathbf{x}) + K_v(\dot{\mathbf{x}}^d - \dot{\mathbf{x}}). \tag{2.43}$$

We will use Lie group theory to compute the derivatives of point task Jacobians [52]. We implement this functionality using the Rigid Body Dynamics Library [1], which is a popular open source dynamics toolbox. In addition, we also devise a new method to compute the time derivative of the CM Jacobian which cannot be computed using Lie group theory.

Time Derivative of Point Jacobian

Lie group operators provide convenient analytic derivations for Jacobian computations. The $SE(3)$ orientation and position representation of a rigid body in three-dimensional space consists

[1] Open-source `https://rbdl.bitbucket.io`

of orientation matrix ($\boldsymbol{R}$) and a position vector ($\mathbf{p}$). It can also be represented via the 4×4 homogeneous transformation,

$$\boldsymbol{T}_{g,i} = \begin{bmatrix} \boldsymbol{R}_{g,i} & \mathbf{p}_{g,i} \\ 0 & 1 \end{bmatrix}, \tag{2.44}$$

where, $\boldsymbol{R}_{g,i}$ and $\mathbf{p}_{g,i}$ represent the orientation and position of the i^{th} frame in global coordinates respectively (See Fig. 2.6). The velocity representation in $se(3)$ consists of the 6-dimensional vector, $[\mathbf{w}, \mathbf{v}]^T$, and yields the 4×4 homogeneous equality,

$$\boldsymbol{V}_i \triangleq \begin{bmatrix} [\mathbf{w}_i]^\times & \mathbf{v}_i \\ 0 & 0 \end{bmatrix}, \tag{2.45}$$

where

$$[\mathbf{w}_i]^\times \triangleq \begin{bmatrix} 0 & -w_{i,3} & w_{i,2} \\ w_{i,3} & 0 & -w_{i,1} \\ -w_{i,2} & w_{i,1} & 0 \end{bmatrix} \tag{2.46}$$

Here, $w_{i,1}, w_{i,2}, w_{i,3}$ are relative angular velocities along the three Cartesian axes, and $\mathbf{v}_i$ is the linear velocity. It can be shown that $\boldsymbol{V}_i = \boldsymbol{T}_{g,i}^{-1}\dot{\boldsymbol{T}}_{g,i}$, and corresponds to the generalized velocity seen from the i^{th} frame. The velocity in the global frame associated with $\boldsymbol{V}_i$ can be obtained via adjoint derivations,

$$\begin{aligned} \mathrm{Ad}_{T_{g,i}}(\boldsymbol{V}_i) &= \boldsymbol{T}_{g,i}\boldsymbol{V}_i\boldsymbol{T}_{g,i}^{-1} \\ &= \boldsymbol{T}_{g,i}\boldsymbol{T}_{g,i}^{-1}\dot{\boldsymbol{T}}_{g,i}\boldsymbol{T}_{g,i}^{-1} \\ &= \dot{\boldsymbol{T}}_{g,i}\boldsymbol{T}_{g,i}^{-1}. \end{aligned} \tag{2.47}$$

The adjoint mapping operator is defined as

$$\mathrm{Ad}_{T_{i,j}} \triangleq \begin{bmatrix} \boldsymbol{R}_{i,j} & 0 \\ [\mathbf{p}_{i,j}]^\times \boldsymbol{R}_{i,j} & \boldsymbol{R}_{i,j} \end{bmatrix}, \tag{2.48}$$

where $\boldsymbol{R}_{i,j}$ and $\mathbf{p}_{i,j}$ are relative rotations and positions between points i and j. The generalized velocity of point p in local coordinates (see Fig. 2.6) can be represented as

$$\begin{aligned} \mathbf{V}_p &= \mathrm{Ad}_{T_{p,n}}\mathbf{V}_n \\ &= \mathrm{Ad}_{T_{p,n-1}}\mathbf{V}_{n-1} + \mathrm{Ad}_{T_{p,n}}\mathbf{S}_n\dot{\mathbf{q}}_n \\ &\vdots \\ &= \mathrm{Ad}_{T_{p,0}}\mathbf{V}_0 + \sum_{i=1}^{n} \mathrm{Ad}_{T_{p,i}}\mathbf{S}_i\dot{\mathbf{q}}_i. \end{aligned} \tag{2.49}$$

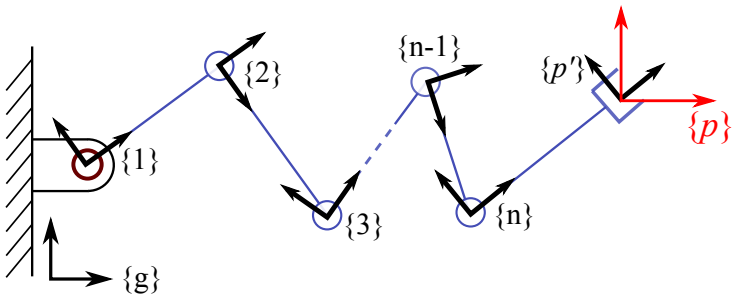


Figure 2.6: **Multi-DoF Openchain.** The openchain consists of n joints. At the end of the chain, the end-effector is attached to link n.

Because $\{0\}$ frame is the global frame (an inertial frame), $\mathbf{V}_0$ is equal to zero. On the other hand $\mathbf{S}_i$ maps joint velocities to R^6, e.g. $\mathbf{S}_i$ is $[\ 0,0,1,0,0,0\ ]^T$ means $\dot{q}_i$ is a revolute joint rotating along the z local axis. The first three positions of $\mathbf{S}_i$ represent rotational axes while the last three positions represent prismatic axes. It can be shown that the Jacobian of a point p, is equal to

$$\boldsymbol{J}_p = \begin{bmatrix} \mathrm{Ad}_{T_{p,1}}\mathbf{S}_1 & Ad_{T_{p,2}}\mathbf{S}_2 & \cdots & \mathrm{Ad}_{T_{p,n}}\mathbf{S}_n \end{bmatrix} \tag{2.50}$$

Furthermore, let's break down the adjoint operators into the following chain operation

$$\mathrm{Ad}_{T_{p,i}}\mathbf{S}_i = \mathrm{Ad}_{T_{p,p'}}\mathrm{Ad}_{T_{p',n}}\mathrm{Ad}_{T_{n,i}}\mathbf{S}_i. \tag{2.51}$$

Here p' is a virtual point representing the position of p but with local orientation with respect to frame n. As such it represents just a position offset. In this case $\boldsymbol{T}_{p',n}$ is constant and the i-th column of the time derivative of the Jacobian can be resolved as

$$\begin{aligned} \dot{\boldsymbol{J}}_{p,i} &= \dot{\overbrace{\left\{\mathrm{Ad}_{T_{p,i}}\mathbf{S}_i\right\}}} \\ &= \dot{\overbrace{\left\{\mathrm{Ad}_{T_{p,p'}}\right\}}}\mathrm{Ad}_{T_{p',n}}\mathrm{Ad}_{T_{n,i}}\mathbf{S}_i + \mathrm{Ad}_{T_{p,p'}}\mathrm{Ad}_{T_{p',n}}\dot{\overbrace{\left\{\mathrm{Ad}_{T_{n,i}}\mathbf{S}_i\right\}}} \\ &= \mathrm{Ad}_{T_{p,p'}}\mathrm{ad}_{V_{p,p'}}\mathrm{Ad}_{T_{p',n}}\mathrm{Ad}_{T_{n,i}}\mathbf{S}_i + \mathrm{Ad}_{T_{p,p'}}\mathrm{Ad}_{T_{p',n}}\left\{\mathrm{Ad}_{T_{n,i}}\mathrm{ad}_{V_{n,i}}\mathbf{S}_i + \mathrm{Ad}_{T_{n,i}}(\dot{\mathbf{S}}_i)\right\} \end{aligned} \tag{2.52}$$

Here we have used $\dot{\mathrm{Ad}}_T = \mathrm{Ad}_T(\mathrm{ad}_V)$ and $\boldsymbol{V} = \boldsymbol{T}^{-1}\dot{\boldsymbol{T}}$. ad_V is defined by

$$\mathrm{ad}_V \triangleq \begin{bmatrix} [\boldsymbol{\omega}] & \mathbf{0} \\ [\mathbf{v}] & [\boldsymbol{\omega}] \end{bmatrix}, \tag{2.53}$$

where $\mathbf{V} = [\boldsymbol{\omega}, \mathbf{v}]^\top$.

Time Derivative of the Centroidal Momentum Jacobian

The equations for the time derivative of point Jacobians are not applicable to the CM Jacobian. The latter can be obtained from the CM task definition of

$$\boldsymbol{I}_{cm}\ddot{\mathbf{x}}_{cm}^{d} = \mathbf{F}_{cm} = \begin{bmatrix} \boldsymbol{W}_{ang}\mathbf{F}_r \\ \Sigma\ \mathbf{F}_{r,lin} \end{bmatrix}, \tag{2.54}$$

of which the linear part is simply the weighted sum of time derivatives of each link's CoM Jacobian. However, the angular part is not straightforward. Instead of finding $\dot{\boldsymbol{J}}_{cm}$, we can find the multiplication of $\dot{\boldsymbol{J}}_{cm}$ and the joint velocities, $\dot{\mathbf{q}}$, via operational space dynamics:

$$\boldsymbol{\Lambda}_{cm}(\mathbf{q})\ddot{\mathbf{x}} + \boldsymbol{\mu}_{cm}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{p}_{cm}(\mathbf{q}) = \mathbf{F}_r, \tag{2.55}$$

where

$$\begin{aligned} \boldsymbol{\Lambda}_{cm} &= (\boldsymbol{J}_{cm}\boldsymbol{A}\boldsymbol{J}_{cm}^{\top})^{-1}, \\ \boldsymbol{\mu}_{cm} &= \boldsymbol{\Lambda}_{cm}(\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\mathbf{b} - \dot{\boldsymbol{J}}_{cm}\dot{\mathbf{q}}), \\ \mathbf{p}_{cm} &= \boldsymbol{\Lambda}\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\mathbf{g}. \end{aligned} \tag{2.56}$$

Here, $\boldsymbol{\Lambda}_{cm}$, $\boldsymbol{\mu}_{cm}$, and $\mathbf{p}_{cm}$ are an inertia matrix, coriolis and centrifugal force, and gravitational force of the CM operational task, respectively. Since there is no coriolis and centrifugal effects in CM space, $\boldsymbol{\mu}_{cm}$ is zero. Thus, $\dot{\boldsymbol{J}}_{cm}\dot{\mathbf{q}}$ must be equal to $\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\mathbf{b}$:

$$\dot{\boldsymbol{J}}_{cm}\dot{\mathbf{q}} = \boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\mathbf{b}. \tag{2.57}$$

All terms in $\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\mathbf{b}$ are easily computable using off-the-shelf dynamics libraries.

2.3 Whole-Body Locomotion Control

We devise a new whole-body locomotion control algorithm, dubbed WBLC, that specifies tasks using a hierarchy of accelerations and uses quadratic programming to determine contact forces. Fig. 2.7 describes the overall process for computing the torque commands. The details are described below.

2.3.1 Acceleration-based Formula with Hierarchy

Task level controllers are computed in operational space as acceleration commands and converted to joint accelerations using differential forward kinematics,

$$\begin{aligned} \dot{\mathbf{x}}_1 &= \boldsymbol{J}_1\dot{\mathbf{q}}, \\ \ddot{\mathbf{x}}_1 &= \boldsymbol{J}_1\ddot{\mathbf{q}} + \dot{\boldsymbol{J}}_1\dot{\mathbf{q}}, \end{aligned} \tag{2.58}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{q} \in \mathbb{R}^m$ represent the task's operational coordinate and the joint positions, respectively, and $\boldsymbol{J}$ is the corresponding Jacobian matrix. Then, the joint acceleration for a desired task acceleration, $\ddot{\mathbf{x}}_1^d$ can be resolved as

$$\ddot{\mathbf{q}}_1 = \overline{\boldsymbol{J}}_1 \left(\ddot{\mathbf{x}}_1^d - \dot{\boldsymbol{J}}_1 \dot{\mathbf{q}} \right) = \overline{\boldsymbol{J}}_1 \ddot{\mathbf{e}}_1^d, \tag{2.59}$$

where $\overline{\boldsymbol{J}}_1$ indicates the dynamically consistent inverse of $\boldsymbol{J}_1$, i.e.

$$\overline{\boldsymbol{J}}_1 = \boldsymbol{A}^{-1} \boldsymbol{J}_1^\top \left(\boldsymbol{J}_1 \boldsymbol{A}^{-1} \boldsymbol{J}_1^\top \right)^{-1}, \tag{2.60}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times m}$ indicates the mass/inertia matrix of the rigid body model of the robot. If we consider now the mapping of two operational tasks $\ddot{\mathbf{x}}_1^d$ and $\ddot{\mathbf{x}}_2^d$, we propose the following task hierarchy mapping

$$\ddot{\mathbf{q}} = \overline{\boldsymbol{J}}_1 \ddot{\mathbf{e}}_1^d + \overline{\boldsymbol{J}}_{2|1} \left(\ddot{\mathbf{e}}_2^d - \boldsymbol{J}_2 \ddot{\mathbf{q}}_1 \right), \tag{2.61}$$

where $\overline{\boldsymbol{J}}_{2|1} \triangleq \overline{(\boldsymbol{J}_2 \boldsymbol{N}_1)}$ represents the Jacobian associated with the second task, $\boldsymbol{J}_2$, projected into the null space of the first task, $\boldsymbol{N}_1 = \boldsymbol{I} - \overline{\boldsymbol{J}}_1 \boldsymbol{J}_1$, which by definition is orthogonal to the Jacobian associated with the first task, $\boldsymbol{J}_1$. The Equation (2.61) can be extended to the general n task case, using the following hierarchy

$$\ddot{\mathbf{q}}_{[task]} = \overline{\boldsymbol{J}}_1 \ddot{\mathbf{e}}_1^d + \sum_{k=2}^{n} \ddot{\mathbf{q}}_k, \quad (n \geq 2) \tag{2.62}$$

with

$$\begin{aligned}
&\ddot{\mathbf{q}}_k = \overline{\boldsymbol{J}}_{k|prec(k)} \left(\ddot{\mathbf{e}}_k^d - \boldsymbol{J}_k \sum_{i=1}^{k-1} \ddot{\mathbf{q}}_i \right), \\
&\boldsymbol{J}_{k|prec(k)} = \boldsymbol{J}_k \boldsymbol{N}_{prec(k)}, \\
&\boldsymbol{N}_{prec(k)} = \prod_{s=1}^{k-1} \boldsymbol{N}_{s|s-1} \quad (k \geq 2, \quad \boldsymbol{N}_{1|0} = \boldsymbol{N}_1), \\
&\boldsymbol{N}_{s|s-1} = \boldsymbol{I} - \overline{\boldsymbol{J}}_{s|prec(s)} \boldsymbol{J}_{s|prec(s)} \quad (s \geq 2).
\end{aligned} \tag{2.63}$$

This task hierarchy is similar, albeit not identical to [91]. Compared to it, our proposed method is more concise, resulting in less computations for similar control specifications. In particular we do not require the computation of time derivatives of prioritized Jacobians. Details on the similarities and differences between these two works are discussed in [47].

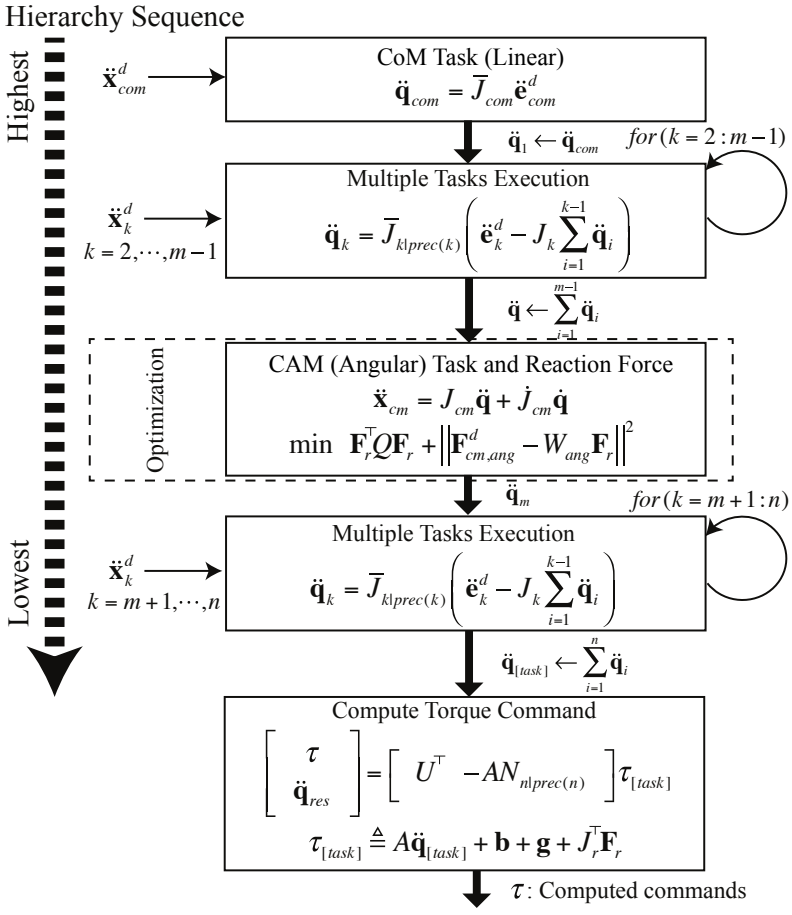


Figure 2.7: **Block Diagram of Whole-Body Locomotion Controller.** In WBLC, motion commands are compounded as joint accelerations based on null-space projection methods. The CM task specification is used to compute reaction forces via QP optimization including unilateral contacts and friction cone constraints. The computed joint acceleration and reaction forces are used to solve for torque commands, which are the final output of WBLC.

2.3.2 Optimizing Reaction Force of Underactuated Robot

Based on the desired joint acceleration given in (Eq. (2.62)), WBLC finds torque commands via the following equation:

$$\boldsymbol{A}(\mathbf{q})\ddot{\mathbf{q}}^d + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \boldsymbol{J}_r^\top \mathbf{F}_r = \boldsymbol{U}^\top \boldsymbol{\tau}, \tag{2.64}$$

where $\mathbf{q} \in \mathbb{R}^{n+6}$ and $\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{g}(\mathbf{q})$ are the joint space coriolis/centrifugal and gravity terms, respectively. $\mathbf{F}_r$ and $\boldsymbol{J}_r$ represent the reaction forces and the corresponding contact Jacobian. $\boldsymbol{\tau} \in \mathbb{R}^n$ and $\boldsymbol{U}^\top \in \mathbb{R}^{(n+6)\times(n)}$ represent the actuator torque commands and the selection matrix mapping actuated torque to the floating base dynamics. Note that $\ddot{\mathbf{q}}^d$ is chosen as,

$$\ddot{\mathbf{q}}^d = \ddot{\mathbf{q}}_{[task]} + \boldsymbol{N}_{n|prec(n)} \ddot{\mathbf{q}}_{res}, \tag{2.65}$$

with

$$\boldsymbol{N}_{n|prec(n)} = \boldsymbol{N}_{prec(n)} \boldsymbol{N}_{n|n-1} \tag{2.66}$$

where $\ddot{\mathbf{q}}_{res}$ is a residual joint acceleration command.

To find the reaction forces $\mathbf{F}_r$, we specify a CM operational task. A CM task consists of linear and angular momenta portions. The linear part, corresponds to the robot's CoM behavior, $\mathbf{F}_{cm,lin}$, and is typically used for locomotion planning. On the other hand, the angular behavior, the so-called CAM, $\mathbf{F}_{cm,ang}$, is typically set to zero value. Setting the angular task to zero creates conflict with other tasks, such as body rotational tasks. We circumvent this problem by projecting angular behavior as a lower priority task than body rotational tasks as we will soon see. In addition, sometimes it is not possible to simultaneously fulfill linear and angular momentum specifications. For that reason, we specific CoM behavior as a hard constraint while relaxing angular behavior, i.e.

$$\begin{aligned} \min_{\mathbf{F}_r} \quad & \mathbf{F}_r^\top \boldsymbol{Q} \mathbf{F}_r + \| \mathbf{F}_{cm,ang}^d - \boldsymbol{W}_{ang} \mathbf{F}_r \|^2 \\ \text{Subject to.} \quad & \mu |\mathbf{F}_{r,z}| \geq |\mathbf{F}_{r,x}| \\ & \mu |\mathbf{F}_{r,z}| \geq |\mathbf{F}_{r,y}| \\ & \mathbf{F}_{cm,lin}^d - \boldsymbol{W}_{lin} \mathbf{F}_r = \mathbf{0} \end{aligned} \tag{2.67}$$

where $\mathbf{F}_{cm,lin}^d$ and $\mathbf{F}_{cm,ang}^d$ are the desired linear and angular parts of the CM, μ represents a friction coefficient related to the contact surfaces, $\boldsymbol{Q}$ is a weighting matrix, and $\boldsymbol{W}_{ang}$ and $\boldsymbol{W}_{lin}$ are mappings from reaction forces to angular and linear momenta behaviors. Based on the results of this optimization, $\mathbf{F}_r$, the desired value of the CAM task can be calculated as follows:

$$\boldsymbol{I}_{cm} \ddot{\mathbf{x}}_{cm}^d = \mathbf{F}_{cm}^d = \begin{bmatrix} \mathbf{F}_{cm,lin}^d & \boldsymbol{W}_{ang} \mathbf{F}_r \end{bmatrix}^\top, \tag{2.68}$$

where $\boldsymbol{I}_{cm}$ is a spatial inertial term. Notice that the term $\boldsymbol{W}_{ang}\mathbf{F}_r$ might be different than $\mathbf{F}^d_{cm,ang}$ since the desired angular behavior might violate friction cone constraints. From the above equation, we extract the desired CM acceleration command $\ddot{\mathbf{x}}^d_{cm}$ for usage in the controller hierarchy. More concretely, $\ddot{\mathbf{x}}^d_{cm} = \left(\ddot{\mathbf{x}}^d_{CoM}\ \boldsymbol{\alpha}^d_{ang}\right)$, where the first term within the parenthesis is the desired CoM acceleration and the second term is the desired angular acceleration. Both of these commands are used separately in the hierarchy defined in Eq. (2.62), to produce the joint acceleration command $\ddot{\mathbf{q}}_{[task]}$ which in turn yields $\ddot{\mathbf{q}}^d$ via Eq. (2.65). Plugging this last term into Eq. (2.64) we obtain

$$\boldsymbol{A}\left(\ddot{\mathbf{q}}_{[task]} + \boldsymbol{N}_{n|prec(n)}\ddot{\mathbf{q}}_{res}\right) + \mathbf{b} + \mathbf{g} + \boldsymbol{J}_r^\top \mathbf{F}_r = \boldsymbol{U}^\top \boldsymbol{\tau}, \tag{2.69}$$

which can be written in matrix form as

$$\begin{bmatrix} \boldsymbol{U}^\top & -\boldsymbol{A}\boldsymbol{N}_{n|prec(n)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau} \\ \ddot{\mathbf{q}}_{res} \end{bmatrix} = \boldsymbol{\tau}_{[task]}, \tag{2.70}$$

where we have defined the term

$$\boldsymbol{\tau}_{[task]} \triangleq \boldsymbol{A}\ddot{\mathbf{q}}_{[task]} + \mathbf{b} + \mathbf{g} + \boldsymbol{J}_r^\top \mathbf{F}_r. \tag{2.71}$$

We now have an underdetermined matrix system which can be solved via pseudo inversion as

$$\begin{bmatrix} \boldsymbol{\tau} \\ \ddot{\mathbf{q}}_{res} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}^\top & -\boldsymbol{A}\boldsymbol{N}_{n|prec(n)} \end{bmatrix}^+ \boldsymbol{\tau}_{[task]} \tag{2.72}$$

where $(.)^+$ represents the Moore-Penrose pseudo inverse operation. Finally, the obtained command torque can control tasks related to locomotion task such as the body orientation with hierarchy as shown in Figure 2.8.

2.3.3 Summary

The newly developed WBLC takes into account realistic contact and friction cone constraints. At the same time, WBLC maintains task priorities using projection operators which is a feature missing in previous QP based WBCs. Overall, WBLC simultaneously exploits the benefits of QP based WBC's and projection based WBC's, achieving versatility and computational efficiency. However, we made several mistakes in the formulations, which are addressed in the latest WBC presented in Chapter. 3. We explain two issues in WBLC:

1. **Contact Constraints Violation.** The first-priority task must address contact constraints to prohibit the other tasks to produce accelerations at the contact points. In the proposed formulation, WBLC has the CoM task as the first one, but this choice could potentially violates contact constraints.

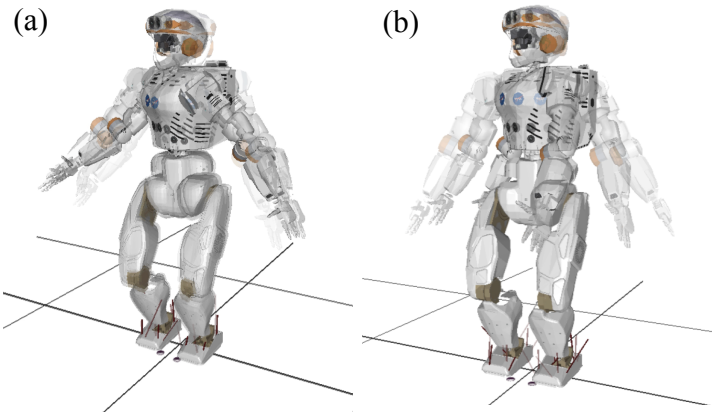


Figure 2.8: **Body orientation test.** (a) CAM task is above the joint position task. (b) CAM task is below the joint position task

2. **Non-zero Virtual Torque.** The last step of WBLC is multiplying a pseudoinverse of the augmented matrix representing under-actuation and residual joint accelerations (Eq. (2.72)). This process can result in non-zero virtual torques since the pseudoinverse finds the nearest solution rather than an exact one.

Chapter 3

Whole-Body Dynamic Control

We devise a new type of WBC which we call whole-body dynamic controller (WBDC) that focuses on speed, unilateral contact constraints, and prioritized task control. The proposed WBC algorithm leverages the efficient computation of projection-based hierarchical task controllers [85] and at the same time incorporates contact inequality constraints, and torque limits which are handled via quadratic programming (QP). The new, projection-based recursive structure incorporates unilateral contact and friction constraints, yielding the desired reduction in computational cost compared to other QP-based algorithms. In addition to the computational savings that result from combining QP- and projection-based methods, we also make important improvements to the computational efficiency of the projection-based operations themselves.

Indeed, in conventional projection-based methods, the computational cost of some operations is considerably high. For instance, one well-known WBC algorithm that uses joint acceleration, [91], includes costly terms such as the time derivative of a null space projection matrix. In addition, many WBC algorithms require the computation of the Coriolis/centrifugal and gravitational forces projected into operational space [57, 86], which are costly to calculate, especially as the number of control tasks increases. Our WBDC eliminates these problems. An analytic solution for the time derivative of the Jacobian is devised by employing Lie group operators, and we provide an implementation using the Rigid Body Dynamic Library [47]. We also eliminate the need to compute Coriolis/centrifugal terms for each task priority.

This chapter contains material from the following publication:
[44] Donghyun Kim, Junhyeok Ahn, Jaemin Lee, Orion Campbel, and Luis Sentis. Whole- Body Control Incorporating Slip, Inequality Constraints, and Task Hierarchy. Submitted in ICRA. IEEE, 2018.
My role covers algorithm development, programming, and the major portion of writing.

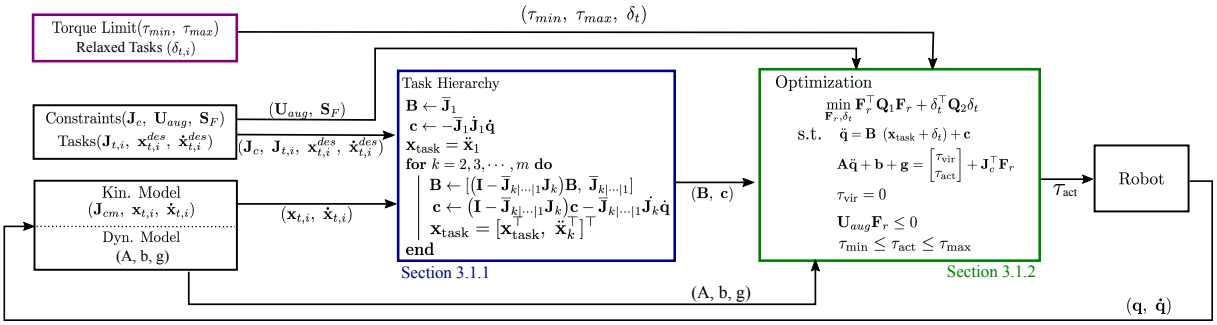


Figure 3.1: **Whole-body dynamic control process flow.** Three main components are described in each section. In every control loop, the dynamic and kinematic model of a system is updated with the sensed configuration ($\mathbf{q}$, $\dot{\mathbf{q}}$). Then, WBDC builds $\boldsymbol{B}$ and $\mathbf{c}$, representing task hierarchy (Section 3.1.1). With $\boldsymbol{B}$ and $\mathbf{c}$, the QP solver computes reaction forces and relaxed task inputs (Section 3.1.2). In the QP, torque limits and task relaxations play an important role in demonstrating various behaviors such as impedance change.

3.1 Formulation of WBDC

We devise a new whole-body control algorithm, dubbed WBDC, that specifies tasks using a hierarchy of accelerations and uses quadratic programming to determine contact forces and relaxed task inputs. Fig. 3.1 illustrates the overall process for computing torque commands, and the details of this process will be explained in the following subsections.

3.1.1 Acceleration Based Iterative Formula with Task Hierarchy

WBDC manages prioritized tasks in a way that is similar to our previous WBLC in Section 2.3. Task-level controllers are computed in operational space as acceleration commands and converted to joint accelerations using differential forward kinematics.

$$\begin{aligned} \dot{\mathbf{x}}_1 &= \boldsymbol{J}_1\dot{\mathbf{q}}, \\ \ddot{\mathbf{x}}_1 &= \boldsymbol{J}_1\ddot{\mathbf{q}} + \dot{\boldsymbol{J}}_1\dot{\mathbf{q}}, \end{aligned} \tag{3.1}$$

where $\mathbf{x}_1 \in \mathbb{R}^n$ and $\mathbf{q} \in \mathbb{R}^m$ represent the first task's operational coordinates and the joint coordinates, respectively, and $\boldsymbol{J}_1$ is the corresponding Jacobian matrix mapping joint coordinates to the task space. $\ddot{\mathbf{x}}_1$ is a task command computed with the desired motion using proportional and derivative (PD) feedback control for a unit mass,

$$\ddot{\mathbf{x}} = \ddot{\mathbf{x}}_{ff} + k_p(\mathbf{x}^d - \mathbf{x}) + k_d(\dot{\mathbf{x}}^d - \dot{\mathbf{x}}). \tag{3.2}$$

$\ddot{\mathbf{x}}_{ff}$, $\mathbf{x}^d$, and $\dot{\mathbf{x}}^d$ are acceleration, position, and velocity, respectively, obtained from the desired motion trajectories. Eq. (3.2) is PD control but integral control could be added if needed. The joint acceleration for a desired task acceleration, $\ddot{\mathbf{x}}_1^d$ can be calculated as

$$\ddot{\mathbf{q}}_1 = \overline{\boldsymbol{J}}_1 \left(\ddot{\mathbf{x}}_1^d - \dot{\boldsymbol{J}}_1 \dot{\mathbf{q}} \right), \tag{3.3}$$

where $\overline{\boldsymbol{J}}_1$ indicates the dynamically consistent inverse of $\boldsymbol{J}_1$, i.e.

$$\overline{\boldsymbol{J}}_1 = \boldsymbol{A}^{-1} \boldsymbol{J}_1^\top \left(\boldsymbol{J}_1 \boldsymbol{A}^{-1} \boldsymbol{J}_1^\top \right)^{-1}, \tag{3.4}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times m}$ indicates the mass/inertia matrix of the rigid body model of the robot. When the second task $\ddot{\mathbf{x}}_2$ is considered with lower priority, the joint space acceleration $\ddot{\mathbf{q}}$ is defined as

$$\ddot{\mathbf{q}}_2 = \ddot{\mathbf{q}}_1 + \overline{\boldsymbol{J}}_{2|1} (\ddot{\mathbf{x}}_2 - \dot{\boldsymbol{J}}_2 \dot{\mathbf{q}} - \boldsymbol{J}_2 \ddot{\mathbf{q}}_1) + \boldsymbol{N}_2 \ddot{\mathbf{q}}', \tag{3.5}$$

where

$$\begin{aligned} \boldsymbol{N}_1 &= \boldsymbol{I} - \overline{\boldsymbol{J}}_1 \boldsymbol{J}, \\ \boldsymbol{J}_{2|1} &= \boldsymbol{J}_2 \boldsymbol{N}_1, \\ \boldsymbol{N}_2 &= \boldsymbol{N}_1 (\boldsymbol{I} - \overline{\boldsymbol{J}}_{2|1} \boldsymbol{J}_{2|1}) = \boldsymbol{N}_1 \boldsymbol{N}_{2|1}. \end{aligned} \tag{3.6}$$

With the assumption that $\boldsymbol{N}_2 = 0$ and $\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_2$, which means that there are no redundant DoF, Eq. (3.5) can be expanded as

$$\ddot{\mathbf{q}} = (\overline{\boldsymbol{J}}_1 - \overline{\boldsymbol{J}}_{2|1} \boldsymbol{J}_2 \overline{\boldsymbol{J}}_1) \ddot{\mathbf{x}}_1 + \overline{\boldsymbol{J}}_{2|1} \ddot{\mathbf{x}}_2 + (-\overline{\boldsymbol{J}}_1 \dot{\boldsymbol{J}}_1 - \overline{\boldsymbol{J}}_{2|1} \dot{\boldsymbol{J}}_2 + \overline{\boldsymbol{J}}_{2|1} \boldsymbol{J}_2 \overline{\boldsymbol{J}}_1 \dot{\boldsymbol{J}}_1) \dot{\mathbf{q}}, \tag{3.7}$$

which can be simplified into matrix form as:

$$\ddot{\mathbf{q}} = \boldsymbol{B}\ \ddot{\mathbf{x}}_{\text{task}} + \mathbf{c}, \tag{3.8}$$

where

$$\begin{aligned} \ddot{\mathbf{x}}_{\text{task}} &= \begin{bmatrix} \ddot{\mathbf{x}}_1 & \ddot{\mathbf{x}}_2 \end{bmatrix}^\top, \\ \boldsymbol{B} &= \begin{bmatrix} (\overline{\boldsymbol{J}}_1 - \overline{\boldsymbol{J}}_{2|1} \boldsymbol{J}_2 \overline{\boldsymbol{J}}_1) & (\overline{\boldsymbol{J}}_{2|1}) \end{bmatrix}, \\ \mathbf{c} &= (-\overline{\boldsymbol{J}}_1 \dot{\boldsymbol{J}}_1 - \overline{\boldsymbol{J}}_{2|1} \dot{\boldsymbol{J}}_2 + \overline{\boldsymbol{J}}_{2|1} \boldsymbol{J}_2 \overline{\boldsymbol{J}}_1 \dot{\boldsymbol{J}}_1) \dot{\mathbf{q}}. \end{aligned} \tag{3.9}$$

In the case of additional tasks ($\ddot{\mathbf{x}}_1,\ \ddot{\mathbf{x}}_2,\ \cdots\ \ddot{\mathbf{x}}_m$), Eq. (3.9) can be iteratively applied for each task and simplified in the matrix form of Eq. (3.8). Algorithm (1) shows the core procedure for

calculating $\boldsymbol{B}$ and $\mathbf{c}$, where

$$\begin{aligned}
\boldsymbol{J}_{k|prec(k)} &= \boldsymbol{J}_k \boldsymbol{N}_{prec(k)}, \\
\boldsymbol{N}_{prec(k)} &= \prod_{s=1}^{k-1} \boldsymbol{N}_{s|s-1} \quad (k \geq 2, \quad \boldsymbol{N}_{1|0} = \boldsymbol{N}_1), \\
\boldsymbol{N}_{s|s-1} &= \boldsymbol{I} - \overline{\boldsymbol{J}}_{s|prec(s)} \boldsymbol{J}_{s|prec(s)} \quad (s \geq 2).
\end{aligned} \tag{3.10}$$

Algorithm 1: Iterative Calculation of $\boldsymbol{B}$ and $\mathbf{c}$ for Multiple Hierarchal Tasks

$\boldsymbol{B} \leftarrow \overline{\boldsymbol{J}}_1$
$\mathbf{c} \leftarrow -\overline{\boldsymbol{J}}_1 \dot{\boldsymbol{J}}_1 \dot{\mathbf{q}}$
$\ddot{\mathbf{x}}_{\text{task}} = \ddot{\mathbf{x}}_1$
for $k = 2, 3 \cdots m$ **do**
  $\boldsymbol{B} \leftarrow [\big(\mathbf{I} - \overline{\boldsymbol{J}}_{k|\cdots|1} \boldsymbol{J}_k\big)\boldsymbol{B},\ \overline{\boldsymbol{J}}_{k|\cdots|1}]$
  $\mathbf{c} \leftarrow \big(\mathbf{I} - \overline{\boldsymbol{J}}_{k|\cdots|1} \boldsymbol{J}_k\big)\mathbf{c} - \overline{\boldsymbol{J}}_{k|\cdots|1} \dot{\boldsymbol{J}}_k \dot{\mathbf{q}}$
  $\ddot{\mathbf{x}}_{\text{task}} = [\ddot{\mathbf{x}}_{\text{task}}^\top,\ \ddot{\mathbf{x}}_k^\top]^\top$
end
return $\boldsymbol{B},\ \mathbf{c}$

Note that if contacts exist, the top priority tasks must represent the contact constraints since every other task needs to recognize that there is no acceleration at the contact points based on the no slip assumption. WBDC can handle slip conditions and the method is described in Section 3.2.

3.1.2 Quadratic Programming With Inequality Constraints

Based on the fact that joint acceleration changes linearly with task space acceleration (Eq. (3.8)), we formulate a QP problem to compute reaction forces, the task relaxation vector, and the torques of virtual joints. The solution of the QP is used to compute final torque commands to

be sent to each joint. The complete QP used in the WBDC is described below.

$$
\begin{aligned}
&\min_{\mathbf{F}_r, \boldsymbol{\delta}_\mathrm{t}} \quad \mathbf{F}_r^\top \boldsymbol{Q}_1 \mathbf{F}_r + \boldsymbol{\delta}_\mathrm{t}^\top \boldsymbol{Q}_2 \boldsymbol{\delta}_\mathrm{t} \\
\text{s.t.} \quad &\ddot{\mathbf{q}} = \boldsymbol{B}\ (\mathbf{x}_\mathrm{task} + \boldsymbol{\delta}_t) + \mathbf{c} && \text{(task hierarchy)} \\
&\boldsymbol{A}\ddot{\mathbf{q}} + \mathbf{b} + \mathbf{g} = \begin{bmatrix} \boldsymbol{\tau}_\mathrm{vir} \\ \boldsymbol{\tau}_\mathrm{act} \end{bmatrix} + \boldsymbol{J}_c^\top \mathbf{F}_r && \text{(multi-body dyn.)} \\
&\boldsymbol{\tau}_\mathrm{vir} = \mathbf{0} && \text{(floating base)} \\
&\boldsymbol{U}_{aug} \mathbf{F}_r \leq \mathbf{0} && \text{(contact)} \\
&\boldsymbol{\tau}_\mathrm{min} \leq \boldsymbol{\tau}_\mathrm{act} \leq \boldsymbol{\tau}_\mathrm{max} && \text{(torque limit)}
\end{aligned}
$$

The decision variable $\mathbf{F}_r$ is an augmented vector of contact wrenches at every contact point (e.g. $\mathbf{F}_r \in \mathbb{R}^{6k}$ for k contact points). $\boldsymbol{\tau}_\mathrm{vir}$ represents virtual forces on the floating base, which should be zero. $\boldsymbol{\delta}_t$ is a vector stacking the task relaxation variables, $\delta_{t,i}$, at designated locations. For example, when the dimension of total tasks is five and we want the second and the third tasks to be open for possible adjustment, the vector should be

$$
\boldsymbol{\delta}_t = \begin{bmatrix} 0 & \delta_{t,1} & \delta_{t,2} & 0 & 0 \end{bmatrix}^\top . \tag{3.11}
$$

$\boldsymbol{\delta}_t$ implies that we can adjust some of task commands if they are not feasible with the given inequality and equality constraints. For example, robots can be compliant to an external disturbance if they do not have enough power to sustain the commanded position. As we can imagine, this potential task command violation (or adjustment) provides versatility for our algorithm.

Note that there is no CM task in the QP because '$\boldsymbol{\tau}_\mathrm{vir} = 0$' implies '$\boldsymbol{I}_{cm}\ddot{\mathbf{x}}_{cm} + \mathbf{g}_{cm} = \mathbf{F}_{cm} (= \boldsymbol{W}\mathbf{F}_r)$' of Eq. (2.68). The following is a simple proof for it:

Proof. To proof the equivalence of CM constraint and zero-virtual-torque constraint, we start from the following multi-body dynamics,

$$
\boldsymbol{A}\ddot{\mathbf{q}} + \mathbf{b} + \mathbf{g} = \boldsymbol{\tau} + \boldsymbol{J}_c^\top \mathbf{F}_r. \tag{3.12}
$$

By multiplying $\boldsymbol{I}_{cm}\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}$ on the both sides, we could obtain

$$
\boldsymbol{I}_{cm}(\boldsymbol{J}_{cm}\ddot{\mathbf{q}} + \boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\mathbf{b}) + \boldsymbol{I}_{cm}\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\mathbf{g} = \boldsymbol{I}_{cm}\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}(\boldsymbol{\tau} + \boldsymbol{J}_c^\top \mathbf{F}_r). \tag{3.13}
$$

From Eq. (2.57), we can find

$$
\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\mathbf{b} = \dot{\boldsymbol{J}}_{cm}\dot{\mathbf{q}}, \tag{3.14}
$$

and

$$\boldsymbol{I}_{cm}\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\mathbf{g} = \mathbf{g}_{cm} \tag{3.15}$$

is obtained from Eq. (2.56). The principle of virtual work give us

$$\boldsymbol{\tau} + \boldsymbol{J}_c^{\top}\mathbf{F}_r = \boldsymbol{J}_{cm}^{\top}\mathbf{F}_{cm}. \tag{3.16}$$

Eq. (3.13) is calculated as

$$\boldsymbol{I}_{cm}(\boldsymbol{J}_{cm}\ddot{\mathbf{q}} + \dot{\boldsymbol{J}}_{cm}\dot{\mathbf{q}}) + \mathbf{g}_{cm} = \boldsymbol{I}_{cm}(\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\boldsymbol{J}_{cm}^{\top})\mathbf{F}_{cm}. \tag{3.17}$$

Since $\ddot{\mathbf{x}}_{cm} = \boldsymbol{J}_{cm}\ddot{\mathbf{q}} + \dot{\boldsymbol{J}}_{cm}\dot{\mathbf{q}}$ and $\boldsymbol{I}_{cm} = (\boldsymbol{J}_{cm}\boldsymbol{A}^{-1}\boldsymbol{J}_{cm}^{\top})^{-1}$,

$$\boldsymbol{I}_{cm}\ddot{\mathbf{x}}_{cm} + \mathbf{g}_{cm} = \mathbf{F}_{cm}, \tag{3.18}$$

which is the CM constraint. □

Most constraints are self-explanatory, but $\boldsymbol{B}$ and $\mathbf{c}$ are the iteratively calculated matrix and vector from Section 3.1.1 representing the task hierarchy. $\boldsymbol{A}$, $\mathbf{b}$, $\mathbf{g}$ are the inertia matrix, centrifugal, and gravitational force respectively, and $\boldsymbol{J}_c$ is a Jacobian matrix mapping joint space to contact constraint space. The contact constraint contains unilateral inequalities Eq. (3.20) on the normal force to prevent flipping/rolling the contacting link and the friction cone inequalities Eq. (3.19) to prevent slipping. In the case of a surface contact with a rectangular support area, the inequalities are represented as

$$\begin{aligned} \mid f_x \mid &\leq \mu f_z, \\ \mid f_y \mid &\leq \mu f_z, \end{aligned} \tag{3.19}$$

$$\begin{aligned} f_z &> 0, \\ \mid \tau_x \mid &\leq d_y f_z, \\ \mid \tau_y \mid &\leq d_x f_z, \\ \tau_{z,\min} \leq \tau_z &\leq \tau_{z,\max}, \end{aligned} \tag{3.20}$$

where

$$\begin{aligned} \tau_{z,\min} &\triangleq -\mu(d_x + d_y)f_z + \mid d_y f_x - \mu\tau_x \mid + \mid d_x f_y - \mu\tau_y \mid, \\ \tau_{z,\max} &\triangleq \mu(d_x + d_y)f_z - \mid d_y f_x + \mu\tau_x \mid - \mid d_x f_y + \mu\tau_y \mid, \end{aligned} \tag{3.21}$$

and $d_x,\ d_y$ represent the distance from the center of the rectangle to the vertex in the local contact frame. Note that, $[\tau_x,\ \tau_y,\ \tau_z,\ f_x,\ f_y,\ f_z]^T$ is the wrench representing resultant forces

on the surface taken at the center of the rectangle in the local coordinate frame. [12] describes the closed-form formula for the contact wrench cone $\mathbf{U} \in \mathbb{R}^{\mathbf{16\times 6}}$ defined by

$$\boldsymbol{U} \triangleq \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & -\mu \\ 0 & 0 & 0 & 1 & 0 & -\mu \\ 0 & 0 & 0 & 0 & -1 & -\mu \\ 0 & 0 & 0 & 0 & 1 & -\mu \\ -1 & 0 & 0 & 0 & 0 & -d_y \\ 1 & 0 & 0 & 0 & 0 & -d_y \\ 0 & -1 & 0 & 0 & 0 & -d_x \\ 0 & 1 & 0 & 0 & 0 & -d_x \\ \mu & \mu & -1 & -d_y & -d_x & -(d_x+d_y)\mu \\ \mu & -\mu & -1 & -d_y & d_x & -(d_x+d_y)\mu \\ -\mu & \mu & -1 & d_y & -d_x & -(d_x+d_y)\mu \\ -\mu & -\mu & -1 & d_y & d_x & -(d_x+d_y)\mu \\ \mu & \mu & 1 & d_y & d_x & -(d_x+d_y)\mu \\ \mu & -\mu & 1 & d_y & -d_x & -(d_x+d_y)\mu \\ -\mu & \mu & 1 & -d_y & d_x & -(d_x+d_y)\mu \\ -\mu & -\mu & 1 & -d_y & -d_x & -(d_x+d_y)\mu \end{bmatrix}. \tag{3.22}$$

Then the contact constraints are concisely expressed by

$$\mathbf{U}\mathbf{F_r} \leq \mathbf{0}. \tag{3.23}$$

The final step is to include contact frame rotations and augment multiple contact points. For example, the formulation for k contact points is

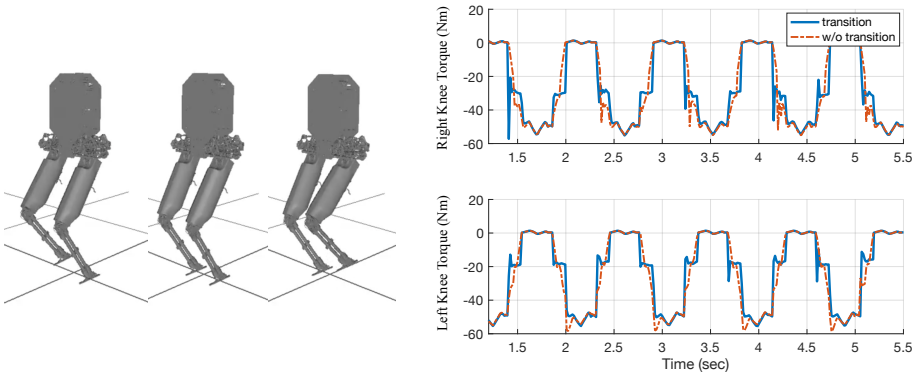
$$\mathbf{U_{aug}}\mathbf{F_r} = \begin{bmatrix} \mathbf{U_1}\mathbf{R_{aug,1}} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{U_k}\mathbf{R_{aug,k}} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{r,1} \\ \vdots \\ \mathbf{F}_{r,k} \end{bmatrix} \leq \mathbf{0}, \tag{3.24}$$

where

$$\mathbf{R_{aug,i}} = \begin{bmatrix} \boldsymbol{R}_i^{cm} & \mathbf{0}^{3\times 3} \\ \mathbf{0}^{3\times 3} & \boldsymbol{R}_i^{cm} \end{bmatrix}. \tag{3.25}$$

Note that $\boldsymbol{R}_i^{cm}$ is SO(3) from the local contact coordinate frame to the CoM frame and is multiplied to operate with the reaction forces represented in the CoM frame.

When selecting the weight matrices $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$, we recommend using heavier weight on $\boldsymbol{Q}_2$. For example, if we relax the height control task, then the QP will adjust the task to allow the robot to fall below the commanded height to reduce the vertical directional reaction forces because these forces are generally the largest numbers among the optimization variables.



(a) Stepping Test Simulation (b) Knee Joint Torques

Figure 3.2: **Stepping Motion With and Without Contact Transitions.** Simulation results show that knee joint torques become seriously discontinuous w/o contact transition process.

The last step of WBDC is to compute a torque command from the optimization results. The products of QP are $\mathbf{F}_r$ and $\boldsymbol{\delta}_t$, and we obtain $\boldsymbol{\tau}_{\text{act}}$ by plugging $\mathbf{F}_r$ and $\boldsymbol{\delta}_t$ into the equation,

$$\boldsymbol{\tau}_{\text{total}} = \boldsymbol{A}\left(\boldsymbol{B}(\mathbf{x}_{\text{task}} + \boldsymbol{\delta}_t) + \mathbf{c}\right) + \mathbf{b} + \mathbf{g} - \boldsymbol{J}_c^{\top}\mathbf{F}_r. \tag{3.26}$$

Then, we take the actuated joints of $\boldsymbol{\tau}_{\text{total}}$ and send the torque command to a robot and do not use $\boldsymbol{\tau}_{\text{vir}}$.

3.1.3 Contact Switching Transition of WBDC

In Section 2.2.1, we introduced a contact switching method for the WBCs using projection-based contact constraints. In WBDC formulation, we can constrain reaction forces directly by adding upper bounds for reaction forces. By increasing or decreasing the bounds, QP solvers find properly truncated reaction forces. The simulation results of this transition method are presented in Fig. 3.2.

3.2 Incorporating Slip into WBDC

When the reaction forces of legged systems are inside of the friction cones, the contacts have zero acceleration relative to the contact surface. This is generally a basic assumption of WBCs called a no-slip condition. However, WBDC can account for the slip condition as well by utilizing mixed-integer programming. The slip can be interpreted as the case where there is a non-zero relative acceleration at the contact points. Therefore, the action we need to take for the slip

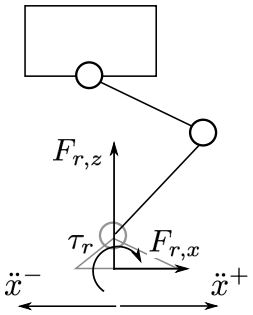


Figure 3.3: **2D Single Foot Model.** This is a simple 2D model for the description of slip condition handling.

condition handling is to open the contact constraint task for adjustment by adding relaxation variables. At the same time, we need to switch the constraints for the horizontal reaction forces from inequalities ($|F_{x,y}| \leq \mu F_z$) to equalities ($|F_{x,y}| = \mu F_z$) when robots slide.

The slip conditions are described by several if-then statements. Here, we explain the process with an example of a simple two-dimensional robot. The nomenclatures of the following equations are identical to ones in Fig. 3.3.

1. If $\ddot{x}^+ = 0$ and $\ddot{x}^- = 0$, then

$$\begin{aligned} F_{r,x} - \mu F_{r,z} \leq 0 \\ 0 \leq F_{r,x} + \mu F_{r,z} \end{aligned} \tag{3.27}$$

2. If $\ddot{x}^+ > 0$, then $F_{r,x} + \mu F_{r,z} = 0$.

3. If $\ddot{x}^- > 0$, then $F_{r,x} - \mu F_{r,z} = 0$.

These statements are formulated in mixed-interger programming as following.

$$\begin{aligned} \ddot{x}^+ - \epsilon \leq \eta_1(\ddot{x}^+_{\max} - \epsilon), \\ \ddot{x}^+ - \epsilon \geq (1-\eta_1)(-\epsilon), \\ 0 \leq F_{r,x} + \mu F_{r,z} \leq (1-\eta_1)M. \end{aligned} \tag{3.28}$$

And the opposite directional sliding is described by

$$\begin{aligned} \ddot{x}^- - \epsilon \leq \eta_2(\ddot{x}^-_{\max} - \epsilon), \\ \ddot{x}^- - \epsilon \geq (1-\eta_2)(-\epsilon), \\ (1-\eta_2)m \leq F_{r,x} - \mu F_{r,z} \leq 0. \end{aligned} \tag{3.29}$$

M and m are a large and small number, respectively, and η_i is integer. The equations say that η_i becomes 1 if $\ddot{x}^{\cdot}$ is larger than ϵ and the constraints for reaction forces will be eqaulities.

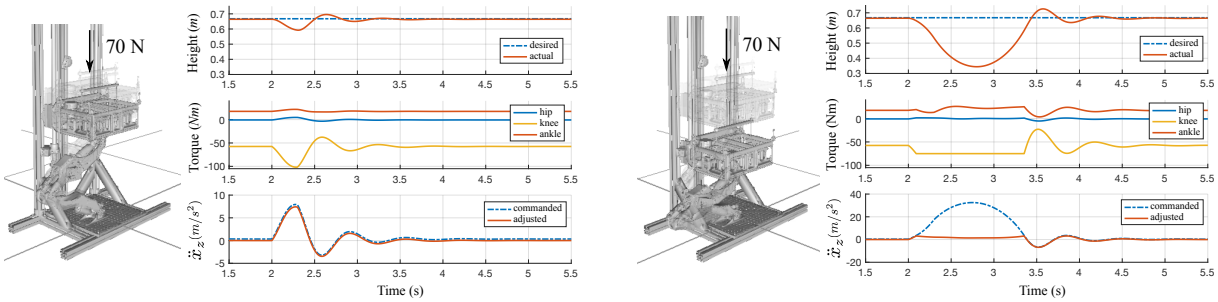
3.3 Numerically Robust WBDC

Standard WBDC in Section 3.1 can fail to find a solution if given task commands and relaxations of the tasks are not correctly set. For example, given contact constraints, if users command unachievable tasks without task relaxations, QP fails to find reaction forces. We found a circumvent on this issue. We merely remove the constraints $\boldsymbol{\tau}_{\text{vir}} = \mathbf{0}$ and add the variable to the cost, which is equivalent to the relaxation of under-actuation constraints.

$$
\begin{aligned}
\min_{\mathbf{F}_r, \boldsymbol{\delta}_{\text{t}}, \boldsymbol{\tau}_{\text{vir}}} \quad & \mathbf{F}_r^{\top} \boldsymbol{Q}_1 \mathbf{F}_r + \boldsymbol{\delta}_{\text{t}}^{\top} \boldsymbol{Q}_2 \boldsymbol{\delta}_{\text{t}} + \boldsymbol{\tau}_{\text{vir}}^{\top} \boldsymbol{Q}_3 \boldsymbol{\tau}_{\text{vir}} & \\
\text{s.t.} \quad & \ddot{\mathbf{q}} = \boldsymbol{B} \ (\mathbf{x}_{\text{task}} + \boldsymbol{\delta}_t) + \mathbf{c} & \text{(task hierarchy)} \\
& \boldsymbol{A}\ddot{\mathbf{q}} + \mathbf{b} + \mathbf{g} = \begin{bmatrix} \boldsymbol{\tau}_{\text{vir}} \\ \boldsymbol{\tau}_{\text{act}} \end{bmatrix} + \boldsymbol{J}_c^{\top} \mathbf{F}_r & \text{(multi-body dyn.)} \\
& \boldsymbol{I}_{cm} \left(\boldsymbol{J}_{cm} \ddot{\mathbf{q}} - \dot{\boldsymbol{J}}_{cm} \dot{\mathbf{q}} \right) + \mathbf{g}_{cm} = \boldsymbol{S}_{\mathbf{F}} \mathbf{F}_r & \text{(centroidal dyn.)} \\
& \boldsymbol{U}_{aug} \mathbf{F}_r \leq 0 & \text{(contact)} \\
& \boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau}_{\text{act}} \leq \boldsymbol{\tau}_{\max} & \text{(torque limit)}
\end{aligned}
$$

We re-introduce CM dynamics here. This method has the potential to violate dynamics but is exceptionally robust. One interesting aspect is that this robust WBDC can account for the slip condition without special addition or adjustment. The only thing we need to do is to open the contact constraint task by adding relaxation variables. Unlike the previous formulations in Section 3.2, we do not need to utilize mixed-integer programming to see sliding.

To verify the proposed algorithm, we show physics-based simulations of two robots, Draco P1 and Valkyrie. Draco P1 is a single leg testbed having three viscoelastic liquid cooled actuators. For those who want to know more details about the Draco robot and its new type of actuator, we refer our recent paper [45]. Another robot is Valkyrie, an adult size humanoid robot developed by NASA Johnson Space Center [77]. The simulations are designed to verify our claims: 1) the adjustment of the task commands based on torque limits, 2) stable control while sliding on slippery ground due to task adjustment, 3) execution of prioritized tasks, and 4) computational efficiency.



(a) Torque limit: 175 Nm

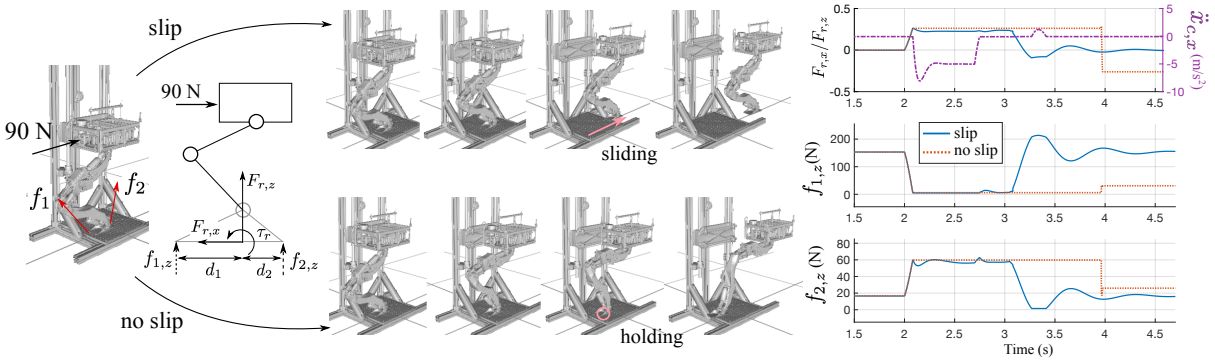
(b) Torque limit: 75 Nm

Figure 3.4: **Maintaining a body height with different torque limits.** For the same external disturbance, the robot shows different impedance behavior with the same feedback gains. We can see the knee torque in (b) is truncated around 75N m because of the torque limit, and this results in the error at commanded task acceleration ($\ddot{x}_z$). As a result, we can see larger tracking error in the height control.

3.3.1 Simulation Result - Draco P1: Single Leg Testbed

The supporting structure connected to Draco P1's body constrains the hip motion to the horizontal and vertical direction; therefore, the robot only moves in the sagittal plane without body rotation. We choose this system because the simple system dynamics can provide better intuition about the WBDC formulation. In the experiment, we push Draco P1 downward and backward while the robot is controlled to maintain the initial body pose. The tasks of the WBDC are set to maintain the contact constraint and the initial body position with hierarchy. Note that the body position is represented with respect to the local frame attached at the foot. The purpose of the experiment with Draco P1 is to show how task relaxation can contribute to the stable posture control of the robot while abiding by the inequality constraints under QP (e.g. torque limits and contact constraints).

In the first experiment, we push down on the robot with a 70N force for 0.3s with two different torque limits. The torque limits are set to 175N m in the first trial and 75N m in the second trial for each actuated joint. The other parameters such as the feedback gains, initial postures, and task setup are all identical in both trials. The vertical direction body movement (z direction in the body position task space) is opened for adjustment, which means that the WBDC can alter the acceleration $\ddot{x}_z$ due to the inequality constraints in the QP described in Section 3.1.2. Indeed, the relaxation is required because the robot might not be able to push as it is supposed to do, due to the torque limits, however it is non-trivial to include torque limits in previous hierarchical-based controllers.



(a) Reaction force diagram (b) Simulation results (c) Data comparison

Figure 3.5: **Body position control on the slippery ground.** (a) shows that we push the body backward with a 90N force during 0.7s. To show the result more intuitively, we compute the vertical directional reaction forces at the toe and heel from the generalized force, which we actually use in WBDC. (b) When we allow x-directional acceleration at the contact point (slip condition), the robot maintains the commanded posture while sliding on the slippery ground. On the other hand, with out relaxation variables for contact task (non-slip condition), the robot generates torque commands to maintain the contact task as much as possible and then eventually loses the contact. (c) shows that $F_{r,x}/F_{r,z}$ is constantly 0.3 in the no-slip case since there is no room to adjust the contact acceleration. However, in the slip case (solid blue lines), WBDC controls the reaction forces by exploiting nonzero acceleration $\ddot{\mathbf{x}}_{c,x}$.

The plot at the top of Fig. 3.4 shows the desired body position task, which is to maintain the initial pose and the actual position of the body. The graph located in the middle shows the torque commands for each joint. In the bottom plot in Fig. 3.4, the blue line shows the commanded acceleration that must be exerted to achieve the task (computed from Eq. (3.2)), and the red line represents the resulting task acceleration that has been adjusted due to the torque limit inequality. In the first trial, in which we push down the robot with the torque limit of 175N m, the leg stiffly reacts to the external force (Fig. 3.4(a)) since it is capable of generating the required torque to execute the commanded accelerations ($\ddot{x}_z$). The data in the last graph in Fig. 3.4(a) shows that blue and red lines are almost identical, which means the desired task acceleration is not large enough to violate the torque limit. However, in the second trial, the robot with 75 N m torque limits in Fig. 3.4(b) shows large differences between the commanded task acceleration and the adjusted acceleration. Specifically, from 2 to 3 s, Draco P1 cannot push up the body with the commanded torque, which leads to the truncation of the knee torque command and a large error in the task space.

In the second experiment, in order to show how relaxation variables can contribute to stability on slippery terrain, we push the robot backward with a 90N force during 0.7s on slippery ground where the friction coefficient is 0.3. The external disturbance is large enough to force the robot to slide to maintain its posture. When we do not relax the zero acceleration constraint at the contact, Draco P1 tries to maintain the contact point and eventually loses its balance (Fig. 3.5). However, when we allow the x direction acceleration to be a non-zero value at the contact point, the robot sustains the body position by allowing the foot to slide on the ground. Note that the actual hip/body box has a limited range of motion, but we remove that limit in the simulation to clearly show the sliding.

Fig. 3.5(c) presents the ratio of the horizontal and vertical elements of the generalized forces $[\tau_r,\ F_{r,x},\ F_{r,z}]^\top$ and the equivalent forces at the toe ($f_{1,z}$) and the heel ($f_{2,z}$). We present the reaction forces at two points instead of a generalized force, which we use in our algorithm, to aid in understanding. The force is computed by the following equations.

$$\begin{aligned} \tau_r &= f_{1,z} d_1 - f_{2,z} d_2 \\ F_{r,z} &= f_{1,z} + f_{2,z} \end{aligned} \tag{3.30}$$

Therefore, vertical reaction forces at toe and heel are

$$f_{1,z} = \frac{F_{r,z} d_1 + \tau_r}{d_1 + d_2}, \quad f_{2,z} = \frac{F_{r,z} d_2 - \tau_r}{d_1 + d_2}. \tag{3.31}$$

The nomenclature in these equations is the same as that shown in Fig. 3.5(a). The data from two different cases are presented in Fig. 3.5(c). The ratio of $F_{r,x}$ to $F_{r,z}$ in the no-slip case is

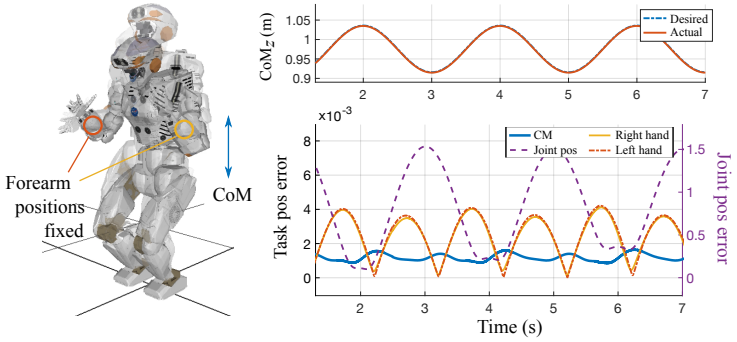


Figure 3.6: **CoM height control while maintaining forearm positions.** With both forearm position fixed, Valkyrie moves up and down as the commanded CoM height changes. The data show that Valkyrie's CoM height is almost identical to the desired trajectory. The norm of each task error becomes larger as the priority of the task becomes lower.

constantly 0.3 until the robot loses the contact and diverges, which means that the robot exerts the maximum horizontal reaction force to keep the contact point. The red dotted lines in the plots of $f_{1,z}$ and $f_{2,z}$ show that the toe reaction force is almost zero and the heel supports the entire body mass rather than carefully modulating them to keep the robot's posture. $\ddot{x}_{c,x}$, shown in purple in the top plot in Fig. 3.5(c) is the horizontal acceleration of the contact point (the foot) in the sliding demonstration. Negative acceleration and the small positive bump explain the backward sliding and a small reactive movement in the forward direction when the sliding is over.

3.3.2 Simulation Result - Valkyrie: NASA's Full Humanoid

We also tested a full humanoid to show task hierarchy and the computational benefits of our method because Draco P1 has only three actuators, which are not enough to demonstrate multiple tasks with hierarchy. In our simulation, Valkyrie has 28 actuated joints, which is fewer than the actual robot, by assuming that hand and wrist joints are fixed. Except for the reduced number of actuated joints, we built the Valkyrie simulation based on data provided by NASA. The total dimensionality of the configuration space is 34 including the 6 virtual joints representing a floating base. The tasks used in the simulation are listed and the subscript indexes indicate the priority of each task.

- $\ddot{\mathbf{x}}_1 \in \mathbb{R}^{12}$: double surface contacts

- $\ddot{\mathbf{x}}_2 \in \mathbb{R}^6$: centroidal momentum
- $\ddot{\mathbf{x}}_3 \in \mathbb{R}^3$: right forearm position
- $\ddot{\mathbf{x}}_4 \in \mathbb{R}^3$: left forearm position
- $\ddot{\mathbf{x}}_5 \in \mathbb{R}^{28}$: full joint posture

In Fig. 3.6, the CoM is commanded to move up and down while maintaining the Cartesian position of both forearms. The task representing contact constraints must have the highest priority, and the task inputs are zero acceleration. The second is the CM task including both angular and linear parts. The inputs of the CM task are minimizing centroidal angular velocity and tracking the planned CoM trajectory. The CoM trajectory used in the test is a z directional sinusoidal path with 0.12m amplitude and 0.5Hz frequency while keeping the initial position in x and y directions. The third and fourth tasks are right forearm and left forearm position control tasks, respectively. The commanded motion is holding the initial positions. The last task is a full joint configuration task, and the command is also keeping the initial joint positions as much as possible.

The results presented in Fig. 3.6 validate that Valkyrie successfully controls the CoM by showing almost identical desired and actual CoM height. To address the effect of hierarchy, we plot the norm of each task's error. The error for the CM and forearm tasks are significantly smaller than the error for joint position control because the joint positions change a lot from the initial posture to execute the other tasks with higher priorities.

We measure the computation time of WBDC using a laptop with a dual-core 3.0 GHz Intel i7 processor while changing task sets as described in Fig. 3.7(a)-(d). In each case, QP takes almost invariably 0.6ms. Building the $\boldsymbol{B}$ matrix and the $\mathbf{c}$ vector requires between $0.7 \sim 0.9$ms of computation time, slightly increasing as the task size increases from case (a) to (d). However, we could significantly reduce this time by optimizing the matrix computations, which has not been done in our current code. Considering the large number of joints in this model and the relatively low computational power of the laptop used here, the presented computation times are enough evidence to demonstrate the computational efficiency of this approach. Moreover, the nearly ignorable time increase resulting from increasing task size is a promising result concerning the computational load.

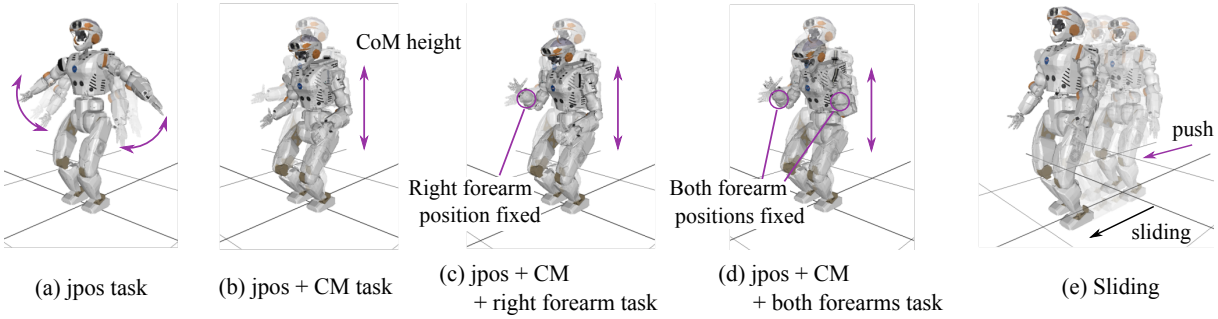


Figure 3.7: **Computation time measurement with various tasksets and sliding demonstration.** (a)-(d) QP consistently spends 0.6ms to finish the computation while the number of tasks increases. (e) Valkyrie successfully controls its the posture while it slides on the slippery ground.

3.4 Summary

We have formulated a new WBC incorporating friction cone constraints, torque limits, slip conditions, and a task hierarchy. The newly developed WBDC takes into account realistic contact and friction cone constraints. At the same time, WBDC maintains task priorities using projection operators which are missing in previous QP based WBCs. Moreover, WBDC allows for potential violations of the no-slip assumption to be more robust to external disturbances. As a result, a robot can stably maintain its balance on a very slippery surface by sliding. Overall, WBDC simultaneously exploits the benefits of QP-based WBCs and projection-based WBCs, achieving both versatility and computational efficiency.

The proposed algorithm enables the investigation of many interesting topics such as locomotion involving sliding or ice skating on slippery terrain. Task command adjustments due to inequality constraints are another attractive functionality of our algorithm. This capability might also make possible the use of a high-level planning algorithm with more generous dynamic constraints. As the WBDC adjusts task commands based on the constraints, robust motions can be generated if the controller is complemented by a re-planning algorithm. We plan to implement WBDC in the actual Draco P1, which is a robot designed to demonstrate an energetic jump with high payloads. Our upcoming study will be about the empirical validation of WBDC, and automatic motion adjustments based on the robots condition (e.g. motion speed reduction based on actuators core temperatures).

Chapter 4

Robust Locomotion Planning

In this chapter, we first review a phase space planner (PSP) method presented in [113], which was developed as a new type of dynamic locomotion planner. The original PSP algorithm utilizes numerical integration and searches for foot switching states, but we found an analytical method to achieve similar capabilities by restricting vertical movement to a set of linear CoM height surfaces. Furnished with this new analytical and computational efficient method, we went on to apply PSP to various applications requiring intensive computational iterations, such as locomotion path planning and machine learning for robust locomotion control. We will shortly explain the results of a kinodynamic RRT planner for locomotion and focus on a reinforcement learning (RL) based planner for robust locomotion. Our RL-based planner significantly improves the robustness of walking by providing fast step decision making from a trained neural network.

Another important method we devised is called velocity reversal planner, which was developed to balance point-foot bipedal robots. This planner replans foot placements by predicting the robot's CoM state and choosing the best next foot locations to achieve a cyclic behavior in a robust manner. The planner has been extensively analyzed both with a simulated robot and with an actual system. In this chapter, we show simulation results while the experimental results will be presented in Chapter 7.

This chapter contains material from the following publications:
[2] Junhyeok Ahn, Orion Campbel, Donghyun Kim, and Luis Sentis. Fast Kinodynamic Bipedal Locomotion Planning with Moving Obstacles. IEEE, 2018.
[47] Donghyun Kim, Jaemin Lee, and Luis Sentis. Robust Dynamic Locomotion via Rein- forcement Learning and Novel Whole Body Controller. arXiv.org, 2017.
[49] Donghyun Kim, Ye Zhao, Gray Thomas, Benito R Fernandez, and Luis Sentis. Stabilizing Series-Elastic Point-Foot Bipeds Using Whole-Body Operational Space Control. IEEE Transactions on Robotics, 32(6):13621379, 2016.
My role covers algorithm development, programming, and the major portion of writing. The development of the sampling based walking path planner has been led by coauthors.

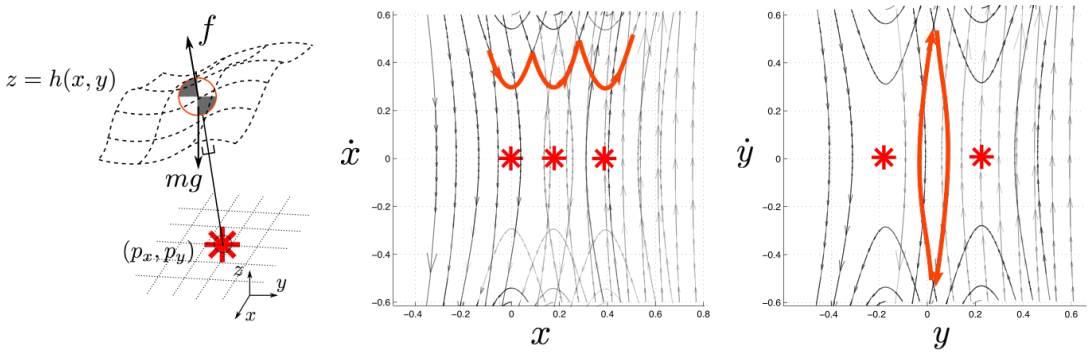


Figure 4.1: **PIP model and CoM phase plots.** The CoM of a PIP model moves on a smooth height surface. Overlapped phase plots show that a CoM phase path in the sagittal plane is a set of connected parabolas. In the coronal plane, the parabolas form a limit cycle.

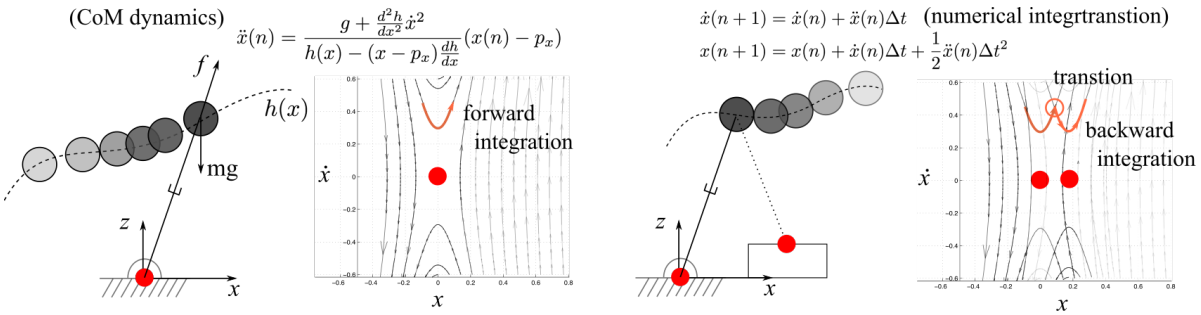
4.1 Phase Space Planner

Phase Space Planning (PSP) is designed based on predictions about the CoM path given Prismatic Inverted Pendulum (PIP) dynamics (Fig. 4.1). The idea is that the CoM path in the sagittal plane is a set of connected parabolas, and in the coronal plane consists of two parabolas forming a closed cycle. We use x, y, and z to represent the sagittal, the lateral, and the vertical directions, respectively. We will call the CoM apex state as the position of the state when the sagittal CoM position equals the sagittal foot support position. Given the CoM's current apex state $[\ x_1,\ y_1,\ \dot{x}_1,\ \dot{y}_1\ ]^\top$ and the next CoM apex state $[\ x_2,\ y_2,\ \dot{x}_2,\ \dot{y}_2\ ]^\top$ those two paths meet together at a single point in the CoM's phase space.

4.1.1 Basic Algorithm

In [113], the parameters are used as control inputs are $p_x,\ \dot{x}_{apex},\ \dot{y}_{apex}$, which correspond to the sagittal foot position, and the sagittal and lateral CoM apex positions. The method consists of searching lateral foot positions in the phase space with given desired sagittal foot positions and sagittal CoM apex velocities. This original PSP method relies on numerical integration thus tolerating the sue of any CoM parametric height surface. Let $h(x)$ represent a predefined center of mass (CoM) height surface parameterized by the CoM horizontal position, x, and let g be the magnitude of the acceleration due to gravity. The following equation holds according to the PIP model,

$$\ddot{x} = \frac{g + \ddot{h}}{h}(x - p_x). \tag{4.1}$$



(a) Progress forward (b) Transition to the next step

Figure 4.2: **PSP Progressive Step.** (a) shows how the CoM state of the PIP model evolves. The phase paths are drawn via numerical integration. (b) explains what the transition state is. Given the next state , PSP integrates backwards and finds the crossing state.

In practice, it is inconvenient to handle the dynamics of h and x separately. Instead, parameterizing the dynamics by one variable (e.g. $h(x)$ instead of $h(t)$ and $x(t)$) is more advantageous when using PIP dynamics. In particular, because the height surface is a function of x, we can write

$$\frac{dh}{dt} = \frac{dh}{dx}\frac{dx}{dt}, \tag{4.2}$$

$$\frac{d^2h}{dt^2} = \frac{d}{dt}\Big(\frac{dh}{dx}\Big)\frac{dx}{dt} + \frac{dh}{dx}\frac{d}{dt}\Big(\frac{dx}{dt}\Big), \tag{4.3}$$

$$\ddot{h} = \frac{d^2h}{dx^2}\dot{x}^2 + \frac{dh}{dx}\ddot{x}. \tag{4.4}$$

By plugging Eq. (4.4) into Eq. (4.1), we obtain,

$$\ddot{x} = \frac{g + \frac{d^2h}{dx^2}\dot{x}^2 + \frac{dh}{dx}\ddot{x}}{h}(x - p_x), \tag{4.5}$$

$$h\ddot{x} = \Big(g + \frac{d^2h}{dx^2}\dot{x}^2\Big)(x - p_x) + (x - p_x)\frac{dh}{dx}\ddot{x}, \tag{4.6}$$

$$\ddot{x} = \frac{g + \frac{d^2h}{dx^2}\dot{x}^2}{h - (x - p_x)\frac{dh}{dx}}(x - p_x). \tag{4.7}$$

Now, the term $\ddot{h}$ has been effectively removed. By numerically integrating the above equation, phase portraits can be obtained for given initial states $(x, \dot{x})$ and foot placements (p_x) (Fig. 4.2(a)).

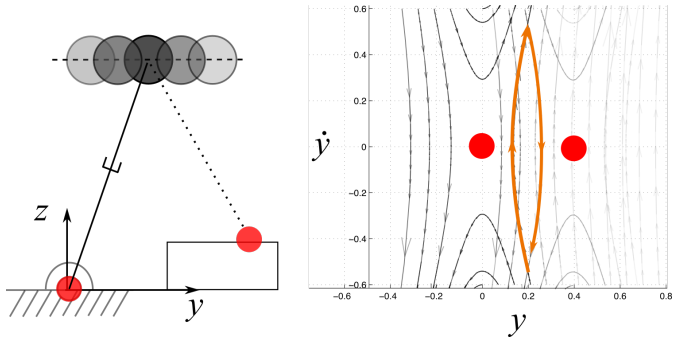


Figure 4.3: **PSP Reversal Step.** In the lateral direction, cyclic behavior is desirable to achiever regular walking behavior.

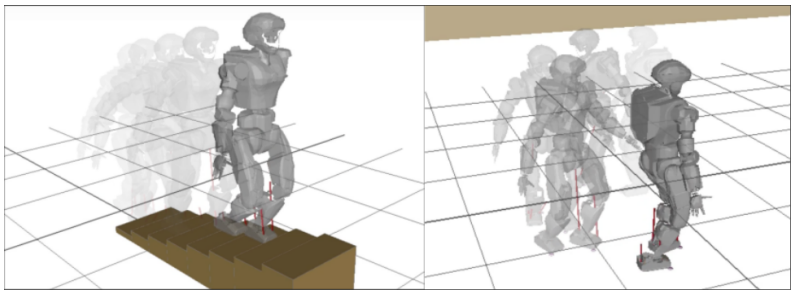


Figure 4.4: **Various Walking Demonstrations.** Valkyrie walks over stairs and makes turns with the path planned by a phase space planner.

Given an initial state, the CoM path is obtained via forward integration. On the other hand, the path associated with the next steps can be predicted backwards also via numerical integration. The crossing state shown in Fig. 4.2(b) depicts the switching state, which is the moment that the stance leg switches over to the new contact. We can easily find lateral foot positions given the switching times and the times to the next apex state. Fig. 4.3 shows the shape of the lateral CoM trajectory.

Fig.4.6(a) shows details on how the planner computes the switching times and lateral foot positions. When the CoM's current state is delivered to the planner together with the desired foot location the planner first computes the time when the stance foot needs to be switched. With this time, the planner computes the lateral CoM switching state and then finds the the next lateral foot placement. This method assigns zero lateral velocity at the apex since it results in natural and stable forward walking.

Fig. 4.4 shows that the PSP method can plan stair walking and turning behaviors. In the

simulations, WBC generates torque commands to execute the planned motions given by the PSP method.

4.1.2 Analytic Computational Algorithm

When we constraint PIP dynamics to a piecewise linear height surface, $z = a(x - p_x) + b$, we can find the foot switching time, t_{switch}, and the lateral foot location, p_y, without numerical integration and bisection search because the system of equations becomes linear, resulting in the following CoM behavior:

$$\begin{aligned} x(t) &= Ae^{\omega t} + Be^{-\omega t} + p_x, \\ \dot{x}(t) &= \omega(Ae^{\omega t} - Be^{-\omega t}), \end{aligned} \tag{4.8}$$

where,

$$\begin{aligned} \omega &= \sqrt{\frac{g}{ap_x + b}}, \\ A &= \frac{1}{2}\Big((x_0 - p_x) + \frac{1}{\omega}\dot{x}_0\Big), \\ B &= \frac{1}{2}\Big((x_0 - p_x) - \frac{1}{\omega}\dot{x}_0\Big). \end{aligned} \tag{4.9}$$

Note that this equation is the same for the lateral CoM, y, direction. Based on Eq. (4.8), we can find an analytical solution for PSP, summarized in Algorithm 2. $\mathbf{x}_1$, $\mathbf{y}_1$, $\mathbf{x}_{apex,2}$, and $\mathbf{x}_{switch}$ are vector quantities corresponding to the variables $(x_1, \dot{x}_1)$, $(y_1, \dot{y}_1)$, $(x_{apex,2}, \dot{x}_{apex,2})$, and $(x_{switch}, \dot{x}_{switch})$. Let us focus on obtaining the step switching time. We can easily manipulate

Algorithm 2: Computation of t_{switch}, p_y

Input: $\mathbf{x}_1, \mathbf{y}_1, p_x, \dot{x}_{apex}, \dot{y}_{apex}$
Result: (t_{switch}, p_y)

$\mathbf{x}_{switch} \leftarrow$ `Find_Switching_State(`$\mathbf{x}_1, p_x, \dot{x}_{apex}$`)` ;
`//` Eq.(4.16), (4.17)

$t_{switch} \leftarrow$ `Get_Time(`$\mathbf{x}_1, \mathbf{x}_{swtich}$`)` ; `//` Eq.(4.11)

$t_{apex} \leftarrow$ `Get_Time(`$\mathbf{x}_{switch}, p_x, \mathbf{x}_{apex}$`)` ; `//` Eq.(4.11)

$\mathbf{y}_{switch} \leftarrow$ `GetState(`$\mathbf{y}_1, t_{switch}$`)` ; `//` Eq.(4.8)

$p_y \leftarrow$ `Find_Py(`$\mathbf{y}_{switch}, \dot{y}_{apex}, t_{apex}$`)` ; `//` Eq.(4.18)

Eq. (4.8) to analytical solve for the time variable,

$$\begin{aligned} x + \frac{1}{\omega}\dot{x} &= 2Ae^{\omega t} + p_x, \\ x + \frac{1}{\omega}\dot{x} - p_x &= 2Ae^{\omega t}, \end{aligned} \tag{4.10}$$

which renders

$$t = \frac{1}{\omega}\ln\Big(\frac{x + \frac{1}{\omega}\dot{x} - p_x}{2A}\Big). \tag{4.11}$$

To find the dynamics, $\dot{x} = f(x)$, which will lead to the switching state solution, let us remove the t term by plugging Eq. (4.11) into Eq. (4.8).

$$x = A\frac{x + \frac{\dot{x}}{\omega} - p_x}{2A} + B\frac{2A}{x + \frac{\dot{x}}{\omega} - p_x} + p_x, \tag{4.12}$$

$$\frac{1}{2}(x - p_x - \frac{\dot{x}}{\omega}) = \frac{2AB}{x + \frac{\dot{x}}{\omega} - p_x} \tag{4.13}$$

$$(x - p_x)^2 - \Big(\frac{\dot{x}}{\omega}\Big)^2 = 4AB \tag{4.14}$$

By performing some algebra we get,

$$\begin{aligned} \dot{x}^2 &= \omega^2((x - p_x)^2 - 4AB), \\ \dot{x}^2 &= \omega^2\Big((x - p_x)^2 - (x_0 - p_x)^2\Big) + \dot{x}_0^2, \end{aligned} \tag{4.15}$$

which yields,

$$\dot{x} = \pm\sqrt{\frac{g}{h}\Big((x - p_x)^2 - (x_0 - p_x)^2\Big) + \dot{x}_0^2}. \tag{4.16}$$

Given two phase trajectories associate with consecutive walking steps, $p_{x,1}$ and $p_{x,2}$ and assuming the robot walks forward, i.e. $\dot{x}_{switch}$ is positive, we calculate the phase space intersection point via continuity of velocities from Eq. (4.16):

$$\begin{aligned} x_{\text{switch}} &= \frac{1}{2}\Big(\frac{C}{p_{x,2} - p_{x,1}} + (p_{x,1} + p_{x,2})\Big) \\ C &= (x_{0,1} - p_{x,1})^2 - (x_{0,2} - p_{x,2})^2 + \frac{\dot{x}_{0,2}^2 - \dot{x}_{0,1}^2}{\omega^2} \end{aligned} \tag{4.17}$$

We can now find the step switching time by plugging the computed switching position into Eqs (4.16) and (4.11). In addition, we can obtain the timing at the apex velocity from Eq. (4.11).

The final step is to find the y directional foot placement. We first calculate $\mathbf{y}_{switch}$ by plugging t_{switch} into the y directional state equation, which has identical form to Eq. (4.8). Then, by using the equality that $\dot{y}(t_{\text{apex}}) = \dot{y}_{apex}$, we can find p_y,

$$
\begin{aligned}
p_y &= \frac{\dot{y}_{apex} - C}{D}, \\
C &= \frac{\omega}{2}\big((y_{switch} + \frac{\dot{y}_{switch}}{\omega})e^{\omega t_{apex}} - (y_{switch} - \frac{\dot{y}_{switch}}{\omega})e^{-\omega t_{apex}}\big) \\
D &= \frac{\omega}{2}(e^{-\omega t_{apex}} - e^{\omega t_{apex}})
\end{aligned}
\tag{4.18}
$$

After calculating p_y, we can easily get y_{apex} by using Eq. (4.8).

The analytic method renders the potential for utilizing PSP in the area requiring heavy computation such as path planning or machine learning. In [2], we proposed an algorithm for sampling-based kino-dynamic planning and control for a bipedal robot in complex environments. We plan both dynamically and kinematically consistent step positions using a kino-dynamic RRT with a steering function customized for a biped. We exploit a novel metric function instead of using the Euclidean metric which is not related to the robot's dynamics. We take advantage of the analytic solutions of PSP to obtain the proper step position and the full state of the CoM for each step. We demonstrate this algorithm in a complex simulation environment which includes both static and moving obstacles with a human-sized humanoid robot, Valkyrie. To control the full body motion of Valkyrie, we use WBLC in Section. 2.3.

To validate the proposed algorithm, we test it with a full human-sized bipedal robot, Valkyrie, in a dynamic simulator, srLib[1]. In the simulation, Valkyrie's starting location is in the right upper corner of a maze in an 18 × 14m room. The maze (shown in Fig. 4.5(a)) is formed with red walls and has three mobile robots (shown as small gray boxes) that move through known trajectories (shown as red arrows). The mission is to walk to the door at the left bottom corner of the room while avoiding static and moving obstacles.

Our planner successfully finds the solution route to the goal location in 70 s at most including rewiring process for 108 steps. In Fig. 4.5(b), the blue trajectory illustrates the initial finding and the red trajectory shows the rewired solution. The multicolored dots in the figure are the nodes found during the exploration process which were not part of the solution sequence. Since the algorithm is based on random sampling, the computation time and final solution from each trial is not identical. As shown in Fig. 4.5(c), the solution can be visualized as navigating

[1]Seoul National University Robotics Library. Open-source `http://robotics.snu.ac.kr/srlib/`

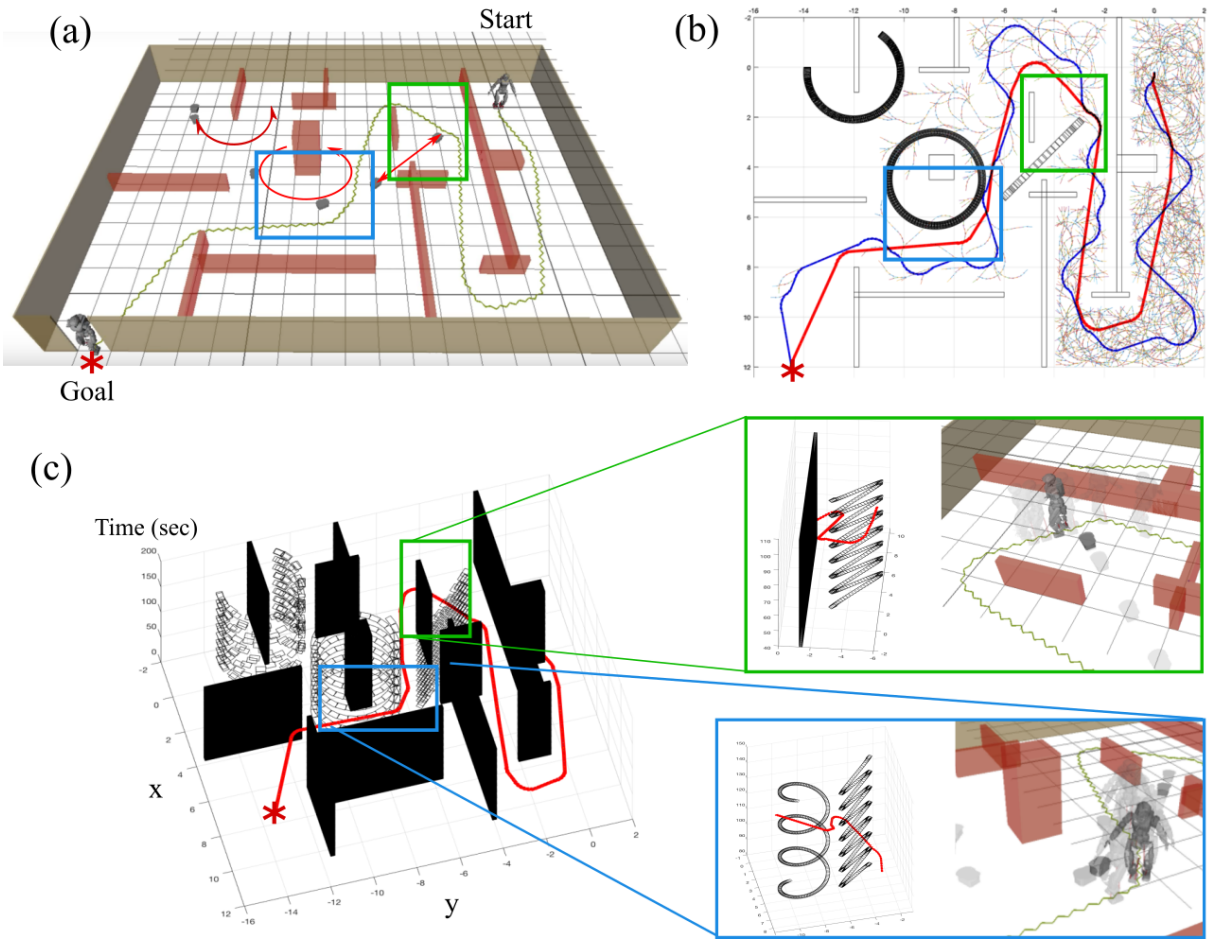


Figure 4.5: **Kinodynamic RRT Locomotion Planning.** (a) shows the robot's initial state and goal position (✱). The rooms are separated by red walls and three mobile robots are moving around at known speeds. Two robots are revolving around a certain point and the other one is moving linearly. (b) shows the top view of (c). After the planner finds the solution path (blue), the rewiring process smooths out the path (red). (c) shows the graph in time domain as well as Cartesian space. We can see the static walls stand still over time and dynamic obstacles moving through time. Over the three figures, the locations indicated by green and blue squares are identical.

through the time domain as well as Cartesian space. Note that static obstacles such as the walls remain stationary as time increases, while the moving obstacles do not. When the biped passes the moving obstacle revolving around the center of the maze (see the blue box magnification), some nodes are included in the tree which cross paths with the mobile robot while it is on the other side of the circle. To control the full body motion of Valkyrie, we generate a CoM task and foot task for the WBLC from the solution sequence. Note this planning situation is significantly more complicated than most practical planning problems, which usually plan footstep sequences over much smaller distances.

Despite the fact that the above situation is somewhat unrealistically complex for practical bipedal planning problems, we chose to test the algorithm with this situation because it demonstrates some of the strengths and capabilities of our algorithm. In the process of solving that problem, the planner typically had on the order of 15000 nodes in the tree when it found a solution. However, this algorithm works as well for simpler planning problems as it does for the highly complex one presented above. When there are fewer tight clearance passages between the starting location and the goal location as is the case in most practical biped navigation planning problems, the planner often can find and rewire a solution with 100's of steps in fractions of a second.

4.2 Reinforcement Learning Based Phase Space Planner

Reinforcement Learning (RL), which commonly involves iterative searches to find a solution, is also an excellent application of PSP with the analytic method. Moreover, PSP is exceptionally beneficial to formulate the RL problem since it inherently constrains the forward walking along the x axis and a cyclic motion in the y plane. This formulation removes a lot of meaningless explorations resulting in back step, side step, or leg crossing, which can be selected many times under plain RL formulation settings.

In this section, we explain a robust dynamic locomotion planner using an RL process. Previous approaches specify either the position or the timing of steps; however, the proposed locomotion planner simultaneously computes both of these parameters as locomotion outputs. Our locomotion strategy relies on devising an RL approach for robust walking. The learned policy generates multi-step walking patterns, and the process is quick enough to be suitable for real-time controls. For learning, we devise an RL strategy that uses a phase space planner (PSP) and a linear inverted pendulum model to make the problem tractable and very fast. Then, the learned policy is used to provide goal-based commands to whole-body locomotion

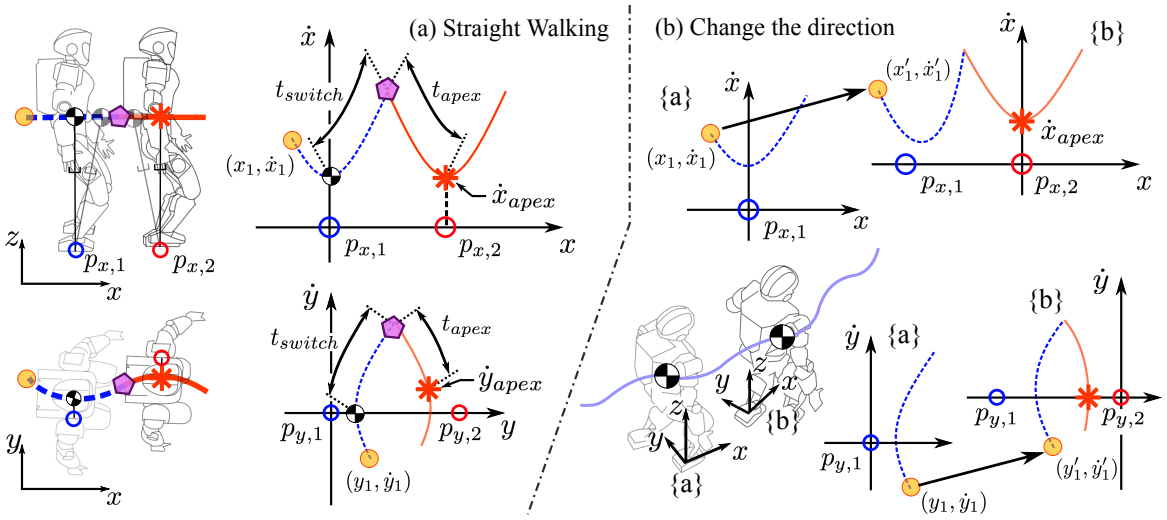


Figure 4.6: **Phase Space Planner.** (a) shows the method to find a switching time and a lateral foot placement position given forward step location and an apex velocity. In the sagittal phase plot, we can see the given current CoM state and the apex state uniquely define switching states ⬟. From an initial state, the planner computes the switching and apex times. These two timing values are used to find the next step location in the lateral plane. (b) is the process to steer the robot's walking direction. When changing the walking direction, we first align the orientation of the next local frame to the direction the robot intends to go, and second we project the current CoM state into the next local frame.

controller (WBLC) in Section 2.3, which calculates the torque commands to be executed in full-humanoid robots. The integration of RL-PSP and the WBLC provides highly robust, versatile, and practical locomotion including steering while walking and handling push disturbances of up to 520 N during an interval of 0.1 s. Theoretical and numerical results are tested through a 3D physics-based simulation of the humanoid robot Valkyrie.

4.2.1 Phase Space Planner Review

Leading step planning generators, such as divergent component of motion (DCM) [17], find CoM paths given step positions and their timing as input information. The ZMP Preview Control method [37] has different mechanics but the generated output can be interpreted as finding the CoM path given step position and timing input information. In contrast, PSP finds the step switching time and lateral foot positions, given sagittal foot positions and apex velocities (Fig. 4.6). The apex states are those at the instant when the sagittal CoM velocity is at its

minimum; equivalently, they can be considered as states where the sagittal CoM position is zero in a local frame attached to the stance foot, i.e. below the CoM sagittal position.

In Fig. 4.6(a), we can see that the current robot's CoM state and the next desired apex state uniquely define a switching state ⬟, a switching time, and an apex time. These timings are used to find the next lateral foot position, $p_{y,2}$. Note that the resultant locomotion trajectory is straight forward if $\dot{x}_{apex}$ is a positive number and $\dot{y}_{apex} = 0$. In contrast, the proposed algorithm, applies a simple and elegant modification that allows to dynamically steer the biped in any direction of walking (see Fig. 4.6(b)). When we need to turn walking direction, we re-initialize the orientation of the local frame $\{b\}$ to the new direction and project the current state to the new frame. The original PSP algorithm devised locomotion trajectories via numerical integration. However, for algorithmic speed purposes, the methods presented here assume that the CoM height is linear allowing us to exploit an analytical solution in Section. 4.1.2.

Considering a one step ahead plan, an initial CoM state, and desired future states $[p_x,\ \dot{x}_{apex},\ \dot{y}_{apex}]^\top$ PSP finds the next step position and timing, $\begin{bmatrix} p_y, & t_{switch} \end{bmatrix}^\top$. Notice that the walking direction is indicated using apex velocities, $[\dot{x}_{apex},\ \dot{y}_{apex}]^\top$. We will now see that our formulation of PSP makes the RL problem more efficient by reducing the dimensionality of the learned state variables.

4.2.2 The Reinforcement Learning Formulation

The technique we use is the actor-critic with eligibility traces method. We summarize this process in Algorithm 3 which is an adaptation [100]. We define $\mathbf{s}$, as CoM apex states, $\mathbf{s} \triangleq \begin{bmatrix} y_{apex}, & \dot{x}_{apex}, & \dot{y}_{apex} \end{bmatrix}^\top$. Notice that $\mathbf{s}$ does not include the variable x_{apex} because it is assumed to be always zero in the local frame. We define actions, $\mathbf{a} \triangleq \begin{bmatrix} p_x, & \dot{x}_{apex}, & \dot{y}_{apex} \end{bmatrix}^\top$, as input parameters to the PSP process. A transition function, $T(\mathbf{s},\ \mathbf{a})$, computes the next apex state, $\mathbf{s}'$, and the instantaneous reward value. In Fig. 4.7, we show the transition function, consisting of two stages: 1) finding step timing and position values via PSP, and 2) computing the next apex state via an analytic solution of the linear inverted pendulum (LIP) model. The first stage is described in Section 4.1.2 and allows to find t_{switch}, t_{apex}, and p_y from the current apex state and chosen action. The second stage, finds the next apex state using the analytic solution of the CoM dynamics (see Eq. (4.8)). In Algorithm 2, the process of finding the switching times and the next apex states is explained in detail.

The next item, $\hat{v}(\mathbf{s}, \boldsymbol{w})$, corresponds to the value function - similar to the cost-to-go function in Dynamic Programming. We store its learned values using a radial basis function (RBF)

Algorithm 3: Actor-Critic with Eligibility Traces

Input: $\hat{v}(\mathbf{s}, \mathbf{w})$, $\forall \mathbf{s} \in \mathcal{S}$, $\mathbf{w} \in \mathbb{R}^{18\times30\times56+1}$
Input: $\pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\theta})$, $\forall a \in \mathcal{A}$, $\mathbf{s} \in \mathcal{S}$, $\boldsymbol{\theta} \in \mathbb{R}^{(18\times30\times56+1)\times6}$

Result: $\boldsymbol{\theta}$, $\mathbf{w}$

Initialize policy weights θ and state-value weights $\mathbf{w}$ **while** *(variances of policy is large)* **do**

  Randomly pick $\mathbf{s}$ in $\mathcal{S}$
  $\mathbf{e}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (eligibility trace of policy parameters)
  $\mathbf{e}^{\mathbf{w}} \leftarrow \mathbf{0}$ (eligibility trace of value parameters)
  $I \leftarrow 1$
  while *(*$\mathbf{s}$ *is not terminal)* **do**
    $\mathbf{a} \sim \pi(\cdot|\mathbf{s}, \theta)$
    $\mathbf{s}', R \leftarrow T(\mathbf{s},\ \mathbf{a})$
    $\delta \leftarrow R + \gamma\hat{v}(\mathbf{s}', \mathbf{w}) - \hat{v}(\mathbf{s}, \mathbf{w})$
    $\mathbf{e}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}}\mathbf{e}^{\mathbf{w}} + I\nabla_{\mathbf{w}}\hat{v}(\mathbf{s}, \mathbf{w})$
    $\mathbf{e}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}}\mathbf{e}^{\boldsymbol{\theta}} + I\nabla_{\boldsymbol{\theta}} \log\pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \beta\delta\mathbf{e}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\mathbf{e}^{\boldsymbol{\theta}}$
    $I \leftarrow \gamma I$
    $\mathbf{s} \leftarrow \mathbf{s}'$
  end
end

neural network [10]. The network uses a three-dimensional input vector consisting of the CoM apex state.

$$\begin{aligned} &\cdot \quad -0.14 \leq y_{apex} \leq 0.2 \text{ (m)}, \\ &\cdot \quad 0.03 \leq \dot{x}_{apex} \leq 0.61 \text{ (m/s)}, \\ &\cdot \quad -0.55 \leq \dot{y}_{apex} \leq 0.55 \text{ (m/s)}. \end{aligned} \tag{4.19}$$

The hidden layer consists of a bias term and $18 \times 30 \times 56$ Gaussian functions with centers on a grid with 2cm spacing along each input dimension. The policy function also consists of an RBF neural network but a little different from the value function because actions are chosen based on an stochastic evaluation.

Fig. 4.8 shows that outputs of the RBF network are means and standard deviations of truncated normal distributions, $\pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\theta})$. The range of the distributions are selected by considering

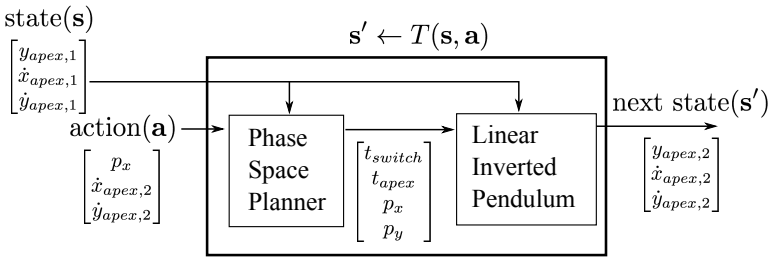


Figure 4.7: **Transition Function.** The transition function relies on two models: PSP and LIP model. Given an apex state and a considered action, PSP computes step timing and location information that serve as inputs to the LIP model. Then, LIP model solves for the next apex state based on the given state and the provided inputs.

the desired walking speed and step length limits as follows:

$$\begin{aligned} &\cdot\ 0.1 \leq p_x \leq 0.5 \text{ (m)}, \\ &\cdot\ 0.03 \leq \dot{x}_{apex} \leq 0.37 \text{ (m/s)}, \\ &\cdot\ -0.25 \leq \dot{y}_{apex} \leq 0.25 \text{ (m/s)}. \end{aligned} \tag{4.20}$$

The network's outputs are linearly weighted by $\boldsymbol{\theta}$; thus, the purpose of RL is to find the weights $\boldsymbol{\theta}$ given candidate actions that minimized the desired cost.

The instantaneous reward is defined by the forward velocity error and lateral step size error:

$$R = -(\dot{x}_{apex}^{nom} - \dot{x}_{apex})^2 - 15 \times (p_y^{nom} - p_y)^2 - (\dot{y}_{apex})^2. \tag{4.21}$$

The set target for the learning process is to achieve recovery behaviors that maintain a straight forward direction, $\dot{y}_{apex} = 0$ while keeping a nominal lateral directional step size. We choose $\dot{x}_{apex}^{nom} = 0.2$m/s and $p_y^{nom} = 0.3$m. The reward comes from the transition function described before, given current apex and action states selected from the truncated distributions.

If the next predicted apex state incurs a terminal condition, the transition function gives a negative reward of -5.0, and the process terminates and starts a new iteration. The set of safe conditions (i.e. opposite to the terminal conditions) is the intersection of the following predicates:

$$\begin{aligned} &\cdot\ t_{apex} > 0.12 \text{ (s)}, \\ &\cdot\ t_{switch} > 0.12 \text{ (s)}, \\ &\cdot\ 0.1 < p_y < 0.5 \text{ (m)}, \end{aligned} \tag{4.22}$$

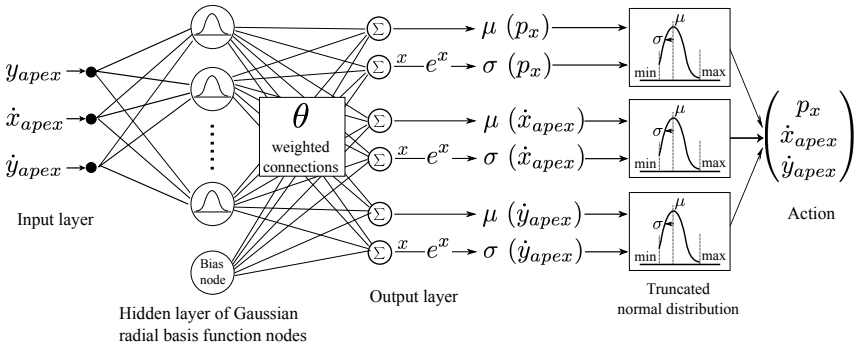
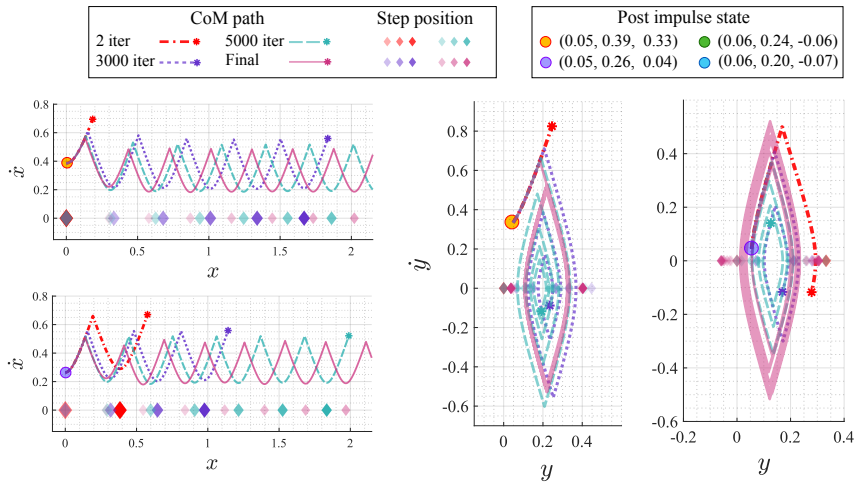


Figure 4.8: **Radial Basis Function Neural Network for Walking Policy Representation.** Outputs of the neural network are means and standard deviations of each action value. The truncated normal distributions defined by the outputs are used to stochastically pick actions.

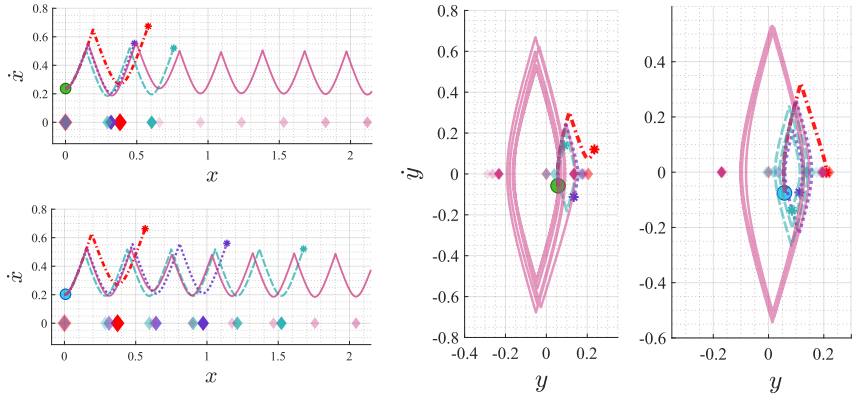
which reflect the ability of the robot to swing its legs and the lateral step length. Notice that we do not include a predicate about the sagittal step length because it is already bounded by the allowable action range. The learning process ends when the variance of the learned policy becomes small enough (< 0.07 units in our case). The usual number of iterations required to complete the learning process is about 30,000, and the process usually takes about 1 min to compute on a dual-core, 3.0 GHz, Intel i7 processor thanks to the speed of our analytic PSP method.

4.2.3 Evaluation of Learned Policy

Fig. 4.9 shows that the performance of the RL-based planner increases as the number of iterations increases. By watching the posture of a robot at different CoM states, we choose the nominal apex state to be $\left[y_{apex},\ \ \dot{x}_{apex},\ \ \dot{y}_{apex}\right] = [\ 0.056,\ 0.2,\ 0\ ]^{\top}$. We proceed by simulating push disturbances to the CoM based on various external forces and directions. We use mean values of the final learned policy as desired actions rather than randomly picking actions from the normal distributions. The results are shown in Fig. 4.9 showing the learned policy obtained after many iterations and their enhancements on the walking patterns. In this figure, initially our simulated robot stands with the right foot on the ground and we simulate push disturbances to the left, right, and forward directions of its body. For example, the 🟡 post impulse apex state, $[\ 0.05,\ 0.39,\ 0.33\ ]^{\top}$, is the result of an impulse applied to the left-forward direction of the robot's body. Red lines are interrupted within a few walking step indicating that the initial policies fail to find proper actions. In contrast, pink lines correspond to the final learned policy which achieves infinite walking steps without falling given the initial push disturbances.



(a) push outward



(b) push inward

Figure 4.9: **Phase Plots of Sequential Steps from Learned policies.** The initial state considered here corresponds to an impulsive disturbance. The candidate phase trajectories generated by the policy function reach terminal states if they are unsuccesful. As learning proceeds, the policy function finds better actions which avoid terminal states. The final policy achieves infinite number of steps without reaching terminal conditions.

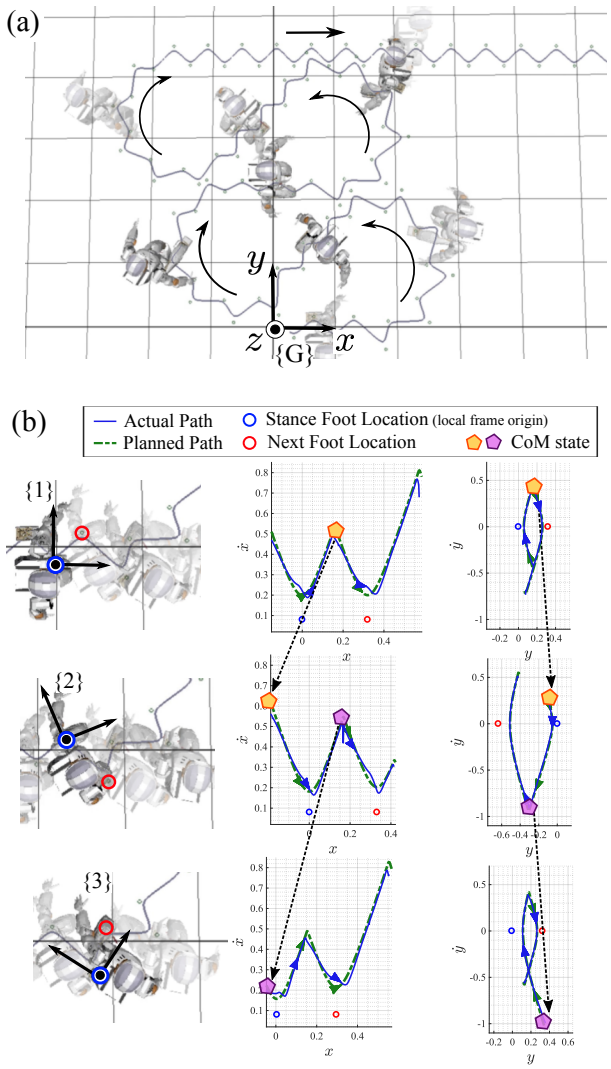


Figure 4.10: **Continuous walking directional change** Valkyrie shows a complex dynamic walking pattern involving changing walking direction. (a) shows a top view of Valkyrie and its walking path. (b) Shows how the robot's CoM state mapped to the next local frame. Local frames rotate with the desired walking direction. For each step, the stance foot becomes the origin of the local frame, and the orientation of the frame is aligned with the desired walking direction. The previous switching CoM state is projected to the current local frame, and the planner finds the foot placement with the new state.

4.2.4 Result 1: Dynamic Walking with Directional Change

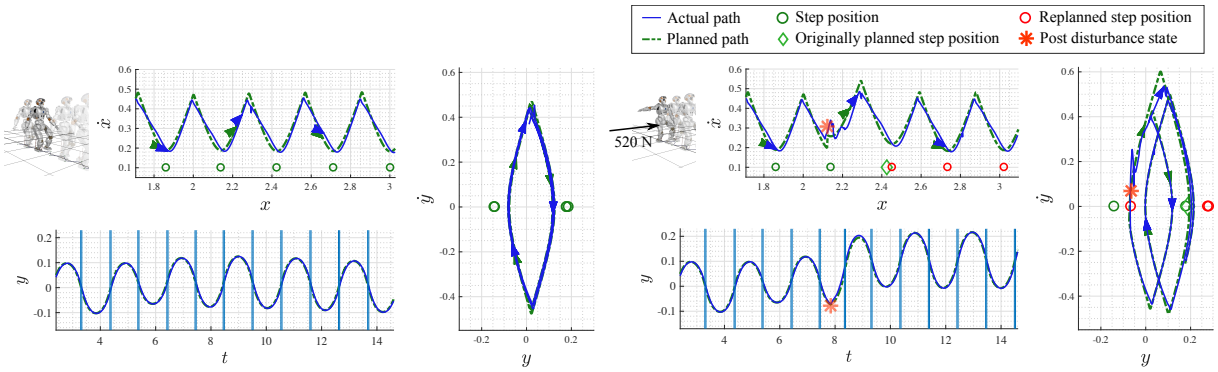
To verify the performance of the proposed methods, we conduct two demonstrations: 2) dynamic locomotion with directional change and 2) push-recovery from various directions while walking. Toward these investigations, we implement our algorithms on a simulation of the Valkyrie humanoid robot, and test it using the physics based simulation SrLib. Because our focus is on locomotion, we fix the finger and wrist joints, bringing the total number of joints to 28. To incorporate floating body dynamics, prismatic and ball joints are introduced to connect Valkyrie's pelvis to a fixed frame.

In the simulation environment, we use a friction coefficient between the ground and the robot's feet of 0.8. On the other hand the friction cone constraints used in WBLC are set to a value of 0.65 to be conservative. In case our contact control solver fails to find proper reaction forces, we allow for solutions that violate friction constraints by relaxing the friction coefficient to a value of 1.75. The resulting control solution implies that slip occurs but only for very short times (in general less than 0.005 s). This simple technique doesn't incur an increase in computational complexity while greatly enhancing the robustness of WBLC with respect to external disturbances.

Walking can be broken down into three phases: double contact, right foot contact, and left foot contact. To represent these phases we define the following task hierarchy in WBLC:

- $\ddot{\mathbf{x}}_1 \in \mathbb{R}^3$: Linear CoM position
- $\ddot{\mathbf{x}}_2 \in \mathbb{R}^3$: Pelvis Orientation
- $\ddot{\mathbf{x}}_3 \in \mathbb{R}^3$: Body Orientation
- $\ddot{\mathbf{x}}_4 \in \mathbb{R}^3$: (for the single contact phases) Foot Orientation
- $\ddot{\mathbf{x}}_5 \in \mathbb{R}^3$: (for the single contact phases) Foot Position
- $\ddot{\mathbf{x}}_6 \in \mathbb{R}^6$: Neck and Torso Joint Posture
- $\ddot{\mathbf{x}}_7 \in \mathbb{R}^3$: Centroidal Angular Momentum
- $\ddot{\mathbf{x}}_8 \in \mathbb{R}^{10}$: Arms Joint Posture

To produce swing foot trajectories, we define third degree B-splines, which guarantee acceleration continuity. The orientation coordinates for the robot's body, pelvis, and feet are described using



(a) Walking w/o external disturbance

(b) Walking w/ replanning

Figure 4.11: **Robustness study.** (a) Shows a walking behavior without external disturbance. (b) When an external impulse of 520 N and 0.1 s duration is exerted on the robot's pelvis, Valkyrie replans its walking trajectories using the learned policy and maintains its balance without stopping.

quaternions. For each step, these orientation tasks are commanded to smoothly switch from the current frame to next one. Given initial CoM states, our locomotion planner computes foot positions and their timing while satisfying the desired walking directional changes. In our test shown in Fig. 4.10, Valkyrie takes first 12 steps while continuously changing its walking direction by 18.8° per step. After that, Valkyrie takes 5 forward steps with no directional change. Then, Valkyrie takes another 12 steps while changing direction by, -18.8° per step. The user only specifies the walking directions while RL-PSP automatically finds the foot positions and their timing using the learned policy. The learned policy consists only on switching states and step locations. The desired position, velocity, and acceleration of the CoM are computed with the analytic equation of the LIP model at runtime.

4.2.5 Result 2: Push Recovery while Walking

To validate push recovery, we conduct simulated experiments under large external disturbances and in various directions. Although WBLC is robust to small deviations of the CoM trajectory, for external disturbances we rely on the learned recovery policies described in the theory sections. When the norm of the CoM state error,

$$\mathbf{error} = \begin{bmatrix} \mathbf{x}^d - \mathbf{x} \\ 0.5(\dot{\mathbf{x}}^d - \dot{\mathbf{x}}) \end{bmatrix}, \tag{4.23}$$

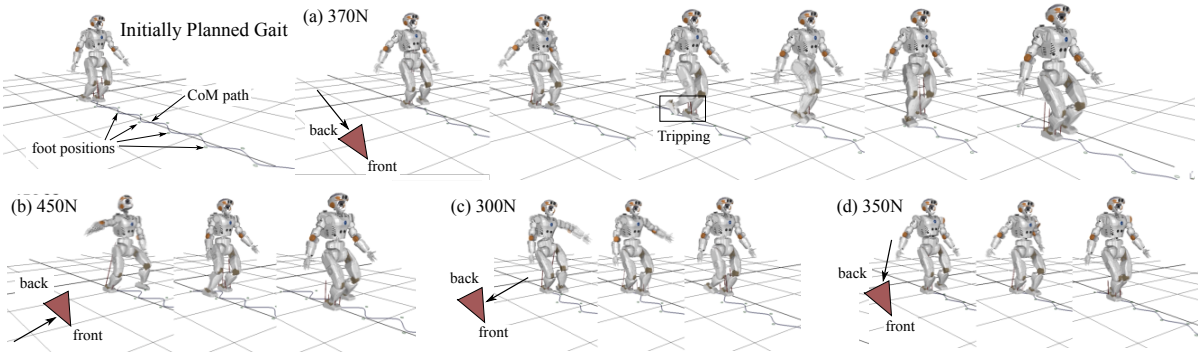


Figure 4.12: **Details of push recovery given various external forces.** In this test we demonstrate the ability of Valkyrie to recover from pushes of various magnitudes and directions. In (a), Valkyrie's feet collide with each other, but the planner finds another path that allows it to recover.

is over a threshold equal to 0.05 m and for longer than 0.02s, our planner computes a new trajectory starting from the current CoM state. Instead of setting the new CoM control goal to be the current (disturbed) CoM state, we have experienced that it is better to define a controller goal, $\mathbf{x}^{new}$, equal to

$$\mathbf{x}^{new} = \gamma \mathbf{x}^{d} + (1 - \gamma)\mathbf{x}, \tag{4.24}$$

where γ can be selected heuristically, and we use a value of 0.8. In our tests, we push Valkyrie while she is dynamically walking using various disturbance forces applied for a duration of 0.1 s. The maximum disturbance impulse that we apply to Valkyrie is 520 N for 0.1 s. The results are shown in Fig. 4.11 compared to the undisturbed trajectories. The CoM phase trajectory in the lateral plane shown in Fig. 4.11 (b) shows that the planner is able to find a new trajectory after an external impulse is applied. The time to compute 15 steps after the disturbance is less than 1ms using a dual-core 3.0 GHz Intel i7 processor. At the moment that the replanning process occurs, we also find a new swing foot trajectory that transitions from the original swing trajectory to the new goal.

Fig. 4.12 shows results of push recoveries while dynamically walking when being subject to various external forces. In all cases, Valkyrie succeeds to sustain the disturbances and continue walking without stopping. The robustness capabilities in this test are competitive to the results of [42] which is not based on statistical learning. In contrast to this state of the art, due to our use of offline learning our planner is able to come up with numerous steps almost instantaneously with respect to the walking time frame.

4.2.6 Summary

In this section, we propose an RL based robust locomotion planner. By utilizing PSP in the RL formulation, we can quickly find locomotion policies for 3D walking. The major benefit of our methods is the planning speed. Our locomotion planner almost instantaneously finds a multistep walking trajectory faster than the state of the art. By devising the replanning process during dynamic walking, robots can quickly react to external forces and achieve significant robustness.

One interesting aspect of our planning algorithm is the value function we used in the learning process. In the future we could use this value function as an indicator for walking risk given the disturbed states. Many researchers have suggested indicators for locomotion quality. For example, ZMP [104] and CP [73] are indicators of balance stability but they don't take into account other important information such as kinematic constraints or swing time limits. Recently, an allowable CoM acceleration region [11] has been proposed for multi-contact stability. However, there is no indication of kinematic or dynamic limitations such as step size or swing time. In contrast our value function takes into account some kinematic and dynamic constraints that could ultimately make it a versatile metric for walking quality evaluation.

In the future, we will experiment with more complex functions to represent learned values and policies (e.g. deep neural network). In this section, we have focused on finding simple walking patterns. However, complex neural networks, which can represent highly nonlinear and abstract behaviors, can enable more versatile planners. For instance, future planners may be able to traverse rough terrain by exploiting various locomotion modes such as walking, running, or jumping. We also plan to implement the proposed algorithms in real systems and evaluate their performance. In our previous work [49], we showed agile bipedal balance with a point-foot biped with series elastic actuators. Since the system is highly unstable by nature, we did not apply external disturbances. We believe that the robustness capabilities we have outlined in this section may allow us to accomplish sophisticated behaviors in the real testbeds.

4.3 Step Position Planner for Velocity Reversal

This section presents a planner scheme for ensuring that a 3D, under-actuated, point-foot biped robot remains balanced while walking. It achieves this by observing the CoM position error relative to a reference path and re-planning a new reference trajectory to remove this error at every step. The PIP model is used to simplify behavioral analysis of the robot. We use phase space techniques to plan the CoM trajectories and foot placement. While obtaining a stable path using this simplified model is easy, when applied to a real robot, there will usually be

deviation from the expected path due to modeling inaccuracies. Although fully-actuated robots can reduce the deviation with relatively simple feedback control loops, when working with under-actuated robots, it is challenging to design such a feedback control loop. Our approach is based on continuous re-planning. By planning the path of the next step based on the observed initial error, we can find the proper landing location of each step. For each step we allocate sufficient time to avoid disturbances from the moment induced by the moving leg, which is not modeled in the PIP model. Our control scheme relies on the PIP model instead of the LIP model to enable non-planar CoM motion, which is essential for rough terrain locomotion.

4.3.1 Velocity Reversal

The step position planner determines footstep locations based on a PIP model that is representative of the robot in single contact phases. The footstep locations are selected such that the PIP model is stabilized. This should result in a stable balance behavior assuming the trajectory generators and WBC in Chapter. 2, 1) successfully place the feet at the desired locations, 2) achieve the desired height of the CoM, and 3) fix the orientation of the robot's body.

We devise a footstep location algorithm called the "phase space constant time to velocity reversal planner" [48]. The swing motion is separated into two phases, lifting and landing the foot. The landing location is computed before the landing phase starts. In every step, when the lifting phase reaches 70% completion, the planner computes the next footstep location. The value of 70% was empirically determined to ensure the planner completes before the landing phase starts. It is a processor intensive task that must be run outside of the realtime thread. The operational space set-point trajectory for the swing foot is then defined based on a polynomial function and the desired landing position, with the trajectory ending once ground contact is sensed. If the ground is at the expected height and the position tracking is ideal, the footstep will land after the nominal swing time. If the planned step is outside the mechanical limits of the robot, the planner chooses the closest reachable step.

The planner attempts to stabilize the robot by making its CoM reverse direction every step. In its simplified model, the feet instantly change between swing and support modes and the CoM reverses its velocity a time t' after the previous contact switch. In contrast to approaches based on linear models with analytically solvable dynamics such as the capture point (CP) or DCM methods, our approach is different because the PIP model is not analytically solvable and numerical search is used to plan the steps. This leads to the primary computational element in the planning procedure: a shooting method over the possible footstep locations to find a CoM trajectory that accomplishes the CoM velocity reversal goal. In [48], which only considered

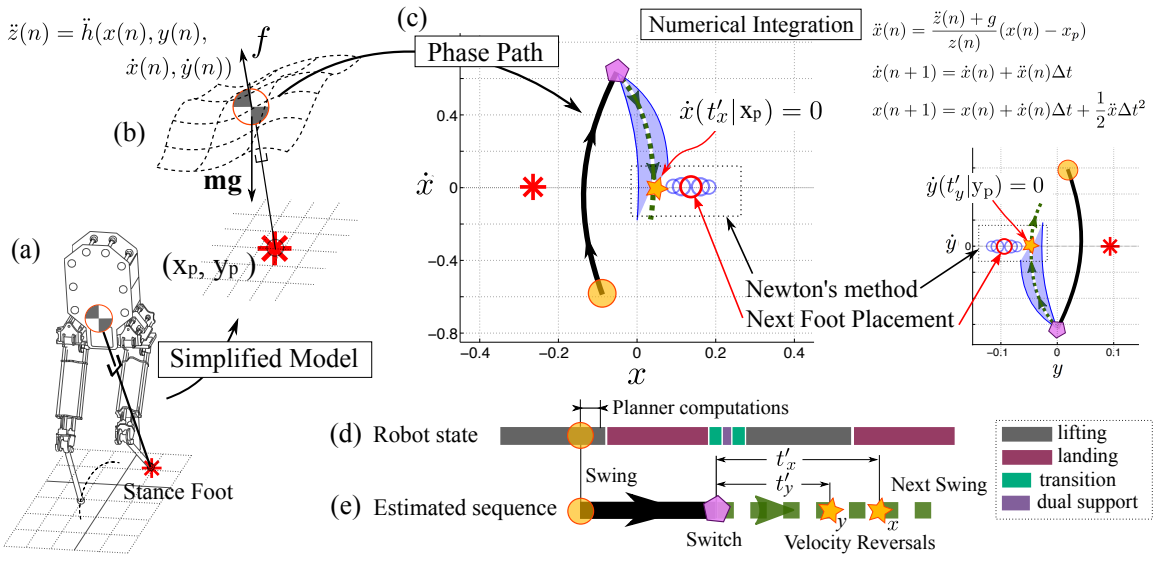


Figure 4.13: **Constant Time to Velocity Reversal Planner.** As shown in (a), we approximate the dynamics of the robot with the prismatic inverted pendulum model, shown in (b). This model predicts the dynamics of the horizontal CoM position x, y given the stance foot location $(p_x,\ y_p) =$ ✱ and the height surface $z = h(x,\ y)$. This can be integrated forward in time via the numerical integration procedure shown in (c). When the planner starts operating it records the initial state ● and integrates this state forward to determine the switching state ⬟. As shown in timelines (d) and (e), the "Estimated sequence" of the planner has an analogy in the "Robot states" of the state machine. In particular, the switching state ⬟ roughly corresponds to the double support phase of the walking state machine. This state ⬟ represents the planner's guess at the time and state $(x, \dot{x}, y, \dot{y})$ values immediately after the switch. The goal of the planner is to stabilize the robot, and this is achieved by choosing the next footstep ○ such that x and y velocity equal zero t'_x and t'_y seconds, respectively, after the foot switch every step. For sufficiently smooth height surfaces, the relationship between the next footstep location and the velocity is monotonic, so only a single solution exists. We use Newton's method with numerical differentiation to identify this solution. There exist two velocity reversal states: ★$_x$ and ★$_y$ as shown in (c).

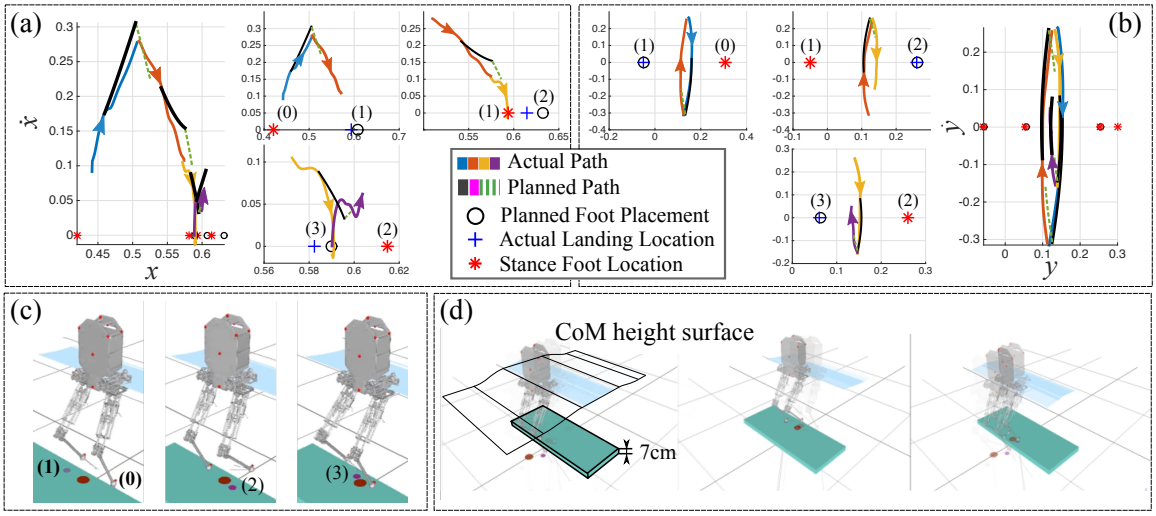


Figure 4.14: **Simulation of Unsupported Point Foot Walking Over Rough Terrain.** Subfig. (a) shows the sagittal CoM phase portrait of three steps of walking over a rough terrain, while (b) shows the lateral CoM phase portrait. The smaller plots correspond to individual steps. The corresponding three steps are shown in (c) using the SrLib Multi-Body Dynamic Simulation Environment. Subfig. (d) shows more steps of the dynamic walking simulation over the rough terrain.

constant elevation of the CoM, we used bisection search to find footstep locations. In this completed version, which considers variable elevations, we use Newton's method based on numerical derivative approximations to find the footsteps.

As illustrated in Fig. 4.13, the planner begins calculating the landing location when 70% of the lifting phase is reached. As such, the planner continuously re-plans to correct for trajectory deviations. Using the current estimate of the CoM velocity and position, it numerically integrates forward until the predefined swing time ends to predict its CoM position and velocity when its stance foot and swing foot will switch roles, ⬟ in Fig. 4.13. The equation used in the numerical integration is presented in Appendix. B.

The implementation of the planner enforces the choice of t'. This time value remains constant for every step, thus the user only needs to specify a single parameter. The method for selecting t' is explained in Section 4.3.2. The planner then finds and returns the footstep location that causes the robot's CoM velocity to reach zero t' seconds after the foot switch. For each potential footstep location considered, the planner integrates forward in time starting from the post-impact state as suggested in Fig. 4.13, ensuring the velocity is zero after t' seconds. This integration

can be viewed as a function mapping footstep location to a future velocity. Newton's method uses this function to find a step position that results in zero CoM velocity.

Newton's method converges very fast because of the monotonic relation between the foot location, p_x and p_y, and velocity change after t' seconds, $\dot{x}(t'_x|p_x)$ and $\dot{y}(t'_y|p_y)$ [48]. Using the equations,

$$\dot{x}(t'|p_x) = \int_0^{t'} \frac{g + \frac{d^2z}{dx^2}\dot{x}^2}{z - (x - p_x)\frac{dz}{dx}}(x - p_x)dt,$$

we can easily derived the partial derivative,

$$\frac{\partial \dot{x}(t'|p_x)}{\partial p_x} = -\int_0^{t'} \frac{z(g + \frac{d^2z}{dx^2}\dot{x}^2)}{\left(z - (x - p_x)\frac{dz}{dx}\right)^2}dt. \tag{4.25}$$

Because the denominator of Equation (4.25) is a square value, if the nominator is always positive, i.e. $\frac{d^2z}{dx^2} > -\frac{g}{\dot{x}^2}$ and $z > 0$, then $\frac{\partial(\dot{x}(t'|p_x)}{\partial p_x} < 0$ forall p_x. This inequality implies that $\dot{x}(t'|p_x)$ will always decreased with p_x which implies monotonicity. In the simulation, the proposed method converges within ten iterations.

As shown in Fig. 4.14, our planner allows Hume to step over a 7 cm tall platform and using a CoM height surface that conforms to the terrain. Specifically, this height surface is defined, piece-wise, as a function of a global x coordinate, with three constant height pieces connected by two sinusoidal segments. The middle constant height piece is 7 cm above the others to account for the obstacle's height. The surface maintains first order continuity. In this simulation the robot's planner follows a moving goal location. As this goal location passes over the platform the robot ascends and descends it while constantly stepping. Although the planner usually finds the proper step position, the robot shows variable forward and backward swinging motions. This variability arises when the robot performs large motions such as stepping down from a platform. Since the simplified PIP model used for planning does not account for the multi-limb robot dynamics, predicting the CoM path becomes increasingly more difficult when estimating large motions. Despite this discrepancy, the planner and controller successfully achieve a stable walk over the challenging terrain. In the simulation, we assume zero time delay, perfect sensor data, perfect torque tracking, and correct dynamic and kinematic models. These assumptions, which are difficult to obtain in the real world, make it easier to check the basic functionality of the planner.

4.3.2 Stability Analysis

It can be shown that t' is stable for a limited range of values assuming linear height surfaces. A natural property of our planner is that the footsteps converge towards the location of the CoM projected to the ground. In practice, this is not a desirable behavior since the feet would move too close to each other making the robot more sensitive to small disturbances. To account for this potential problem, we create a hybrid behavior—if the planned velocity at the next transition will be too small we artificially extend the swing time, which keeps the dynamics away from converging to the origin. In this section, we only consider the system's stability when this convergence effect is not present.

We begin by formulating the well known linear inverted pendulum model,

$$\ddot{x} = \frac{g}{h}(x - p_x), \tag{4.26}$$

where g, h, and p_x are gravitational acceleration, the constant CoM height, and the stance foot location, respectively. The solution of this ODE is,

$$x(t) = p_x + (x_0 - p_x)\cosh(\omega t) + \frac{\dot{x}_0}{\omega}\sinh(\omega t), \tag{4.27}$$

where $\omega = \sqrt{g/h}$, and can be expressed as a discrete time state-space system with a constant step duration T:

$$X((k+1)T) = AX(kT) + Bx_{p,k} \quad k \in \mathbb{Z}, \tag{4.28}$$

where,

$$A = \begin{bmatrix} \cosh(\omega T) & \omega^{-1}\sinh(\omega T) \\ \omega\sinh(\omega T) & \cosh(\omega T) \end{bmatrix}, \tag{4.29}$$

$$B = \begin{bmatrix} 1 - \cosh(\omega T) \\ -\omega\sinh(\omega T) \end{bmatrix}. \tag{4.30}$$

Here, $X(kT) = \begin{bmatrix} x_{kT} & \dot{x}_{kT} \end{bmatrix}^T$ represents the state at the instant when the input changes—the instant the stance foot and swing foot change roles, and a new foot position p_x is put into place.

As explained previously, the planner chooses p_x such that the CoM velocity becomes zero at time t'. This can be expressed as a function of the state of the discrete system,

$$0 = \dot{x}_{kT+t'} = \begin{bmatrix} \omega\sinh(\omega t')\cosh(\omega t') \end{bmatrix} X(kT) - \omega\sinh(\omega t')x_{p,k}, \tag{4.31}$$

$$x_{p,k} = \begin{bmatrix} 1 & \omega^{-1}\coth(\omega t') \end{bmatrix} X(kT). \tag{4.32}$$

However we include an additional linear bias term in the control law above,

$$x_{p,k} = x_d\kappa_p + \begin{bmatrix}(1-\kappa_p) & \omega^{-1}\coth(\omega t')\end{bmatrix} X(kT), \tag{4.33}$$

to move the robot towards a goal location x_d, effectively creating a locomotion behavior.

Without loss of generality, we assume the goal is the origin for the stability analysis. The closed loop system under the proposed feedback law is:

$$X((k+1)T) = (A+BK)X(kT) \tag{4.34}$$
$$K = \begin{bmatrix}(1-\kappa_p) & \omega^{-1}\coth(\omega t')\end{bmatrix} \tag{4.35}$$
$$A + BK = A' \tag{4.36}$$
$$A'_{11} = 1 - \kappa_p + \kappa_p\cosh(\omega T) \tag{4.37}$$
$$A'_{12} = \omega^{-1}(\sinh(\omega T) + (1-\cosh(\omega T))\coth(\omega t')) \tag{4.38}$$
$$A'_{21} = \kappa_p\omega\sinh(\omega T) \tag{4.39}$$
$$A'_{22} = \cosh(\omega T) - \sinh(\omega T)\coth(\omega t') \tag{4.40}$$

Using the eigenvalues of the matrix A', we determine the stability of the closed loop system. To be robust to model uncertainties, we chose parameters κ_p and t' such that they produce eigenvalues with magnitudes close to 0.8 for the desired h and T values. One interesting fact of t' is that an infinitely large t' stops the robot's CoM exactly at the stance foot position, which is equivalent to CP [75]. However, this is not a desired behavior because when the CoM is near the stance leg, crossing of the legs might occur. Thus, we design the planner parameters to gradually decrease the robot's CoM velocity at the moment when the stance leg switches to the other leg. We achieve this decrease in velocity by setting the magnitudes of the eigenvalues of A' close to 0.8.

4.3.3 Impact Model

In many cases, the hybrid dynamics of a robot impacting the ground are significant. This warrants a model which includes a discrete map to represent the sudden velocity changes of impact [23]. However, since our planning algorithm focuses exclusively on the CoM, and since the mechanical design of our robot allows body mass to dominate the leg mass, we have used a model which predicts no change to CoM velocity on impact. Given the other uncertainties in our model, we find more complex impact models difficult to justify.

Consider the following conservative simplification: suppose the swing leg's mass were concentrated at the point of the foot, as shown in Fig. 4.15. Suppose as well that the foot, rather than

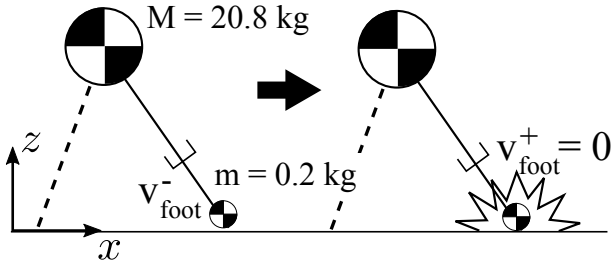


Figure 4.15: **Impact Model** Simplified model of a multi-limb system to represent impact situation at landing. To include leg mass, mass m is attached at the end of a leg.

slowing down at the end of the stride, were to impact at the average foot velocity over the entire 0.15 m, 0.5 s step. The momentum of that foot would be 0.86 kg m/s, and the change in CoM velocity would be 0.041 m/s if the foot suddenly stopped moving (due to the 20.8 kg total robot mass). If we consider the foot mass to be only the mass of the shank then the CoM velocity change would be only 0.0029 m/s. Thus we have not been able to justify a more complex CoM velocity impact model than the identity map.

Chapter 5

Real-Time Feedback Control

In this chapter, we address various issues coming up with the implementation of real-time feedback controllers on actual bipedal systems. Section 5.1 explores issues related to bandwidth cross-coupling in cascaded feedback controllers and explains how to properly address them. We utilized online feedback gain adjustments and sensor-based internal force feedback control. The WBC used in this study is the whole-body operational space controller (WBOSC) presented in Section 2.1. Section 5.2 explores the use of WBOSC on a NAO bipedal robot. The implementation of closed-loop WBOSC in NAO is challenging because of various performance limitations and access to the code on these kind of toy-size humanoid robots. We circumvented these limitations by devising a virtual model based controller. Section 5.3 and 5.4 explore torque/ position controllers tested on the robot Draco P1, a robot testbed using viscoelastic liquid cooled actuators. Draco P1 has a new type of series elastic actuators relying on elastomers instead of metal springs for the drivetrain. We demonstrate high-performance impedance control as well as high-stiffness position control with this kind of system.

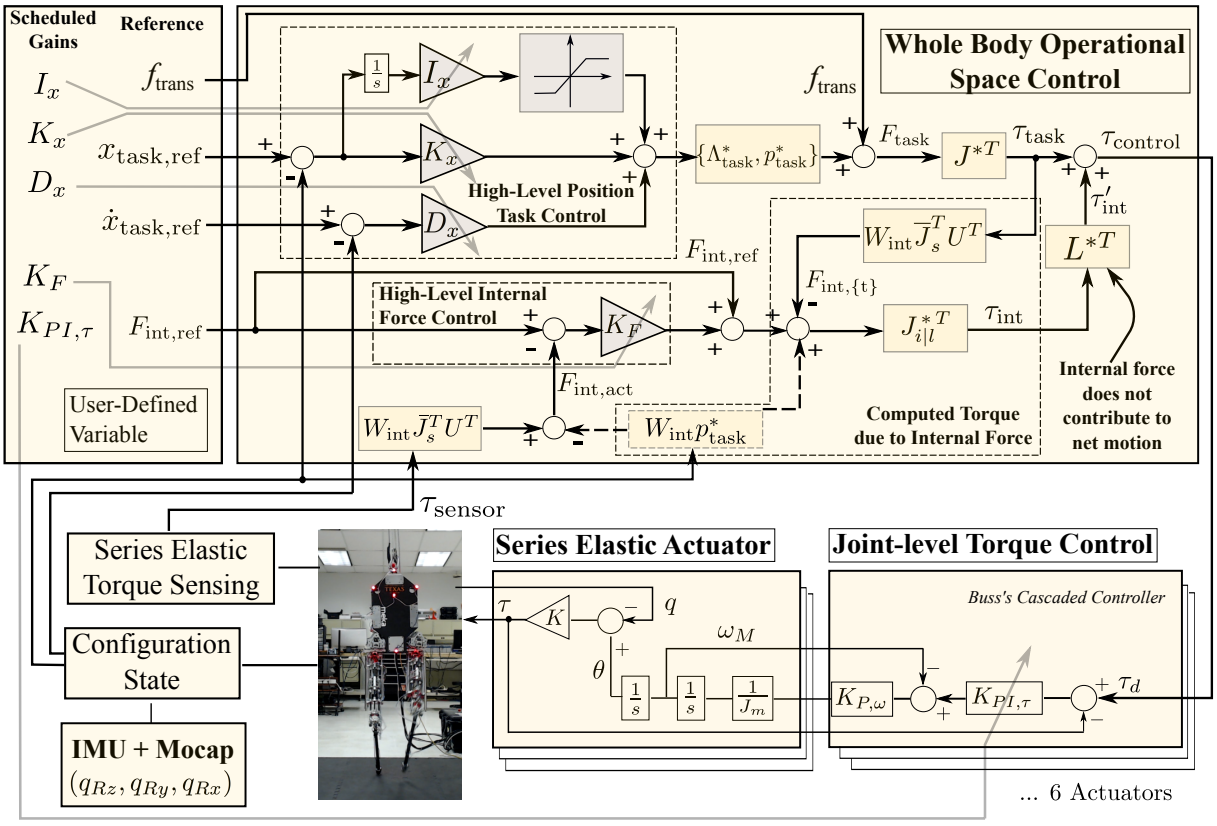


Figure 5.1: **Overall Control Diagram.** This figure illustrates WBOSC and the joint-level torque controllers used in the experiment presented in Section 7.3. One of our main contributions comes from the feedback control of internal forces. Note that the gains for the controllers are treated as additional input parameters to represent the gain scheduling for achieving the best possible performance of each task.

5.1 Bandwidth of Cascaded Feedback Controllers

The feedback control system of the Hume bipedal robot is split into six joint level controllers and a centralized high level controller (see Fig. 5.1). The purpose of the joint level controllers is to achieve good torque tracking given the series elastic actuators. This type of control architecture falls into the category of a distributed control system which allows the joint controllers to focus on high speed actuator dynamics while the centralized controller does not need to deal with this nuance. Yet the feedback at the high level is necessary in order to create the coupling between joints implied by operational space impedance tasks as well as regulating the internal forces between multicontact supports.

5.1.1 Online Feedback Gain Adjustment

To enhance the bandwidth of the proposed WBOSC controller (shown in the upper portion of Fig. 5.1), we lowered the gains of the torque feedback controllers (shown in the lower portion of the same figure). A detailed study on stability and bandwidth trade-off between high-level position control and low-level torque control is presented in [112]. In particular, reducing the low-level torque controller bandwidth allows us to increase the high-level position controller bandwidth without compromising stability. To achieve a higher bandwidth with position control, we increase the PID feedback gains in WBOSC, reduce the joint-level torque feedback gains, and set the integral gains on the low-level torque controllers. Detuning torque gains would reduce the accuracy of internal force tracking performance if we would solely rely on feedforward models – see discussion in Section 5.1.2. For this reason we design internal force feedback controllers to precisely track that subset of the contact forces – this technique was also discussed in Section 5.1.2.

One drawback of detuning torque controllers is that friction and stiction cause torque errors.

This chapter contains material from the following publications:
[43] Donghyun Kim, Junhyeok Ahn, Orion Campbell, Nicholas Paine, and Luis Sentis. Investigations of a Robotic Testbed with Viscoelastic Liquid Cooled Actuators. arXiv.org, November 2017.
[46] Donghyun Kim, Steven Jens Jorgensen, Peter Stone, and Luis Sentis. Dynamic behaviors on the NAO robot with closed-loop whole body operational space control. In 16th International Conference on Humanoid Robots (Humanoids), pages 11211128. IEEE, 2016.
[49] Donghyun Kim, Ye Zhao, Gray Thomas, Benito R Fernandez, and Luis Sentis. Stabilizing Series-Elastic Point-Foot Bipeds Using Whole-Body Operational Space Control. IEEE Transactions on Robotics, 32(6):13621379, 2016.
My role covers algorithm development, programming, experiment, and the major portion of writing. Co-authors have helped hardware experiments presented in Section. 5.3 and Section. 5.4.

This effect results in joints not moving when torque commands are small compared to the stiction threshold. To tackle this problem, we simultaneously adjust the torque controller gains and the WBOSC feedback controller gains based on each robot joint's effective load: in joints belonging to the stance leg, we turn off the integral gain of the torque controller. In joints belonging to the swing leg, we turn on the integral gain of the same controller to reduce the effects of friction in the actuators.

This can be counter-intuitive since larger effective mass implies larger feedback gain. Intuitively, the natural frequency of joint force output in SEA actuators decreases as the joint's effective load increases [66]. When one of the robot's legs is in contact with the ground, its effective mass increases and as a result its natural force frequency decreases. Applying integral torque gain in a controller with a small natural frequency can reduce the phase margin due to an increase in bandwidth and a drop in phase by $-90°$. After various trials, we concluded that a proportional feedback control with motor velocity feedback in the torque controller and a PID control in WBOSC gives the best performance for the stance leg.

5.1.2 Sensor-Based Internal Force Feedback Control

Internal forces are associated with joint torques that produce no net motion. As such, internal forces correspond to mutually canceling forces and moments between pairs or groups of contact points, i.e. tensions, compressions and reaction moments. For instance, a triped point-foot robot has three internal force dimensions while a biped point-foot robot has a single internal force dimension as shown in Figure 5.2.

Internal forces are fully controllable since they are orthogonal to the robot's motion. As such, both the robot's movements and its internal forces can be simultaneously controlled. Moreover, in many types of contact poses, internal forces are easily identifiable using some physical intuition. For instance, in the triped pose of Figure 5.2 the three feet can generate three virtual tensions between the points of contact. The physics of tension forces were analyzed in greater detail using a virtual linkage model in [88].

Internal forces are part of the core WBOSC. In Section 2.1 , we describe the model-based control structures enabling direct control of internal forces. In particular, the basic torque structure derived in Eq. (2.18) is copied here:

$$\boldsymbol{\tau}_{\text{int}} \;=\; \boldsymbol{J}_{i|l}^{*\top}\Big(\mathbf{F}_{\text{int,ref}} - \mathbf{F}_{\text{int},\{t\}} + \boldsymbol{\mu}_i^* + \mathbf{p}_i^*\Big), \tag{5.1}$$

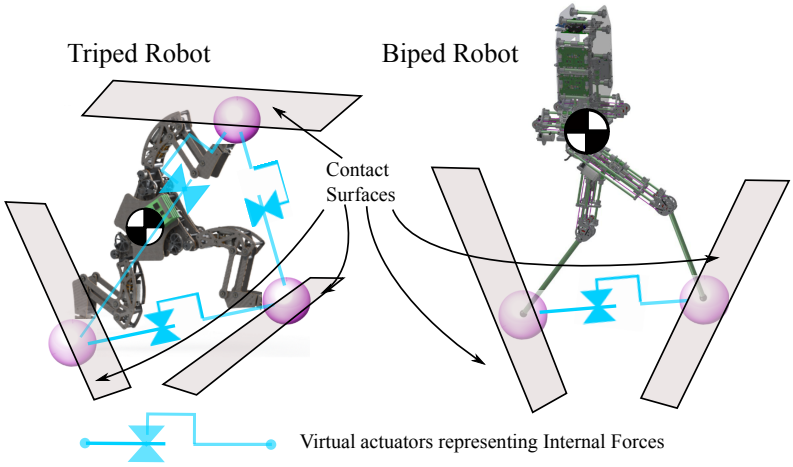


Figure 5.2: **Illustration of Internal Forces for Various Robots.** Internal forces in point-foot robots correspond to tensions or compressions between pairs of supporting contacts.

where $\mathbf{F}_{\text{int,ref}}$ is the vector of desired internal forces and $\mathbf{F}_{\text{int},\{t\}}$ corresponds to the mapping of task torques into the internal force manifold. The above equation would be sufficient for feedforward internal force control if the commanded torques were equal to the actual torques, and if the kinematic and dynamic models were exact. However, as we mentioned in Section 5.1.1, we intentionally lower the bandwidth of joint torque controllers in order to increase the bandwidth of WBOSC's task motion controllers. As a result, we cannot solely rely on feedforward control of internal forces. Formulating internal force feedback control is a significant advantage over increasing the bandwidth of joint torque controller. It enables good force tracking accuracy but only in the dimensions that are rendered of interest to the behaviors, which in our case correspond to internal forces. Because internal forces do not affect task motion, the internal force feedback controller does not negatively affect the accuracy of the motion tasks as the joint torque controllers do. Overall, it is best to formulate feedback control only in the dimensions that are practical, and in our case they are the motion tasks in one hand, and the internal force task on the other. To our knowledge, this is the first successful use of sensor-based feedback control of internal forces in a real robot.

We can achieve better internal force tracking accuracy by incorporating a simple proportional controller of the measured internal force error via Eq. (2.18),

$$\boldsymbol{\tau}_{\text{int}} = \boldsymbol{J}_{i|l}^{*\top}\Big(\mathbf{F}_{\text{int,ref}} - \mathbf{F}_{\text{int},\{t\}} + \boldsymbol{\mu}_i^* + \mathbf{p}_i^* + K_F\left(\mathbf{F}_{\text{int,ref}} - \mathbf{F}_{\text{int,act}}\right)\Big), \tag{5.2}$$

where K_F is a proportional control gain and $\mathbf{F}_{\text{int,act}}$ are the actual sensor-based internal forces.

To obtain these sensor-based forces, we use the torque sensors on the series elastic actuators to find the reaction forces as per Eq. (2.5) and apply a projection $\boldsymbol{W}_{\text{int}}$ to find internal forces,

$$\mathbf{F}_{\text{int,act}} \triangleq \boldsymbol{W}_{\text{int}} \left[\overline{\boldsymbol{J}}_s^{\top} (\boldsymbol{U}^{\top} \boldsymbol{\tau}_{\text{sensor}} - \mathbf{b} - \mathbf{g}) + \boldsymbol{\Lambda}_s \dot{\boldsymbol{J}} \dot{\mathbf{q}} \right], \tag{5.3}$$

where $\boldsymbol{\tau}_{\text{sensors}}$ corresponds to the vector of torques sensed by the spring element in each series elastic actuator (see Figure 5.1).

The above internal force mapping is distinguished from previous work due to its sensor-based force feedback nature, and its mapping is valid due to the physical fact of robot redundancy in the multi-contact case. The induced contact closed loop causes the number of controlled motion tasks to be smaller than that of actuated joints. Correspondingly, additional DOFs are available to be controlled for more force tasks, such as internal forces in Eq. (5.3). This mapped internal forces are consistent with contact constraints and cancellation of accelerations on the robot's base or on the actuated joints [88]. More details can be found in Section 2.1.

To calculate internal forces for Hume we need to define the mapping given in Eq. (2.16) in Section 2.1, where $\boldsymbol{W}_{\text{int}}$ is the matrix representing the map from reaction forces to internal forces. In our case, Hume controls the internal forces between the two feet during dual contact phases. In dual support mode, the reaction forces are $(f_{Rx},\ f_{Ry},\ f_{Rz},\ f_{Lx},\ f_{Ly},\ f_{Lz})^{\top}$, where R and L mean the robot's right and left foot, respectively. According to [88], $\boldsymbol{W}_{\text{int}}$ consists of $\boldsymbol{S}_t$, a selection matrix of tensions, $\boldsymbol{R}_t$, a rotation matrix from global frame to the direction parallel to the line between two contact points, and $\boldsymbol{\Delta}_t$, a differential operator matrix, i.e.

$$\boldsymbol{W}_{\text{int}} = \boldsymbol{S}_t \, \boldsymbol{R}_t \, \boldsymbol{\Delta}_t, \tag{5.4}$$

with

$$\boldsymbol{S}_t = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \tag{5.5}$$

$$\boldsymbol{R}_t = \begin{bmatrix} \hat{\mathbf{x}}^{\top} \\ \hat{\mathbf{y}}^{\top} \\ \hat{\mathbf{z}}^{\top} \end{bmatrix}, \quad \begin{cases} \hat{\mathbf{x}} = \frac{\mathbf{P}_R - \mathbf{P}_L}{\|\mathbf{P}_R - \mathbf{P}_L\|} \\ \hat{\mathbf{y}} = (-\hat{\mathbf{x}}(2), \hat{\mathbf{x}}(1), 0)^{\top} \\ \hat{\mathbf{z}} = \hat{\mathbf{x}} \times \hat{\mathbf{y}} \end{cases}, \tag{5.6}$$

$$\boldsymbol{\Delta}_t = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & -\boldsymbol{I}_{3\times 3} \end{bmatrix}. \tag{5.7}$$

where $\mathbf{P}_R$ and $\mathbf{P}_L$ are the position of the right and left feet, respectively.

There are two ways to compute desired internal forces: 1) by approximating them via a simple force statics problem and choosing values that comply with friction cones, and 2) by first

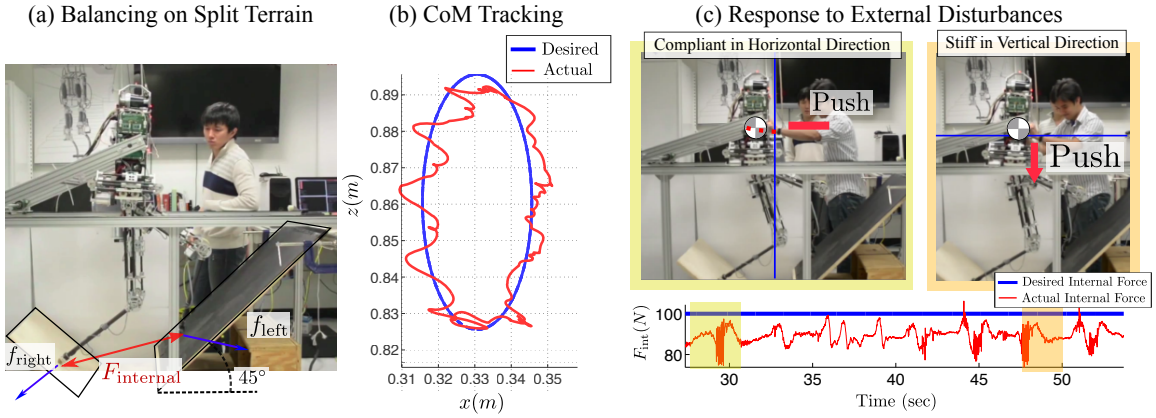


Figure 5.3: **Trajectory Tracking and Human Disturbance Rejection on Disjointed Terrain:** Subfigure (a) shows Hume standing between two inclined wooden panels and tracking a position task with its center of mass. This position set-point follows a constant velocity trajectory along an elliptical path shown, along with the measured CoM path, in Subfigure (b). In Subfigure (c) the CoM has different impedances in the horizontal and vertical directions. When the robot is pushed backwards it moves as though the center of mass were connected to a low-spring-constant spring, whereas when the robot is pushed downwards it reacts as though connected to its set point by a far stiffer spring. Due to the feedback regulation of internal forces, the biped does not fall down when disturbed with large external forces.

solving an optimization problem including task accelerations and friction cones to obtain desired reaction forces, $\mathbf{F}_{r,\text{ref}}$, and then projecting them into internal forces using Eq. (2.16), i.e.

$$\mathbf{F}_{\text{int,ref}} = \boldsymbol{W}_{\text{int}}\, \mathbf{F}_{r,\text{ref}}. \tag{5.8}$$

We use method 1). If one wants to use method 2), she/he can use existing methods such as the QP-based contact solver defined in [95] or the centroidal-momentum-based contact solver defined in [6]. Then the equation above can be used to solve for the desired internal forces to be subsequently used as inputs to WBOSC.

To verify the proposed internal force controller, we conducted the experiment that Hume balances on a high pitch terrain composed of two 45° wedges angled in towards the robot to create a convex floor profile. Because there is no way to control lateral motion, the planarizer is used to constrain the motion of Hume to the sagittal plane.

Note that we do not include the 5 kg sliding linkage in the dynamic model and regard it as unmodeled disturbance. The robot's tasks were to maintain a 100 N internal force pushing

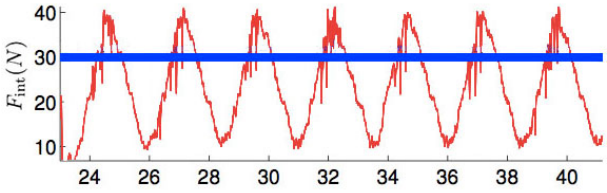


Figure 5.4: **Internal Force Control Without Feedback:** Without feedback loop, internal force control shows larger than 50% error.

outwards against the two contact points, a desired impedance task for the center of mass, and a desired impedance task for the body orientation. The reference of 100 N is roughly calculated with the assumption that the 200 N body weight is equally distributed between each foot and a 100 N horizontal force generates a reaction force normal to the 45° angled surfaces.

By controlling the internal force, Hume does not slip while it tracks the reference CoM path within a 2 cm error. This experiment is divided into two sub-experiments: Hume was controlled to track a time-varying CoM trajectory which followed an elliptical path in the sagittal plane, as shown in Fig. 5.3(b); and Hume was controlled to hold a Cartesian impedance task on the CoM which had low stiffness horizontally and high stiffness vertically as shown in Fig. 5.3(c). In the test, $\mathbf{x}_{\text{task}} = [\text{CoM}_x,\ \text{CoM}_z,\ q_{Ry},\ q_{Rx}]^\top$, $\mathbf{F}_{\text{task,ref}} = [0]_{4\times 1}$ and $F_{\text{int,ref}} = 100\,\text{N}$, using the notation of Fig. 5.1. We do not control CoM_y and q_{Rz} since the planarizer rigidly constrains those directions of motion. However, we do control the robot's roll, q_{Rx}, since the planarizer is slightly flexible in the roll direction. Thus, controlling the roll and pitch helps to sustain the body pose.

CoM tracking performance is shown in Fig. 5.3(b). Although tracking errors exist they are bounded within 2 cm. We believe this error occurs because of various reasons: 1) the legs of Hume are flexible and that flexibility is not modeled, 2) the planarizer that holds Hume on the back is unmodeled, and 3) the low cost IMU that we use suffers from quick orientation drift. The second sub experiment shows that feedback control of internal forces allows Hume to stay balanced despite external forces exerted by a human. Due to the WBOSC's internal force feedback controller, the disturbances we exert on the robot do not produce large deviations in the internal force tracking performance.

Due to feedback control, the errors between desired (blue) and actual (red) internal forces are small enough to keep Hume balance on the difficult terrain. In contrast, in Fig. 5.4 we turned off the feedback control of internal forces. The experiment was conducted on a 20° pitch terrain

due to the robot being unable to stay up on the 45° pitch terrain when turning off the feedback. We can see that the error to command ratio is much larger when using feedforward control only (more than 50 %) relative to the experiment with the feedback enabled (less than 20 %). These experiments ultimately show the effectiveness of closed loop internal force control as applied to maintaining a frictional contact.

5.2 Virtual Model Control

Implementation of WBOSC on the NAO robot is challenging since it is not designed for real-time feedback control. There are four major difficulties. The NAO platform has (1) low computational power, (2) low-quality sensing capability, (3) limited embedded controller access, and (4) particular hardware deficiencies. First the CPU in the NAO robot is a dual core Intel Atom(TM) @ 1.60Ghz. This is a low-end computer compared to other humanoid robots that utilize computationally expensive whole-body controllers. The low computational power limits the types of estimation techniques that can be utilized. For example high dimensional Kalman filters are not possible. Second, NAO's low sensing capability gives low quality velocity data. It is delayed, biased, and noisy. Moreover, the slow control servo rate (10ms) significantly reduce the phase margin and limit the bandwidth of the feedback controller. Furthermore, the quality of the velocity data is critical with stabilizing a feedback system since velocity feedback can be used to damp the system dynamics and provide passivity [65].

Third, in a cascaded control structure, having the ability to manipulate the joint-level controller is crucial to tuning the WBOSC feedback controller. Since NAO only provides a joint position controller that accepts only two parameters (desired joint position and stiffness), this tuning flexibility is no longer available. Additionally other types of low-level controllers such as a joint-level torque controller are not possible. It is important to remember that WBOSC returns joint torque outputs to accomplish the specified tasks. Thus, we must convert joint torque commands to appropriate joint position commands. Finally, while it is possible to imitate a torque controller by reverse-engineering the joint-position controllers and identifying the appropriate inverse function to create a mapping between desired torque outputs and joint position commands, this is not possible due to the limited information on the embedded controllers and NAO's hardware deficiencies. NAO's spur gears in the drive train are not appropriate for torque control due to inherent friction, stiction, and backlash.

5.2.1 Implementing WBOSC on the NAO

To address the above hardware limitations, we provide the following methodologies to lower computational burden, obtain clean velocity data, and send the appropriate joint position commands from the WBOSC torque commands.

To lower computational burden, we first note that using our WBOSC framework is computationally faster due to its projection-based methods and closed-form solution. Secondly, on the NAO system itself, while the WBOSC is computed on the main control loop, the mass matrix is computed on a different thread and is only occasionally updated. Note that while the configuration-dependent mass matrix is slightly behind temporally to the true mass matrix, our empirical tests have shown that this difference is small and negligible even with a 10ms system control loop.

To obtain clean velocity data and send appropriate joint position commands, we create a virtual robot model which simulates the joint accelerations the NAO robot will experience if the links exert the desired torque commands from the WBOSC. The virtual robot model takes the joint accelerations and through integration, gives the joint velocities and positions, which are sent to the joint position controllers and the orientations are sent to the orientation estimator. Clean velocity data is obtained from the joint velocity data in the virtual robot model and is used to stabilize the system dynamics.

5.2.2 WBOSC Control Scheme for the NAO Robot

The virtual robot model consist of forward dynamics and numerical integration (Fig. 5.5). Note that this virtual model is what the WBOSC controls. The virtual robot model is synchronized with the actual robot via two methods. First commanding the desired joint positions of the virtual robot model to the the built-in joint position controllers synchronize the joint states. Second, the orientation estimator estimates the real robot's body orientation by combining the measured angular velocity data and its kinematics. Finally, using another Kalman-filter, we also estimate the robot's yaw velocity. In this way, WBOSC performs closed-loop control with the actual robot.

The numerical integration process requires small step sizes to avoid truncation errors. With a 10 ms system control loop rate, the numeric integrator is not stable on its own. Since this process is not computationally expensive, we run the integration step 70 times in one system control loop to further reduce the step size. By doing so, we enhance the accuracy of numeric integration. To enhance the computational efficiency, we do not include gravity and Coriolis

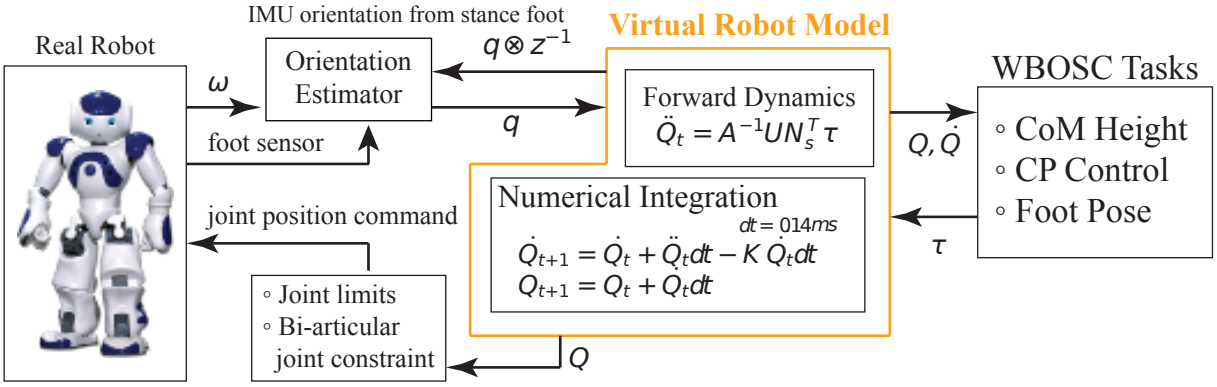


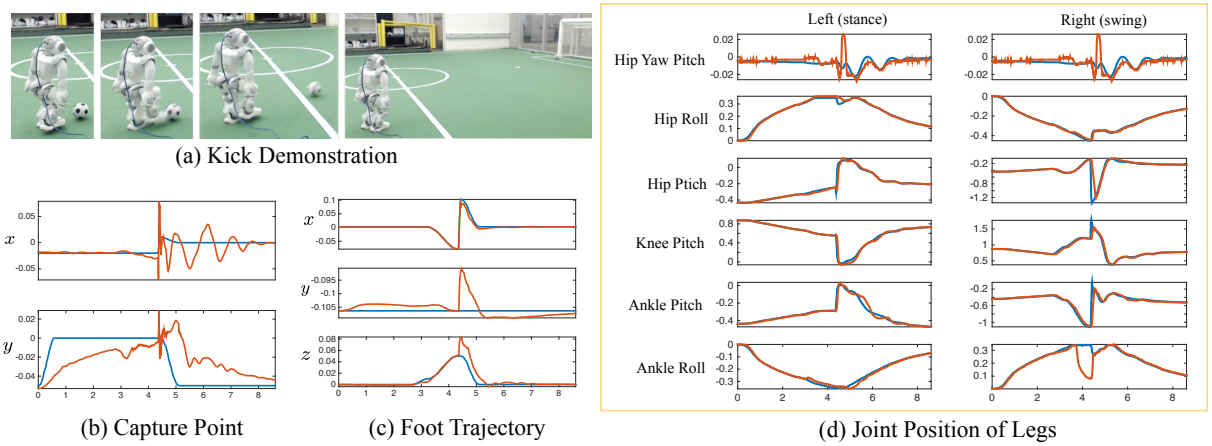
Figure 5.5: **Total Control Scheme of NAO.** The virtual robot model plays an important role in the feedback control. In the figure, ω is the angular velocity data from the IMU, q is the orientation of the IMU, z is the orientation of a stance foot, Q is the full state of the dynamic model including virtual joints, and τ is the torque command from the WBOSC. In the virtual robot model, the forward dynamics convert the torque command to joint accelerations, and through integrations using dt as the time step, Q is delivered to the real robot after checking hardware constraints. Note that dt is $\frac{1}{7}$ms since it runs 70 times in one system control loop from the Real robot to the WBOSC.

terms in both the WBOSC and forward dynamics. Specifically, these terms are canceled out when performing inverse dynamics in the WBOSC and forward dynamics in the virtual robot model. We also remove the neck joints and elbow joints in the virtual model since they are not actively used in the kicking demonstration. Instead, the joint position commands for these joints are fixed and WBOSC sees them as fixed masses.

5.2.3 Dynamic Kick Demonstration

To test the proposed controller's performance, we demonstrate a dynamic kicking behavior in the NAO robot. The WBOSC tasks involved in the kicking motion, in order of priority from high to low, are to: (i) maintain the CoM height, (ii) perform CP control, and (iii) do pose control on the swing foot. The desired CoM height is set to the starting height of the NAO robot after it performs its boot up routine, and the CP and foot trajectories are designed with a sinusoidal function. The ball used is the 2016 black and white competition ball used in SPL, which is 45g.

The result is presented in Fig. 5.6. The ball traveled 5m away. Empirical tests show that a similar performance is obtained in multiple trials. We note that due to the slow update rate of



(a) Kick Demonstration

(b) Capture Point

(c) Foot Trajectory

(d) Joint Position of Legs

Figure 5.6: **Experiment Result.** x, y, and z mean saggittal, lateral, and vertical direction, respectively. In (a) the NAO performs a dynamic kick from the half-way line of the field in which the ball traveled 5m away. In (b) and (c), the blue lines are the desired CP and foot trajectories and the orange lines are the measured trajectories. In (d), the joint position data shows that the real robot's configuration (orange) is well matched with the virtual model configuration (blue)

the NAO robot (10 ms), we cannot use high stiffness gains in the balance controller. The lack of high-stiffness control causes the lateral direction of the capture point to have noticeable error. However, the controller still manages to follow the desired trajectories. Overall, Fig. 5.6 shows that we have successful performed a dynamic motion using approach described in Sections 5.2.1 and 5.2.2.

5.3 Force Control using Disturbance Observers

From our previous study of point-foot biped robots with series elastic actuators (SEAs) [49], we have learned that SEAs effectively protect drivetrains and motors from damage due to external impacts. Our group have devised our own SEAs, which are more compact, energy efficient, and powerful than other SEAs available today [69]. One drawback of SEAs is the difficulty that arises when using a position controller due to the elastic element in the drivetrain. In this section, we introduce a novel SEA using a viscoelastic material instead of a metal spring. In addition to changing the material of the elastic component, we replaced traditional air-convection cooling with liquid cooling, which significantly enhances the power of the electromagnetic (EM) motor. These two major changes successfully address the issues of position control and power density. However, much of the innovation lies in effectively integrating these new ideas into the

existing technology while preserving the original beneficial features of the SEA design.

The challenge in using elastomers comes from the complex force versus displacement characteristics. Existing studies of elastomer-based actuators also discussed the difficulty of force control. [40] experimented with a damped SEA based on a piezoelectric damper. Although it had a metal spring with controllable dampers, the results showed the beneficial features of dampers for dynamic performance, stability, and controllability of SEAs. [1] showed reasonably good estimated torque tracking and no steady-state errors. [3] studied the use of rubber for the viscoelastic material in a SEA. They model the force-displacement curve of rubber using a "standard linear model." The estimated rubber force is employed in a closed-loop force controller. Unfortunately, the hysteresis in the urethane rubber destabilized the system at frequencies above 2 Hz. Previous studies imply the feasibility of torque control based on the deflection measurement of viscoelastic elements, but the performance is degraded due to material hysteresis.

To achieve stable and robust force control, we study various feedback control schemes. In our previous work, we have shown that the active passivity obtained from motor velocity feedback [49] and model-based control such as disturbance observer (DOB) [67] play an essential role in achieving high-fidelity force feedback control. We analyze the phase margins of various feedback controllers and empirically show how these effects appear in the actual system. We verify the stability and robustness of our controller by demonstrating impedance control including an impact test.

5.3.1 Viscoelastic Liquid Cooled Actuator

The design objectives of the VLCA are 1) power density, 2) efficiency, 3) impact tolerance, 4) position controllability, and 5) force controllability. Our previous work [68] shows a significant improvement in motor current, torque, output power and system efficiency for liquid cooled commercial off-the-shelf (COTS) electric motors and studied several Maxon motors for comparison. As an extension of this previous work, in this new study we studied COTS motors and their thermal behavior models and selected the Maxon EC-max 40 brushless 120 W (Fig. 5.7(e)), with a custom housing designed for the liquid cooling system (Fig. 5.7(h)). The limit of continuous current increases by a factor of 3.59 when liquid convection is used for cooling the motor. Therefore, a continuous motor torque of 0.701 $\mathrm{N} \cdot \mathrm{m}$ is theoretically achievable. Energetically, this actuator is designed to achieve 366 W continuous power and 1098W short-term power output with an 85% ball screw efficiency (Fig. 5.7(b)) since short-term power is generally three time larger than continuous power. With the total actuator mass of 1.692 kg, this translates into a continuous power of 216W/kg and a short-term power of 650W/kg. By combining

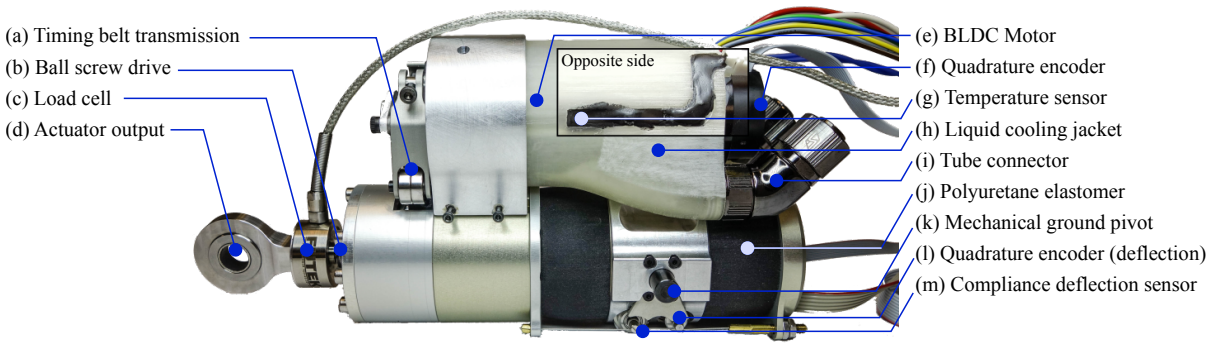


Figure 5.7: **Viscoelastic Liquid Cooled Actuator.** The labels are explanatory. In addition, the actuator contains five sensors: a load cell, a quadrature encoder for the electric motor, a temperature sensor, and two rubber deflection sensors. One of the rubber deflection sensors is absolute and the other one is a quadrature encoder. The quadrature encoder gives high quality velocity data of the rubber deflection.

convection liquid cooling, high power brushless DC (BLDC) motors, and a high-efficiency ball screw, we aim to surpass existing electric actuation technologies with COTS motors in terms of power density.

In terms of controls, a common problem with conventional SEAs is their lack of physical damping at their mechanical output. As a result, active damping must be provided from torque produced by the motor [31]. However, the presence of signal latency and derivative signal filtering limit the amount by which this active damping can be increased, resulting in SEA driven robots achieving only relatively low output impedances [111] and thus operating with limited position control accuracy and bandwidth. Our VLCA design incorporates damping directly into the compliant element itself, reducing the requirements placed on active damping efforts from the controller. The incorporation of passive damping aims to increase the output impedance while retaining compliance properties, resulting in higher position control bandwidth. The material properties we took into consideration will be introduced in Appendix C. The retention of a compliant element in the VLCA drive enables the measurement of actuator forces based on deflection. The inclusion of a load cell (Fig. 5.7(c)) on the actuators output serves as a redundant force sensor and is used to calibrate the force displacement characteristics of the viscoelastic element.

Mechanical power is transmitted when the motor turns a ball nut via a low-loss timing belt and pulley (Fig. 5.7 (a)), which causes a ball screw to apply a force to the actuator's output (Fig. 5.7(d)). This drivetrain is very efficient and our empirical verifications are presented in

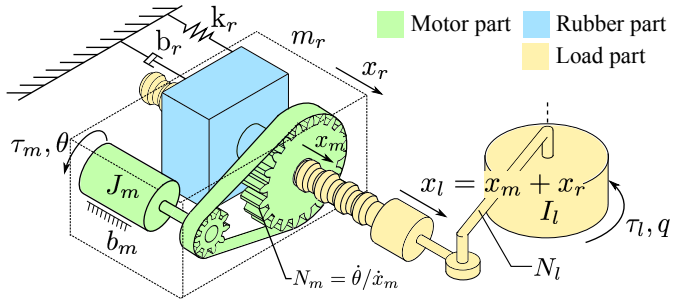


Figure 5.8: **Drivetrain structure.** Linear motion is changed to rotational motion through linkage, therefore, reduction ratio changes depending on configurations.

Appendix D. The rigid assembly consisting of the motor, ball screw, and ball nut connects in series to a compliant viscoelastic element (Fig. 5.7(j)), which connects to the mechanical ground of the actuator (Fig. 5.7(k)). When the actuator applies a force, the reaction force compresses the viscoelastic element. The viscoelastic element enables the actuator to be more shock tolerant than rigid actuators yet also maintain high output impedance due to the inherent damping in the elastomer.

5.3.2 Analysis of Torque Feedback Controllers

To demonstrate various impedance behaviors in operational space, robots must have a robust torque (or force) controller. Stable and robust operational space control (OSC) is not trivial to achieve because of the bandwidth cross-coupling issue between outer position feedback control (OSC) and inner torque feedback control [49]. Since stable torque control is a critical component for a successful OSC implementation, we comprehensively analyze various force feedback controllers.

The first step in this analysis is to identify the actuator dynamics. The transfer functions of the reaction force sensed in the series elastic actuators (rubber deflection) are well explained in [70]. The transfer function from motor torque to rubber deflection is

$$\frac{x_r}{\tau_m} = \frac{N_m^{-1} P_m P_r}{N_m^{-2} P_m + P_l + P_r}, \tag{5.9}$$

$I_m(\mathrm{kg\,m^2})$	$b_m(\mathrm{N\,m\,s})$	$m_r(\mathrm{kg})$	$\mathrm{b}_r(\mathrm{N\,s/m})$	$\mathrm{k}_r(\mathrm{N/m})$
3.5e−5	2.0e−4	1.3	2.0e4	5.5e6

Table 5.1: Actuator Parameters

where

$$\begin{aligned} P_m(s) &= \frac{1}{J_m s^2 + b_m s}, \\ P_r(s) &= \frac{1}{m_r s^2 + \mathrm{b}_r s + \mathrm{k}_r}, \\ P_l(s) &= \frac{1}{(I_l/N_l^2)s^2}. \end{aligned} \tag{5.10}$$

N_m is the speed ratio of the motor to the ball screw, and N_l is the length of the effective moment arm of the linkage at each joint. The equations follow the nomenclature in Fig. 5.8. When the actuator output is fixed, which is equivalent to $I_l = \infty$ ($P_l = 0$), the transfer function from the motor current input to the rubber deflection is given by

$$P_x = \frac{x_r}{i_m} = \frac{\eta k_\tau N_m}{(J_m N_m^2 + m_r)s^2 + (b_m N_m^2 + \mathrm{b}_r)s + \mathrm{k}_r}, \tag{5.11}$$

where η, k_τ, and i_m are the ball screw efficiency, the torque constant of a motor, and the current input for the motor, respectively. We can find η, k_τ, and N_m in data sheets, which are 0.9, 0.0448 $\mathrm{N \cdot m/A}$, and 3316 respectively. However, we need to experimentally identify k_r, b_r, J_m, and b_m. We infer k_r by dividing the force measurement from the load cell by the rubber deflection. The other parameters are estimated by comparing the frequency response of the model and experimental data. The frequency response test is done with the ankle actuator while prohibiting joint movement with a load and an offset force command. The results are presented in Fig. 5.9 with solid gray lines. Note that the dotted gray lines are the estimated response from the transfer function using the parameters of Table. 5.1. The estimated response and experimental result match closely with one another, implying that the parameters we found are close to the actual values.

We also study the frequency response for different load masses to understand how the dynamics changes as the joint moves. When 10kg is attached to the end of link, the reflected mass to the actuator varies from 1500kg to 2500kg because the length of the effective moment arm changes depending on joint position. In Fig. 5.9, the bode plots are presented and the response

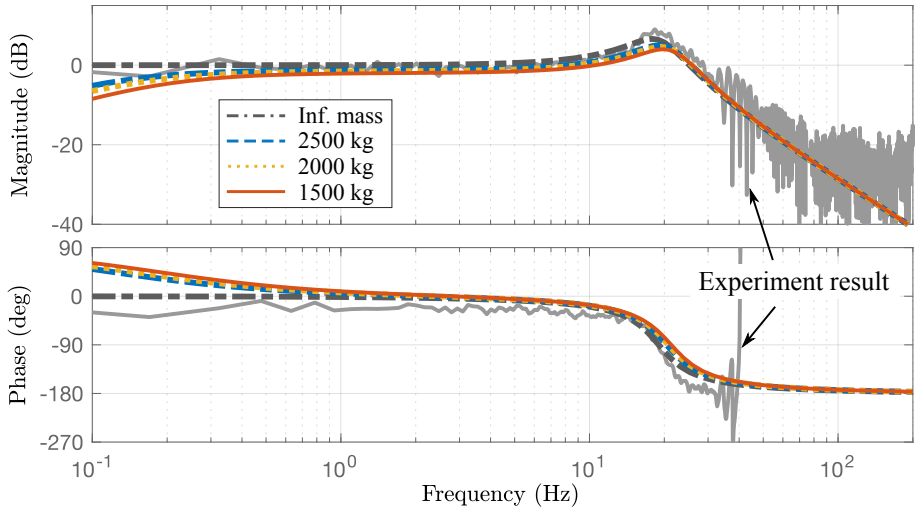


Figure 5.9: **Frequency response from the actuator force to the measured force for different loads.** Gray solid lines are experimental data and the other lines are estimated response with the model using empirically parameters.

is not significantly different than the fixed output case. Therefore, we design and analyze the feedback controller based on the fixed output dynamics.

For the force feedback controller, we first compare two options, which we have used in our previous studies [49, 67]:

1. Proportional (P) + Derivative (D_f) using velocity signal obtained by a low-pass derivative filter

2. Proportional (P) + Derivative (D_m) using motor velocity signal measured by a quadrature encoder connected to a motor axis

The second controller (PD_m) has benefits over the first one (PD_f) with respect to sensor signal quality. The velocity of motor is directly measured by a quadrature encoder rather than low-pass filtered rubber deflection data, which is relatively noisy and lagged. In addition, Fig. 5.10 shows that the phase margin of the second controller (47.6) is larger than the first one (17.1).

To remove the error at low frequencies, we consider two options: augmenting the controller either with integral control or with a DOB on the PD_m controller. To compare the two controllers, we analyzed the phase margins of all the mentioned controllers. First, we chose to

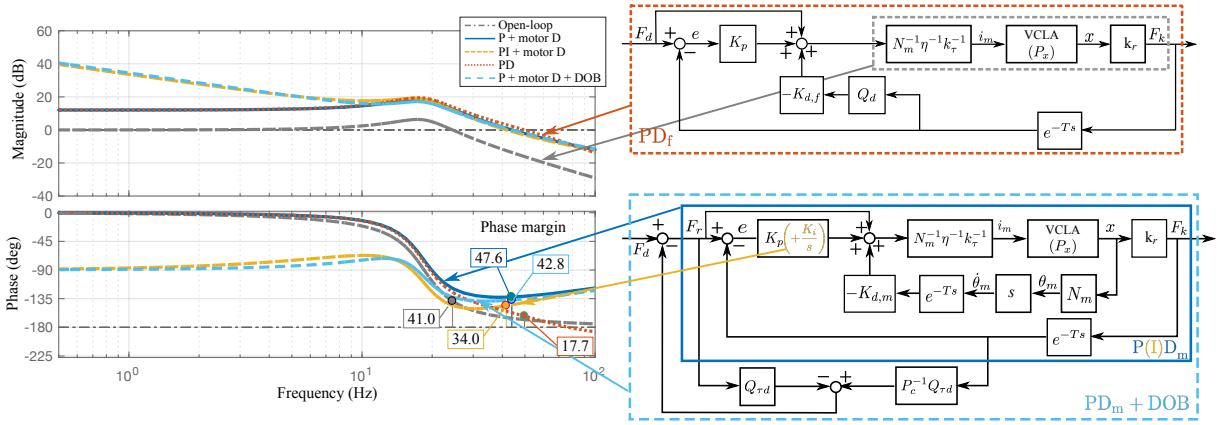


Figure 5.10: **Stability analysis of controllers.** Phase margins of each controllers and open-loop system are presented. Phase margins are $\mathrm{PD_m}(47.6°) > \mathrm{PD_m + DOB}(42.8°) > \mathrm{PID_m}(34.0°) > \mathrm{PD_f}(17.7°)$.

focus on the location where the sensor data returns in order to address the time delay of digital controllers (Fig. 5.10 (a) and (c)). Next, we have to compute the open-loop transfer function for each closed loop system. For example, the $\mathrm{PD_f}$ controller's closed loop transfer function is

$$F_k = \frac{\mathrm{k}_r P_x}{N}\left(k_p(F_r - e^{-Ts}F_k) + F_r - k_{d,f}Q_d e^{-Ts}F_k\right), \tag{5.12}$$

where T is the time delay and

$$Q_d = \frac{s}{(s/w_c)^2 + 1.414(s/w_c) + 1}, \tag{5.13}$$

which is a low pass derivative filter. For convenience, we use N instead of the multiplication of three terms, $\eta k_\tau N_m$. When gathering the term with e^{-Ts} of Eq. (5.12), we obtain

$$\frac{F_k}{F_r} = \frac{\mathrm{k}_r P_x(K_p + 1)/N}{1 + e^{-Ts}\mathrm{k}_r P_x(K_p + K_{d,f}Q_d)/N}. \tag{5.14}$$

Then, the open-loop transfer function of the closed system with the time delay is

$$P^{open}_{\mathrm{PD_f}} = \mathrm{k}_r P_x(K_p + K_{d,f}Q_d)/N. \tag{5.15}$$

We can apply the same method for the $\mathrm{PID_m}$ and $\mathrm{PD_m + DOB}$ controllers.

The transfer function of $\mathrm{PID_m}$, which is presented in Fig. 5.10(c), is

$$F_k = \frac{\mathrm{k}_r P_x}{N}\left((F_r - e^{-Ts}F_k)(K_p + K_i\frac{1}{s}) + F_r - K_{d,m}e^{-Ts}sN_m\frac{F_k}{\mathrm{k}_r}\right). \tag{5.16}$$

Then it becomes

$$\frac{F_k}{F_r} = \frac{\mathrm{k}_r P_x(K_p + K_i/s + 1)/N}{1 + e^{-Ts} P_x(\mathrm{k}_r(K_p + K_i/s) + K_d s N_m)/N}. \tag{5.17}$$

When we apply a DOB instead of integral control, we need the inverse of the plant. In our case, the plant of the DOB is $\mathrm{PD_m}$, which is similar to Eq. (5.17) except that K_i and e^{-Ts} are omitted:

$$P_{\mathrm{PD_m}}(= P_c) = \frac{\mathrm{k}_r P_x(K_p + 1)}{N + P_x(\mathrm{k}_r K_p + K_{d,m} s N_m)}. \tag{5.18}$$

The formulation of $\mathrm{PD_m}$ including the DOB, which is shown in Fig. 5.10(c), is

$$F_k = \frac{\mathrm{k}_r P_x(K_p + 1)}{N + e^{-Ts} P_x(\mathrm{k}_r K_p + K_{d,m} s N_m)} \frac{F_d - e^{-Ts} P_c^{-1} Q_{\tau d} F_k}{1 - Q_{\tau d}}, \tag{5.19}$$

where $Q_{\tau d}$ is a second order low-pass filter. Then the transfer function is

$$\frac{F_k}{F_d} = \frac{\mathrm{k}_r P_x(K_p + 1)}{N(1 - Q_{\tau d}) + e^{-Ts}\left(N Q_{\tau d} + P_x(\mathrm{k}_r K_p + K_{d,m} s N_m)\right)}. \tag{5.20}$$

The open-loop transfer function is

$$P^{open}_{\mathrm{PD_m+DOB}} = \frac{N Q_{\tau d} + P_x(\mathrm{k}_r K_p + K_{d,m} s N_m)}{N(1 - Q_{\tau d})} \tag{5.21}$$

The bode plots of $P^{open}_{\mathrm{PD_f}}$, $P^{open}_{\mathrm{PD_m}}$, $P^{open}_{\mathrm{PID_m}}$, and $P^{open}_{\mathrm{PD_m+DOB}}$ are presented in Fig. 5.10(b). The gains (K_p, $K_{d,m}$, K_i) are the same as the values that we use in the experiments presented in Section 5.3.4, which are 4, 15, and 300, respectively. The $\mathrm{PD_f}$ controller uses $K_d N_m/\mathrm{k}_r$ for $K_{d,f}$ to normalize the derivative gain. The cutoff frequency of the DOB is set to 15Hz because this is where the $\mathrm{PD_m} + \mathrm{DOB}$ shows a magnitude trend similar to the integral controller ($\mathrm{PID_m}$). The results imply that the $\mathrm{PD_m} + \mathrm{DOB}$ controller is more stable than $\mathrm{PID_m}$ with respect to phase margin and maximum phase lag. This analysis is also experimentally verified in Section 5.3.4.

5.3.3 Single Leg Testbed

As a pilot test for a future full biped robot, we built a single leg testbed presented in Fig. 5.11. To demonstrate dynamic motion, we implemented an operational space controller (OSC) incorporating the multi-body dynamics of the robot.

We designed and built a single leg testbed (Fig. 5.11) consisting of two VLCAs - one for the ankle (q_0) and one for the knee (q_1). The design constrains motion to the sagittal plane, the leg carries 10kg, 23.0kg, or 32.5kg of weight at the hip, and the foot is fixed on the ground. With

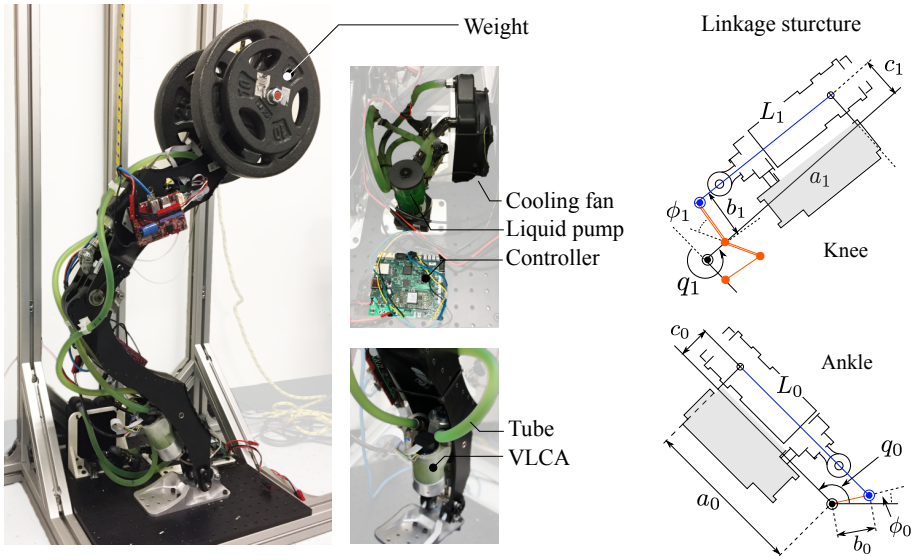
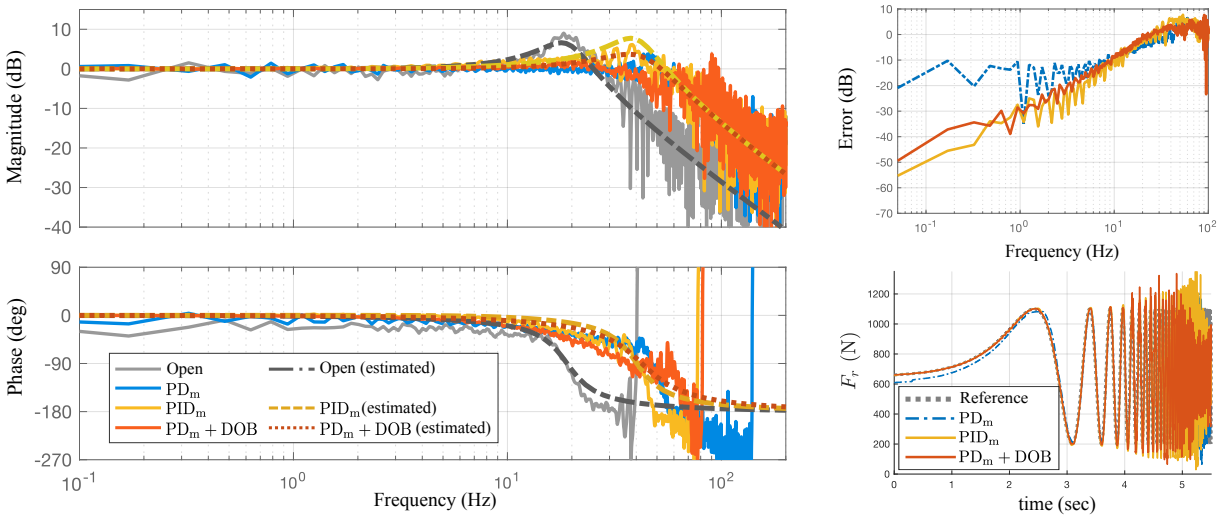


Figure 5.11: **Single leg testbed.** Our testbed consists of two VLCAs at the ankle and the knee. The foot of the testbed is fixed on the ground. The linkages are designed to vary the maximum peak torques and velocities depending on postures. As the joint positions change, the ratios between ball crew velocities ($\dot{L}_{0,1}$) and joint velocities ($\dot{q}_{0,1}$) also change because of effective lengths of moment arms vary. The linkages are designed to exert more torque when the leg crouches, which is the posture that the gravitational loads on the joints are large.



(a) Frequency response

(b) Error magnitude and chirp test trj

Figure 5.12: **Chirp signal test result of open-loop and three different controllers.** (a) Frequency response. Experimental data and estimated response based on the transfer functions are presented. Estimated response of PD controller is identical to the PD+DOB since DOB theoretically does not change the transfer function. The plot show PD+DOB shows better performance in terms of less overshoot and smaller phase drop near to the natural frequency. (b) Error magnitude and chirp test trajectories. We choose integral controller feedback gain that shows similar accuracy of PD+DOB's. The left is error magnitude of three controllers. PD controller has larger error than the other two controller in the low frequency region. The right is torque trajectories in the time domain.

this testbed, we intended to demonstrate coordinated position control with two VLCAs, the viability of liquid cooling on an articulated platform, cartesian position control of a weighted end effector, and verification of a linkage design.

The two joints each have a different linkage structure that was carefully designed so that the moment arm accommodates the expected torques and joint velocities as the leg posture changes (Fig. 5.11). For example, each joint can exert a peak torque of approximately 270 N m and the maximum joint velocity ranges between 7.5 rad/s and 20+ rad/s depending on the mechanical advantage of the linkage along the configurations. The joints can exert a maximum continuous torque of 91 N m at the point of highest mechanical advantage. This posture dependent ratio of torque and velocity is a unique benefit of prismatic actuators.

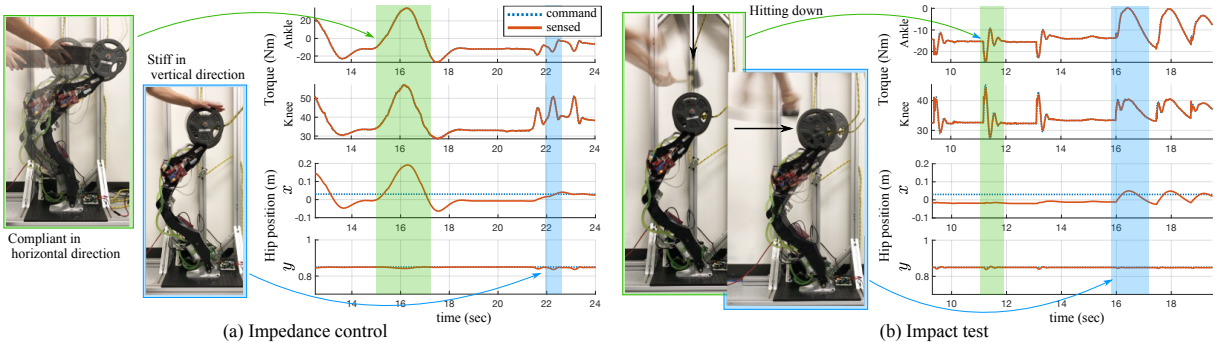


Figure 5.13: **Impedance control.** (a) The robot demonstrates different impedance: stiff in the vertical direction and compliant in the horizontal direction. The high tracking performance of force feedback control results in the overlapped commanded and sensed torques. (b) To show the stability and robustness, we hit the weight with a hammer while operating the impedance control. Even under the impact, force control show stable and accurate tracking.

5.3.4 Torque Feedback Control Test

We test with two types of experiments to demonstrate the performance of our torque feedback controller. First, we test the frequency response of three different feedback controllers with a single actuator. Second, we demonstrate cartesian space impedance control using PD + DOB torque feedback controllers with the single leg testbed. In both tests, we rely only on the rubber deflection measurement and do not use load cell data except for calibrating the rubber deflection offset. To obtain clear deflection data, we used accumulated quadrature encoder data (Fig. 5.7(l)) offset by the initial deflection measured by an absolute encoder (Fig. 5.7(m)) when the robot was powered on.

Fig. 5.12(a) shows the experimental results of our frequency response testing as well as the estimated response based on the transfer functions. We compare three types of controllers: $\mathrm{PD_m}$, $\mathrm{PID_m}$, and $\mathrm{PD_m}$ + DOB. As we predicted in the analysis of Section. 5.3.4, the $\mathrm{PD_m}$ + DOB controller shows less phase drop and overshoot than $\mathrm{PID_m}$. The integral control feedback gain used in the experiment is 300 and the cutoff frequency of DOB's $Q_{\tau d}$ filter is 60Hz, which shows similar error to the $\mathrm{PID_m}$ controller (Fig. 5.12(b)).

Fig. 5.13 shows operational space impedance control, which is explained in Section. 5.4.2. The commanded behavior is to be compliant in the horizontal direction (x) and to be stiff in the vertical direction (y). When pushing the hip with a sponge in the x direction, the robot smoothly moves back to comply with the push, but it strongly resists the given vertical disturbance to maintain the commanded height (Fig. 5.13(a)). To show the stability of our controller, we also

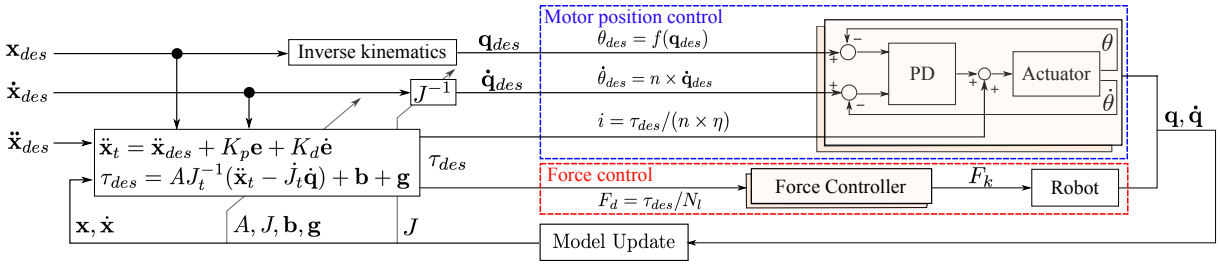


Figure 5.14: **Cartesian motion control diagram.** The commands for each joint, which are a desired joint position, velocity, and torque, are converted to the motor commands. n is the gear ratio and η is the efficiency of the transmission.

test impact by hitting the weight with a hammer. Even when there are sudden disturbances, the torque controllers rapidly respond to maintain good torque tracking performance as shown in Fig. 5.13(b).

5.4 Collocated Position Feedback Control

Collocated feedback control is designing feedback loops using sensors near the actuator, e.g., feedback using encoders attached to motors. Similarly, we use motor position feedback control instead of force feedback control when accurate and robust motion control is desired (rather than precise impedance behavior).

5.4.1 Motor Position Feedback Control

In motor position control mode, the distributed controllers take three commands – desired joint position, velocity, and torque. Then the desired motor position is computed with the equation

$$\theta_{des} = f(q_{des}) + N_m\tau_{cmd}/(\mathrm{k}_r N_l), \tag{5.22}$$

where θ, q, and $f()$ are motor position, joint position, and the mapping from a joint position to a motor position, respectively. Converting motor positions to joint positions requires nonlinear mapping functions, which we can find by solving a closed chain kinematic equation for each joint. As in the previous section, N_m, k_r, and N_l are the speed reduction ratio of the drivetrain, the rubber stiffness, and the moment arm length. The second term on the right hand side of Eq. (5.22) is the expected rubber deflection required to produce the desired torque command. Desired motor velocity is the product of the commanded joint velocity, the moment arm length,

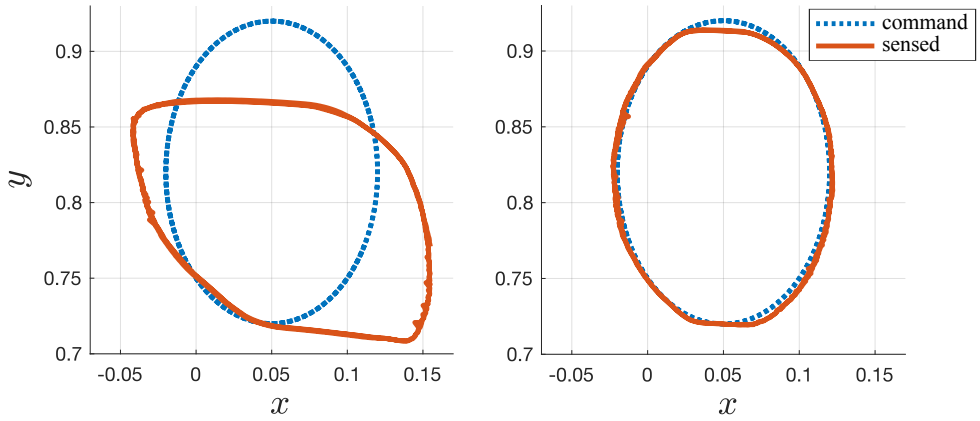


Figure 5.15: **Circular trajectory tracking w/o and w/ the inclusion of rotor inertias.** When the model dynamics includes rotor inertia terms, the tracking performance notably enhances.

and the drivetrain speed ratio, $\dot{\theta}_{des} = \dot{q}_{des} N_l N_m$. Then the final current command is

$$i_m = \eta k_\tau N_m \frac{\tau_{cmd}}{N_l} + k_{p,m}(\theta_{des} - \theta) + k_{d,m}(\dot{\theta}_{des} - \dot{\theta}). \tag{5.23}$$

The motor position control uses the quadrature encoders connected to the motor's axis depicted in Fig. 5.7(f). Since quadrature encoders do not provide absolute position (rather they return the incremental count), we obtain the joint positions by synchronizing the joint position and motor position when the robot is powered on, and then accumulating the tick count from the initial positions. The signal is far less noisy than using the joint encoders. Additionally, this strategy removes the possibility of linkage backlash affecting the stability of the controller.

5.4.2 Cartesian Motion Control

Given cartesian motion trajectories, which are 2nd order B-spline or sinusoidal functions, the centralized controller computes the commands for distributed joint controllers, which are desired joint positions (q), velocities ($\dot{q}$), and torques (τ). The commands are delivered to each embedded microcontroller. In the microcontrollers, we implement the functions converting joint positions, velocities, and torques to motor positions (θ), velocities ($\dot{\theta}$), and feedforward current commands (i). The computation of the current commands account not only for the speed reduction (n) but also the approximate transmission efficiency ($\eta = 0.9$). The control diagram is shown in Fig. 5.14.

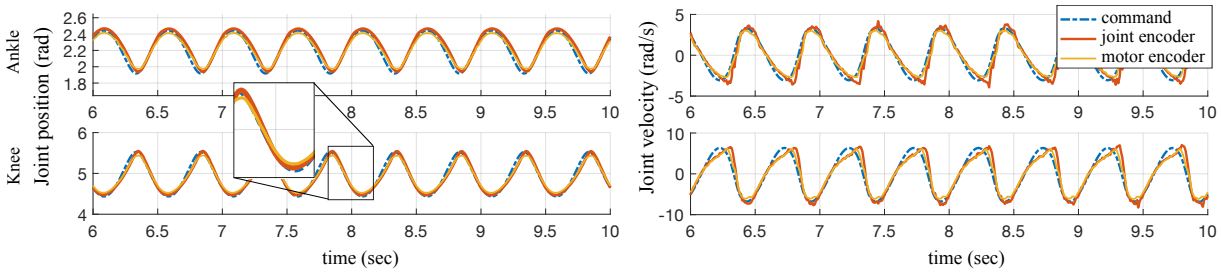


Figure 5.16: **2Hz up and down motion.** Joint position data from joint encoder and motor encoder are presented. When we watch closely, we can see the joint position measured by joint encoder is closer to the reference signal than joint position computed by motor encoder. This is the result of motor position offsets based on torque commands. In this experiment, the maximum observed torque of the ankle joint is 250 N $\cdot$ m and the maximum observed mechanical power of the knee joint is 310W.

To obtain the desired configurations and joint velocities, we use inverse kinematics and the inverse of the Jacobian matrix. Since our testbed has two DoF and the control point is also two dimensional (x, y), there is a unique solution for the commanded cartesian position and velocity within the joints' workspace. When we compute feedforward torque inputs, we use multi-body dynamics and an operational space control formulation.

$$\boldsymbol{\tau} = \boldsymbol{A}\overline{\boldsymbol{J}}_{hip}(\ddot{\mathbf{x}} - \dot{\boldsymbol{J}}_{hip}\dot{\mathbf{q}}) + \mathbf{b} + \mathbf{g}, \tag{5.24}$$

where $\boldsymbol{A}$, $\mathbf{b}$, and $\mathbf{g}$ represent inertia, coriolis, and gravity joint forces, respectively. $\dot{\mathbf{q}} \in \mathcal{R}^2$ is the joint velocity of the leg and τ is the joint torque. $\boldsymbol{J}_{hip}$ is a jacobian of the hip and $\overline{\boldsymbol{J}}_{hip}$ is a dynamically consistent pseudo inverse, which is defined as $\overline{\boldsymbol{J}} \triangleq \boldsymbol{A}^{-1}\boldsymbol{J}^{\top}(\boldsymbol{J}\boldsymbol{A}^{-1}\boldsymbol{J}^{\top})^{-1}$. In our case, $\boldsymbol{J}_{hip}$ is a 2×2 square matrix and assumed to be full-rank. Therefore, the jacobian has a unique inverse, which means $\overline{\boldsymbol{J}} = \boldsymbol{J}^{-1}$.

Note that we need to account for reflected rotor inertia at the joint when computing the feedforward torque command. The current input is transferred to the joint through the drive-train and linkage described in Fig. 5.8 and Fig. 5.11. To account for the rotor inertia in the dynamics, we need to add a diagonal matrix representing these inertias.

$$\boldsymbol{A}_{link+rotor} = \boldsymbol{A}_{link} + \begin{bmatrix} n_1^2 I_1 & 0 \\ 0 & n_2^2 I_2 \end{bmatrix}, \tag{5.25}$$

where I_i is a rotor inertia and $n_i = N_l N_m$. Since N_l is a function of a joint position, n_i is not constant but changes as joint position changes. As stated in Section. 5.3.3, we obtain N_l

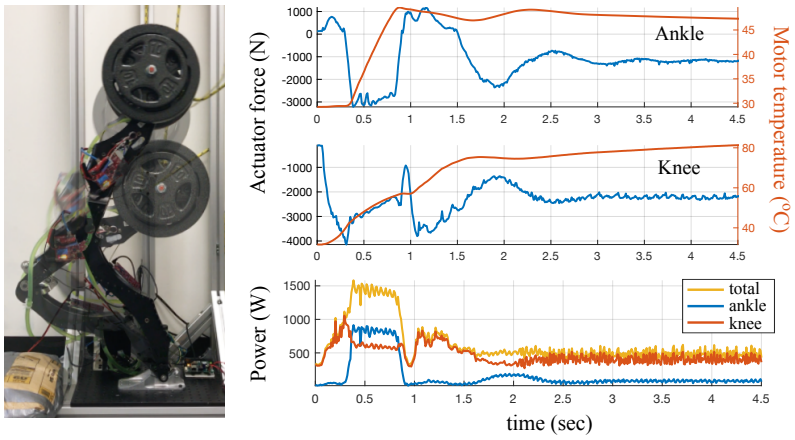


Figure 5.17: **Heavy weight lift.** The leg lifts by 0.3m a 32.5 kg load during 0.4s. There are still large margin to the limit (7300N, 150°C). However, we meet the limit of a power supply, which is 1500W.

by solving for the kinematics of the closed chain linkages shown in Fig. 5.11. Including rotor inertias helps to enhance accuracy of trajectory tracking as we can see in Fig. 5.15.

5.4.3 High Power Motion Experiment

When we demonstrate high power motions such as fast up and down motion and heavy payload lifting, we use the motor position control mode. Fig. 5.16 presents the results of a test comprised of 2Hz up and down motion with 0.32 m of travel while carrying a load of 10 kg at the hip. The joint position is measured by two sensors, which are the joint encoders and motor quadrature encoders. By careful inspection, one can see that joint encoder data is closer to the commanded trajectory than motor encoder data, which is accomplished by accounting for the rubber deflection when computing the motor position command.

Fig. 5.17 presents another test in which the leg lifts a 32.5kg weight. Since the joint torque limits depend on the posture, we plot the linear actuator force instead to show how much capacity remains between the measured output and the theoretical force limit (7300N). We can see that the leg operates in the safe region ($\leq$ 7300N and $\leq$ 155°C) while demonstrating high power motion.

Chapter 6

State Estimation

One of the most challenging issues in experimental studies is to identify the state of the actual system. Unlike simulation, sensor data is noisy, biased, and delayed. Moreover, some information is not available even though it might be crucial for performance, e.g., mechanical structure bending. In this chapter, we introduce estimators we developed for controls. Our estimators are designed based on Kalman filtering, and the performance is verified both in simulations and actual hardware experiments.

6.1 Extended Kalman Filter Using IMU Data and Kinematics

We designed an Extended Kalman-filter (EKF) based orientation estimator and used it for closed-loop balance control of a NAO robot. We integrate angular velocity data of IMU and kinematics of the robot to identify the orientation of body. The estimator also uses the foot contact sensor to adjust the uncertainty covariance.

6.1.1 Review of Extended Kalman Filter

We first review the formulations of EKF. The dynamics of the system is defined by

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k, \\ \mathbf{s}_{k+1} &= h(\mathbf{x}_k) + \mathbf{v}_k, \end{aligned} \tag{6.1}$$

This chapter contains material from the following publication:
[46] Donghyun Kim, Steven Jens Jorgensen, Peter Stone, and Luis Sentis. Dynamic behaviors on the NAO robot with closed-loop whole body operational space control. In 16th International Conference on Humanoid Robots (Humanoids), pages 11211128. IEEE, 2016.
Co-authors contributed on the paper writing.

where $\mathbf{w}_k$ and $\mathbf{v}_k$ are noise following normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{Q}_k)$ and $\mathcal{N}(\mathbf{0}, \boldsymbol{R}_k)$, respectively. The prediction is done by the following process.

$$\begin{aligned} \hat{\mathbf{x}}_{k+1}^{-} &= f(\hat{\mathbf{x}}_k^{+}, \mathbf{u}_k) \\ \boldsymbol{P}_{k+1}^{-} &= \boldsymbol{F}_k \boldsymbol{P}_k^{+} \boldsymbol{F}_k^{\top} + \boldsymbol{Q}_k \end{aligned} \tag{6.2}$$

We use $[\,]^{+}$ and $[\,]^{-}$ to represent estimated and predicted values, respectively. The observation function is $h(\mathbf{x})$, then the update is

$$\begin{aligned} \tilde{\mathbf{y}}_k &= \mathbf{s}_k - h(\hat{\mathbf{x}}_{k+1}^{-}), \\ \boldsymbol{S}_k &= \boldsymbol{R}_k + \boldsymbol{H}_k \boldsymbol{P}_{k+1}^{-} \boldsymbol{H}_k^{\top}, \\ \boldsymbol{K}_k &= \boldsymbol{P}_{k+1}^{-} \boldsymbol{H}_k^{\top} \boldsymbol{S}_k^{-1}, \\ \hat{\mathbf{x}}_{k+1}^{+} &= \hat{\mathbf{x}}_{k+1}^{-} + \boldsymbol{K}_k \tilde{\mathbf{y}}_k, \\ \boldsymbol{P}_{k+1}^{+} &= (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k+1}^{-}, \end{aligned} \tag{6.3}$$

where $\mathbf{s}_k$ is sensed data. Here $\boldsymbol{F}_k$ and $\boldsymbol{H}_k$ are jacobians of $f(\mathbf{x}, \mathbf{u})$ and $h(\mathbf{x})$, respectively.

6.1.2 Formulation of EKF using IMU and Robot Kinematics

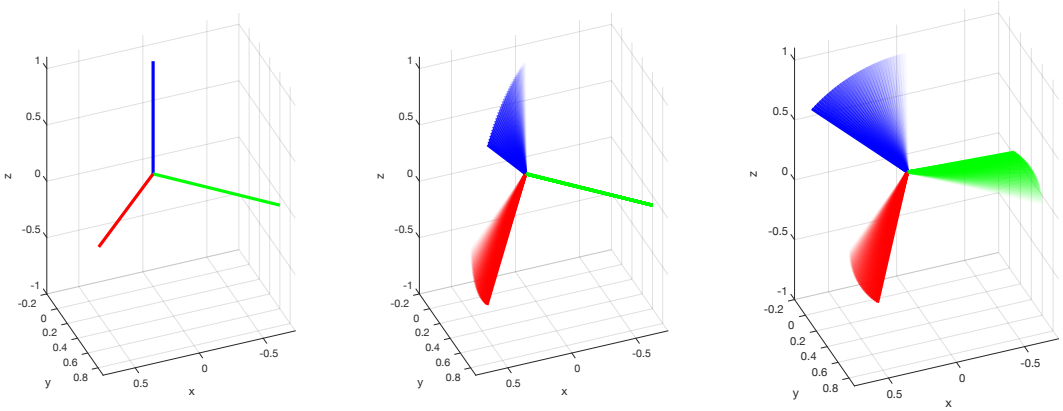
There have been several studies to use kinematic information to estimate the orientation of a floating body. Since the MEMS-based IMU suffers from significant drift, estimation techniques to avoid inherent noise and bias are required to control legged systems. Our approach is based on [7, 83], but a key difference is that we fix the error related to intrinsic and extrinsic rotation. In addition to the ordinary Kalman filter, we utilize the foot force sensor data to manipulate the covariance of the state transition model.

To avoid confusion related with quaternion conventions, we clarify the definition of quaternions used in our estimator in Appendix. A. Due to limited computational power and poor signal quality in the accelerometer data, our estimator does not include linear position related terms, which results in a smaller number of states than [7, 83]. The state of our estimator is

$$\hat{\mathbf{x}} = \begin{pmatrix} \mathbf{q} & \mathbf{z} & \mathbf{b}_\omega \end{pmatrix}^{\top}, \tag{6.4}$$

where $\mathbf{q}$ and $\mathbf{z}$ are the orientation of IMU and stance foot represeted in quaternion. $\mathbf{b}_\omega$ is the bias in angular velocity measurement in an IMU.

In the existing estimator design, there could be confusion regarding intrinsic and extrinsic rotations. Additionally, further confusion comes from conflicting quaternion conventions. To



(a) q_1 (b) $q_1 \otimes q_2$ (c) $q_2 \otimes q_1$

Figure 6.1: **Intrinsic and Extrinsic Rotation.** The figures shows the difference between intrinsic (b) and extrinsic (c) rotation. Here, q_1 is the 45° rotation about z axis and q_2 is the 63^o rotation about y axis.

clarify our approach, we present our convention in Fig. 6.1. Since the Kalman filter update process occurs in the global frame, this update must be done in an extrinsic manner. However, the IMU data from the robot should be added to the estimated orientation in an intrinsic manner since the IMU data is the angular velocity from the local frame (body orientation). Therefore, the Kalman filter prediction rules are

$$\hat{\mathbf{q}}_{k+1}^{-} = \hat{\mathbf{q}}_{k}^{+} \otimes \exp([\hat{\boldsymbol{\omega}}_k]^{\times} \Delta t), \tag{6.5}$$

$$\hat{\mathbf{z}}_{k+1}^{-} = \hat{\mathbf{z}}_{k}^{+}, \tag{6.6}$$

$$\hat{\mathbf{b}}_{\boldsymbol{\omega},k+1}^{-} = \hat{\mathbf{b}}_{\boldsymbol{\omega},k}^{+} \tag{6.7}$$

and the observations are

$$\mathbf{s}_{1,k} = \hat{\mathbf{q}}_{k}^{-} \otimes (\hat{\mathbf{z}}_{k}^{-})^{-1}, \tag{6.8}$$

$$\mathbf{s}_{2,k} = \hat{\mathbf{z}}_{k}^{-}, \tag{6.9}$$

where $\mathbf{s}_{1,k}$ is the orientation difference between the IMU and the stance foot, and $\mathbf{s}_{2,k}$ is the global orientation of the stance foot.

For EKF implementation, we need to find the jacobian of the update and observation function. Here, we first linearize the system and then discretize it. The prediction process for the IMU

orientation is

$$\mathbf{q} = \delta\mathbf{q} \otimes \overline{\mathbf{q}}, \tag{6.10}$$

$$\mathbf{b}_\omega = \overline{\mathbf{b}}_{\boldsymbol{\omega}} + \delta\mathbf{b}_{\boldsymbol{\omega}}, \tag{6.11}$$

where $\mathbf{q}$ is the actual orientation of IMU, $\overline{\mathbf{q}}$ is the estimated value, $\delta\mathbf{q}$ is the orientation adjustment, and similarly, $\mathbf{b}_{\boldsymbol{\omega}}$ is the bias, $\overline{\mathbf{b}}_{\boldsymbol{\omega}}$ is the estimated value, and $\delta\mathbf{b}_{\boldsymbol{\omega}}$ is the bias adjustment. Since the adjustment is done in the global coordinate system, we apply an explicit rotation.

We take a time derivative of the IMU orientation for linearization,

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{q} \otimes \frac{1}{2}\begin{bmatrix} \boldsymbol{\omega} + \mathbf{w}_{\boldsymbol{\omega}} + \mathbf{b}_{\boldsymbol{\omega}} \\ 0 \end{bmatrix} \\ &= \mathbf{q} \otimes \frac{1}{2}\begin{bmatrix} \tilde{\boldsymbol{\omega}} + \mathbf{b}_{\boldsymbol{\omega}} \\ 0 \end{bmatrix}, \end{aligned} \tag{6.12}$$

where $\boldsymbol{\omega}$ is the angular velocity measurement, and $\mathbf{w}_{\boldsymbol{\omega}}$ is the angular velocity noise. From the Eq. (6.12), we can obtain the same equation for $\overline{\mathbf{q}}$,

$$\dot{\overline{\mathbf{q}}} = \overline{\mathbf{q}} \otimes \frac{1}{2}\begin{bmatrix} \boldsymbol{\omega} + \overline{\mathbf{b}}_{\boldsymbol{\omega}} \\ 0, \end{bmatrix}. \tag{6.13}$$

Next,

$$\dot{\mathbf{q}} = \frac{d}{dt}(\delta\mathbf{q} \otimes \overline{\mathbf{q}}) \tag{6.14}$$

$$\mathbf{q} \otimes \frac{1}{2}\begin{bmatrix} \tilde{\boldsymbol{\omega}} + \mathbf{b}_{\boldsymbol{\omega}} \\ 0 \end{bmatrix} = \delta\dot{\mathbf{q}} \otimes \overline{\mathbf{q}} + \delta\mathbf{q} \otimes \overline{\mathbf{q}} \otimes \frac{1}{2}\begin{bmatrix} \boldsymbol{\omega} + \overline{\mathbf{b}}_{\omega} \\ 0 \end{bmatrix} \tag{6.15}$$

$$\mathbf{q} \otimes \frac{1}{2}\begin{bmatrix} \tilde{\boldsymbol{\omega}} + \mathbf{b}_{\boldsymbol{\omega}} \\ 0 \end{bmatrix} \otimes \overline{\mathbf{q}}^{-1} = \delta\dot{\mathbf{q}} + \delta\mathbf{q} \otimes \overline{\mathbf{q}} \otimes \frac{1}{2}\begin{bmatrix} \boldsymbol{\omega} + \overline{\mathbf{b}}_{\omega} \\ 0 \end{bmatrix} \otimes \overline{\mathbf{q}}^{-1} \tag{6.16}$$

Since $\mathbf{q} \otimes \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} \otimes \mathbf{q}^{-1} = Q[\boldsymbol{R}[\mathbf{q}]\mathbf{v}]$, which we described in Appendix. A, we simplify the equation to

$$\delta\dot{\mathbf{q}} = -\delta\mathbf{q} \otimes Q\Big[\boldsymbol{R}[\overline{\mathbf{q}}](\boldsymbol{\omega} + \overline{\mathbf{b}}_{\boldsymbol{\omega}})\Big] + \mathbf{q} \otimes \overline{\mathbf{q}}^{-1} \otimes Q\Big[\boldsymbol{R}[\overline{\mathbf{q}}](\tilde{\boldsymbol{\omega}} + \mathbf{b}_{\boldsymbol{\omega}})\Big] \tag{6.17}$$

$$= -\delta\mathbf{q} \otimes Q\Big[\boldsymbol{R}[\overline{\mathbf{q}}](\boldsymbol{\omega} + \overline{\mathbf{b}}_{\boldsymbol{\omega}})\Big] + \delta\mathbf{q} \otimes Q\Big[\boldsymbol{R}[\overline{\mathbf{q}}](\boldsymbol{\omega} + \mathbf{w}_{\boldsymbol{\omega}} + \overline{\mathbf{b}}_{\boldsymbol{\omega}} + \delta\mathbf{b}_{\boldsymbol{\omega}})\Big] \tag{6.18}$$

$$= \delta\mathbf{q} \otimes Q\Big[\boldsymbol{R}[\overline{\mathbf{q}}](\delta\mathbf{b}_{\boldsymbol{\omega}} + \mathbf{w}_{\boldsymbol{\omega}})\Big] \tag{6.19}$$

$$= \frac{1}{2}\begin{bmatrix} [\boldsymbol{R}[\overline{\mathbf{q}}](\delta\mathbf{b}_{\boldsymbol{\omega}} + \mathbf{w}_{\boldsymbol{\omega}})]^{\times} & \boldsymbol{R}[\overline{\mathbf{q}}](\delta\mathbf{b}_{\boldsymbol{\omega}} + \mathbf{w}_{\boldsymbol{\omega}}) \\ -\boldsymbol{R}[\overline{\mathbf{q}}](\delta\mathbf{b}_{\boldsymbol{\omega}} + \mathbf{w}_{\boldsymbol{\omega}}) & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\phi} \\ 1 \end{bmatrix}, \tag{6.20}$$

where $\delta\mathbf{q} = [\frac{1}{2}\delta\boldsymbol{\phi}, \quad 1]^T$. From Eq (6.19) to Eq (6.20) we apply the matrix transformation from Hamiltonian multiplication,

$$\mathbf{q} \otimes \mathbf{p} = \begin{bmatrix} p_w\boldsymbol{I} + [\mathbf{p}_v]^{\times} & \mathbf{p}_v \\ -\mathbf{p}_v^{\top} & p_w \end{bmatrix} \begin{bmatrix} \mathbf{q}_v \\ q_w \end{bmatrix}. \tag{6.21}$$

Since the multiplication of small numbers is small enough to ignore,

$$\dot{\delta\mathbf{q}} = \boldsymbol{R}[\overline{\mathbf{q}}](\delta\mathbf{b}_{\boldsymbol{\omega}} + \mathbf{w}_{\boldsymbol{\omega}}). \tag{6.22}$$

Eq (6.22) is substituted into the $\boldsymbol{F}_c$ matrix in the linearized system written as $\dot{\delta\mathbf{x}} = \boldsymbol{F}_c\delta\mathbf{x} + \boldsymbol{L}_c\mathbf{w}$, where $\delta\mathbf{x} = [\delta\mathbf{q}, \delta\mathbf{z}, \delta\mathbf{b}]^T$ Therefore,

$$F_c = \begin{bmatrix} 0 & 0 & \boldsymbol{R}[\overline{\mathbf{q}}] \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{6.23}$$

and $\boldsymbol{F}_k = e^{\boldsymbol{F}_c\Delta t}$. As for the orientation difference between the IMU and the stance foot, it is the same as the process described in [83]. $\boldsymbol{F}_k$ with first order Taylor expansion is

$$\boldsymbol{F}_k = \begin{bmatrix} I & 0 & R[\hat{q}_k]\Delta t \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}. \tag{6.24}$$

In the case of the observation, we use the matrix from [83],

$$\boldsymbol{H}_k = \begin{bmatrix} I & -R[\hat{q}_k \otimes \hat{z}_k^{-1}] & 0 \\ 0 & I & 0 \end{bmatrix}. \tag{6.25}$$

The covariance matrix in the prediction is given by the equation, $\boldsymbol{Q}_k = \boldsymbol{F}_k \boldsymbol{L}_c \boldsymbol{Q}_c \boldsymbol{L}_c^\top \boldsymbol{F}_k^\top \Delta t$, where $\boldsymbol{L}_c = \text{diag}\{\boldsymbol{R}[\hat{\mathbf{q}}_k], \boldsymbol{I}, \boldsymbol{I}\}$, and the update matrix is $R_k = R_c/\Delta t$.

Note that the orientation observation in Eq (6.9) is assumed to be constant since the stance foot in the global frame is expected to be flat on the ground. If more than one of the four force sensors' signal in the foot is zero, this implies that the foot is not completely flat on the ground. To handle this scenario, the estimator adjusts the uncertainty covariances for the z term in both the prediction (Q_c) and update (R_c) process to be very high. This results to the Kalman-filter relying more on the IMU data over the kinematics.

6.2 Foot Position Estimator

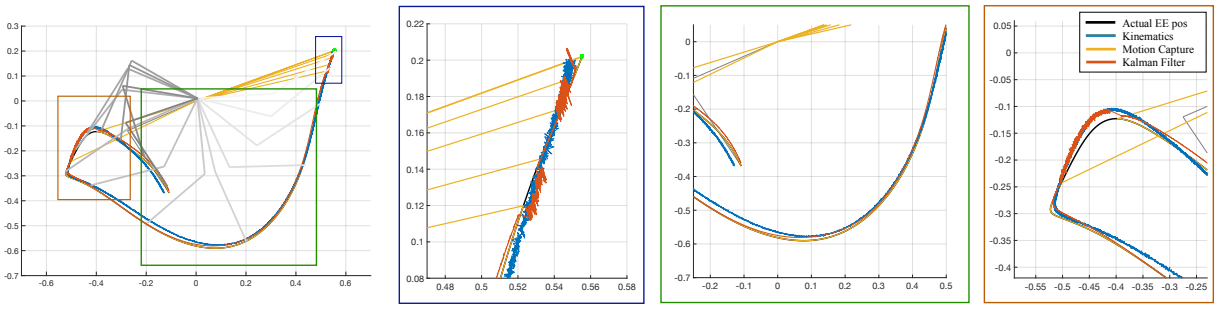
In the experiment presented in Section 7.3, the foot positions measured by a motion capture system and computed via forward kinematics shows around 5cm error. This error is significant because the average step size computed by the foot placement planner (Section. 4.3) is around 10 cm. The kinematic error comes from the bending of carbon tubes making up the structure of Hume's legs. The stance foot location is critical to predicting the CoM's trajectory, and relatively small errors can cause large differences in the next step location.

Our motion capture system provides the global position of feet, but the system has two problems. Firstly, the motion capture sensors can be occluded during tests. The second problem is that the update rate of the motion capture system (480 Hz) is three times slower than the real-time feedback controller (1.5 kHz). To overcome these two problems, we devised foot position estimators to integrate the motion capture data and the robot's kinematics.

6.2.1 EKF and Unscented Kalman Filter Study

Before implementing estimators in the actual system, we tested three different formulations with a simple simulation: two degree-of-freedom open-chain. The simple open-chain has joint encoders at each joint and a motion capture sensor at its end-effector ($\mathbf{p}$). A motion capture system senses the position of the robot's end-effector, and joint encoders provide joint position and velocity. In this simulation, we intentionally add the following errors:

1. The errors in linkage length in the kinematics model,:
   The length of the first linkage is longer than the actual length (0.3m) by 0.01m
   The length of the second linkage is shorter than the actual length (0.3m) by 0.02m



(a) Trajectory of 2 DoF Linkage

(b) Zoomed in views

Figure 6.2: **EKF) Configuration Estimation.** To test the performance of our estimator, we made a simple 2 DoF Openchain simulator and saw how the EKF estimate the end effector position with kinematics and motion capture data.

2. Biases in joint position:
   The bias in the first joint encoder is 0.03rad
   The bias in the second joint encoder is -0.05rad

3. Motion capture data are sometimes not available (occlusion).

4. Both motion capture data and encoder data are corrupted by white noise.

All tested estimators adopt the on-line uncertainty adjustment method presented in the previous section. When the motion capture sensor is occluded, the numbers of the observation covariance matrix associated with motion capture data become huge, which implies large uncertainty in the observation data. The followings are formulations and a simulation result of three estimators.

EKF) Basic Configuration Estimation. The first basic estimator estimates the robot's configuration. The state of this estimator is simply joint positions and velocities:

$$\mathbf{x} = \begin{bmatrix} \mathbf{q} & \dot{\mathbf{q}} \end{bmatrix}^\top, \tag{6.26}$$

and the discrete dynamics used in the prediction process is

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + \dot{\mathbf{q}}_n \Delta t, \\ \dot{\mathbf{q}}_{n+1} &= \dot{\mathbf{q}}_n + \boldsymbol{A}^{-1}(\boldsymbol{\tau} - \mathbf{b} - \mathbf{g})\Delta t. \end{aligned} \tag{6.27}$$

The Jacobian of the dynamics is

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & \boldsymbol{I}_{2\times 2}\Delta t \\ \frac{\partial}{\partial \mathbf{q}}\boldsymbol{A}^{-1}(\boldsymbol{\tau} - \mathbf{b} - \mathbf{g})\Delta t & -\frac{\partial}{\partial \dot{\mathbf{q}}}\boldsymbol{A}^{-1}\mathbf{b}\Delta t + \boldsymbol{I}_{2\times 2} \end{bmatrix} \tag{6.28}$$

The observation consists of joint positions and velocities measured by encoders, and the end-effector position sensed by a motion capture system.

$$\mathbf{z} = \begin{bmatrix} \mathbf{q} & \dot{\mathbf{q}} & \mathbf{p} \end{bmatrix}^\top. \tag{6.29}$$

Therefore, the Jacobian of the observation function is

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{2\times 2} \\ \boldsymbol{J}_{\text{ee}} & \mathbf{0} \end{bmatrix}. \tag{6.30}$$

The result of this estimator is shown in Fig. 6.2. The end-effector position measured by a motion capture system (yellow line) occasionally becomes zero because of occlusions. The red line representing the estimated positions proves that the estimator behaves as we expected. The estimated positions are similar to the motion capture data most time but follow kinematics when the motion capture data are not available. However, noticeable noise exists in the estimation, and the estimation shows the difference from motion capture data during the period without occlusions. This is because the state of this estimator does not include the end-effector position but includes only robot configuration.

EKF) Configuration and End Effector Position Estimation. In addition to the joint configuration, we added the end-effector position in the state,

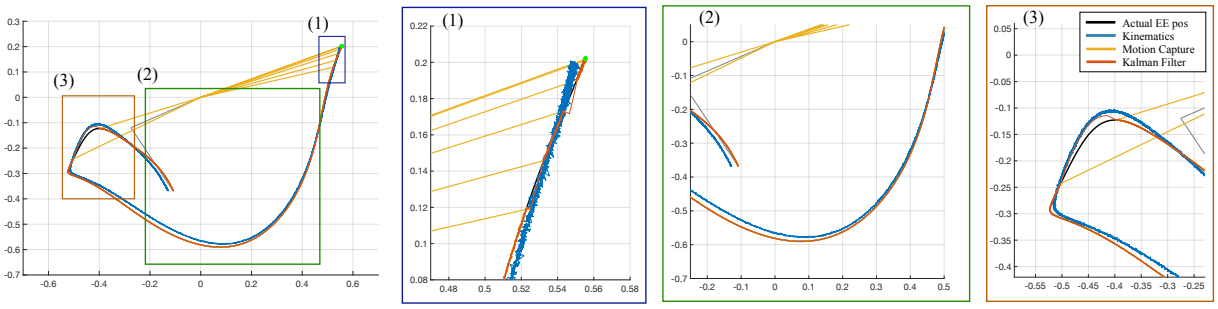
$$\mathbf{x} = \begin{bmatrix} \mathbf{q} & \dot{\mathbf{q}} & \mathbf{p} \end{bmatrix}^\top. \tag{6.31}$$

Then the dynamics becomes

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + \dot{\mathbf{q}}_n \Delta t, \\ \dot{\mathbf{q}}_{n+1} &= \dot{\mathbf{q}}_n + \boldsymbol{A}^{-1}(\boldsymbol{\tau} - \mathbf{b} - \mathbf{g})\Delta t. \\ \dot{\mathbf{p}}_{n+1} &= \dot{\mathbf{p}}_n + \boldsymbol{J}_{\text{ee}}\dot{\mathbf{q}}_n \Delta t. \end{aligned} \tag{6.32}$$

The Jacobian of the dynamics used in the prediction is

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & \boldsymbol{I}_{2\times 2}\Delta t & \mathbf{0} \\ \frac{\partial}{\partial \mathbf{q}}\boldsymbol{A}^{-1}(\boldsymbol{\tau} - \mathbf{b} - \mathbf{g})\Delta t & -\frac{\partial}{\partial \dot{\mathbf{q}}}\boldsymbol{A}^{-1}\mathbf{b}\Delta t + \boldsymbol{I}_{2\times 2} & \mathbf{0} \\ \frac{\partial}{\partial \mathbf{q}}\boldsymbol{J}_{\text{ee}}\dot{\mathbf{q}}\Delta t & \boldsymbol{J}_{\text{ee}}\Delta t & \boldsymbol{I}_{2\times 2} \end{bmatrix} \tag{6.33}$$



(a) Trajectory of 2 DoF Linkage

(b) Zoomed in views

Figure 6.3: **EKF) End Effector Position Estimation.** We add the end-effector position in the state to get better position estimation. The noise in the estimation is much less than the previous estimator.

Observation is identical to the one of the previous estimator.

$$\mathbf{z} = \begin{bmatrix} \mathbf{q} & \dot{\mathbf{q}} & \mathbf{p} \end{bmatrix}^\top, \tag{6.34}$$

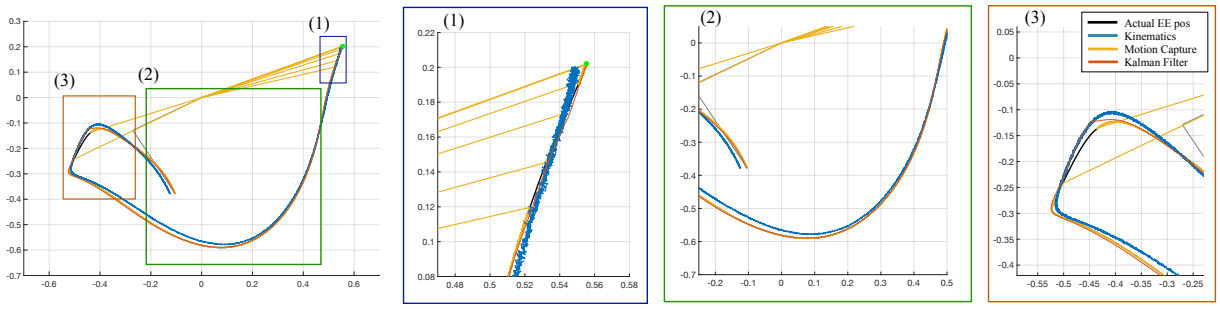
and $\boldsymbol{H} = \boldsymbol{I}_{6\times 6}$ because state and observation are same.

The result presented in Fig. 6.3 shows that we can get better estimate with this formulation. Compared with the previous one, Fig. 6.3(1) shows much less noise. Estimation (red line) in Fig. 6.3(2) is well matched to the motion capture data (yellow line). We can find that the estimator rapidly converges to motion capture data when the position data become available in Fig. 6.3(3).

UKF) Configuration and End Effector Position Estimation. We also tested unscented-Kalman filter (UKF) with the same formulation of the previous one. All setups are identical except we do not need to linearize the dynamics and observation this time because UKF propagates estimation by projecting sigma points instead of relying on linearization. The result presented in Fig. 6.4 shows the best performance among the three estimators we tested. Noise level is similar to EKF with the end-effector position estimation, but the trend of estimation is more stable than the previous one.

6.2.2 Experimental Verification of Foot Position Estimator

Unfortunately, we could not use UKF based estimator because of limited computation power of our control PC. In the experiment, we use 1.5 kHz update rate, which means that all computation must be finished in 0.3ms because Ethernet communication spends 0.3ms. Since WBOSC



(a) Trajectory of 2 DoF Linkage

(b) Zoomed in views

Figure 6.4: **Unscented Kalman Filter.** We also tested UKF with the same formulation of the previous EKF. UKF based estimator shows the best performance among the three candidates.

spends 0.2ms including robot dynamics update, only 0.1ms is allowed for estimators. This strict time limit constrain us to use simple formulations; therefore, the estimator used in the experiment is much simpler than ones in the previous section. The state is

$$\mathbf{x} = \begin{bmatrix} \dot{\mathbf{q}} & \mathbf{p} \end{bmatrix}, \tag{6.35}$$

and the prediction is defined by

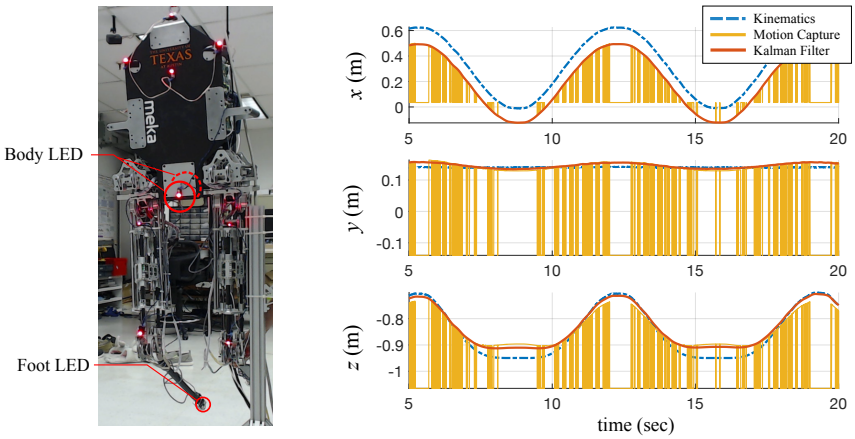
$$\begin{aligned} \dot{\mathbf{q}}_{n+1} &= \dot{\mathbf{q}}_n, \\ \mathbf{p}_{n+1} &= \mathbf{p}_n + \boldsymbol{J}\dot{\mathbf{q}}\Delta t. \end{aligned} \tag{6.36}$$

We remove the acceleration input in configuration velocity equation since the inclusion of the acceleration requires partial derivatives of mass, coriolis, and gravity terms. The Jacobian of the dynamics is

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I}_{12\times 12} & \mathbf{0} \\ \boldsymbol{J}_{3\times 12} & \boldsymbol{I}_{3\times 3} \end{bmatrix}. \tag{6.37}$$

Observation is simply configuration velocity measured by an IMU and joint encoders and foot position measured by a motion capture system. Therefore, the state and the observation are identical, which means that $\boldsymbol{H} = \boldsymbol{I}_{15\times 15}$.

Experimental result are shown in Fig. 6.5. In this test, we fix the left leg of Hume at the bar and swing the right foot in the sagittal plane. The data shows the clear discrepancy between motion capture data (yellow) and kinematics data (blue). The motion capture sensor frequently disappears and the occlusions are shown as sudden drops of the position data in these plots. However, the estimation (red) is not affected by the missing signals and shows clear data.



(a) LED Marker Configuration

(b) Foot Position

Figure 6.5: **Experimental Validation of Our Foot Position Estimator.** After we verify the performance of the proposed EKF with a simple simulation, we implemented the estimator in Hume and test it by watching the position data of the swing foot.

Chapter 7

Experiment: Point-Foot Biped Robot, Hume

In this chapter, we focus on dynamic point-foot locomotion, which has been extensively explored yet remains a challenging topic [15, 35, 53, 63, 74, 79, 94, 105]. We designed a robot, Hume, with point feet in order to lower the complexity of the mechanical system and allow for agile behaviors by minimizing the inertia of the swing leg. Extensive efforts have been put into feedback control strategies [53, 94]. Many attempts have been made to design control systems for walking robots which avoid the use of the most complex, but most accurate, models. Very early studies used feedback control to tie the walking motion to stable limit cycles of the well-understood stable oscillator system [41]. Compass gait robots with simple sensory feedback were made to traverse unknown rough terrain. [35] Another highly successful control strategy for simple but highly dynamic hopping behavior was to use separate simple controllers for each of the forward running velocity, hopping height, and body attitude. [78] A similarly simple strategy was latter successfully applied to biped walking robots with point feet under a Virtual Model Control (VMC) framework [74]. Using physically intuitive virtual springs and dampers, this approach was used to stabilize planar robots with series elastic actuators.

On the other end of the model complexity spectrum, numerous advances in model-based control using rigid body dynamics have achieved high tracking performance at the cost of computational efficiency [79, 94]. Rabbit [105] employs event-based control of the average walking rate by using virtual constraints synchronized with impact dynamics. However, because of the actuator limitations and lack of leg compliance, Rabbit's running is somewhat limited. This led to the design of a planar biped, MABEL, which had springs in its drivetrain. MABEL [94] successfully performs stable and fast running through a time-invariant feedback control law, virtual constraints, and hybrid zero dynamics. The ATRIAS [79] and MARLO robots went on to successfully achieve untethered point foot bipedal locomotion.

This chapter contains material from the following publications:
[49] Donghyun Kim, Ye Zhao, Gray Thomas, Benito R Fernandez, and Luis Sentis. Stabilizing Series-Elastic Point-

7.1 Open-loop Locomotion Test

In this section, we present an experimental study of the first steps with Hume. In this study, Hume is first constrained to planar motion by a linkage mechanism. We present our application of phase space planning to one, two, and three step walking, the last one over an obstacle. In the implementation, we modified the original theory and added ad-hoc adjustments since the robot could not follow the original theory's planned walking trajectories despite their theoretical stability. We present a good correlation between the phase space plans and our various experiments, and an analysis of the robot's final behavior. Overall the planner and ad-hoc modifications allowed us to execute very smooth gaits even over non-flat surfaces but at the same time demonstrated the shortcomings of open loop techniques.

Although a closed-loop control system is preferable, it is not always available for various reasons-e.g., time delay between high-level and low-level controls. In this case, open-loop control is another candidate strategy that is straightforward to implement. By properly tuning predefined actuator commands, this strategy enables bipedal robots to achieve extremely dynamic motions such as sprinting [63] and running flips [29].

In our study, we eschewed high-level feedback control and relied more heavily on a motion planner based on inverted pendulum dynamics. The inverted pendulum is a simplified model used to abstract complex biped or humanoid dynamics [28,38,101]. Zero Moment Point (ZMP), [28, 98, 104], a mainstream planning method which also uses the inverted pendulum model, is designed to manipulate the center of mass (CoM) of a full humanoid in order to avoid tipping the foot onto one of its edges. Capture point based methods (CP) [54] take advantage of first-order dynamics within the analytic solution of the inverted pendulum problem to efficiently predict CoM behavior given the center of pressure location.

The planning strategy developed by our group, which we named phase space planning, avoids use of analytic solutions and thus can handle arbitrary height surfaces [90, 113]. The algorithm, which we proposed in [113], determines when a walking biped should switch feet when the CoM

Foot Bipeds Using Whole-Body Operational Space Control. IEEE Transactions on Robotics, 32(6):13621379, 2016.

[50] Donghyun Kim, Ye Zhao, Gray Thomas, and Luis Sentis. Empirical Modifications to a Phase Space Planner Which Compensates for Low Stiffness Actuation in a Planar, Point-Foot, Biped Robot. In the ASME 2014 Dynamic Systems and Control Conference, page V001T11A001. ASME, 2014.

[51] Donghyun Kim, Ye Zhao, Gray Thomas, and Luis Sentis. Assessing Whole-Body Operational Space Control in a Point-Foot Series Elastic Biped: Balance on Split Terrain and Undirected Walking. arXiv.org, page 2855, January 2015.

My role covers algorithm development, programming, experiment, and the major portion of writing.

is known to be maintaining a height defined by an arbitrary continuous surface. The use of an arbitrary continuous surface is an improvement over methods which assume a flat height surface and is better suited to rough terrain locomotion and natural walking with slight height variations throughout the stride. In essence, it uses numerical integration of the inverted pendulum model constrained to a somewhat arbitrary continuous height surface. Numerically computing the intersection of two phase space paths corresponds to associating the time frames of two steps. Phase space planning explains the dynamic limitations of dual support using limiting single-support cases and can predict the maximum and minimum acceleration for any point in dual contact.

Although the previously developed planner successfully tackled the problem of path generation for stable walking, a number of limitations (e.g., inaccurate model, impulse from landing, poor position control) prevented its implementation in a real system. In this section, we present additional methods to address the real system dynamics that do not correlate well with the simple dynamic model of our planner. First, we modified the equation of an inverted pendulum dynamics to consider the influence of a slider linkage. Second, we modified the height surface to smoothly decrease during a landing motion to alleviate the swing foot landing impulse. Third, the CoM velocity was boosted at the lifting time to ensure that each step began smoothly.

7.1.1 Hardware Setup

Hume is a teen-size humanoid robot measuring 1.5 meters in height and 20 kg in weight. The leg kinematics resemble simplified human kinematics and contain an adduction/abduction hip joint followed by a flexion/extension hip joint followed by flexion/extension knee joint as shown in Fig. 7.12. The lack of ankle joints allows the shank to be essentially just a lightweight carbon fiber tube. At the tip of the shank we have incorporated contact sensors which are essentially limit switches. The series elastic actuators (SEA) on all six joints are based on a sliding spring carriage connected to the output by steel ropes. The deformation in the springs are directly measured within the carriage assembly. The concept, kinematics, and specifications of the robot were proposed by our team at UT Austin, and the design was executed in collaboration with Meka Robotics and manufactured by that company. For fall safety the robot is attached to a trolley system with a block and tackle system which allows easy lifting and locking at a height. In addition, the robot has a removable planarizer mechanism which constrains the motion of the robot to the saggital plane. The pitch of the robot remains unconstrained, as the planarizer connects to the robot through a set of bearings. Ultimately, pitch, forward motion, and vertical motion are allowed, while lateral motion, roll, and yaw are prohibited.

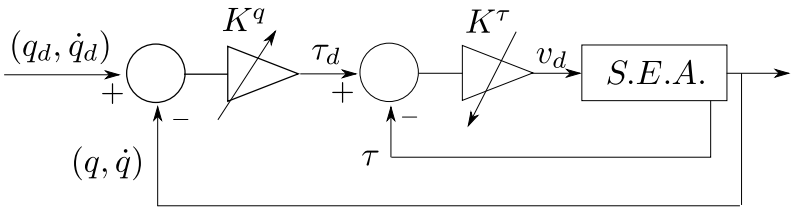


Figure 7.1: **Cascaded control structure.** There are two cascaded feedback loops. The inner loop is a torque feedback loop at the low level. The outer loop is the a joint feedback loop at M3 server level. Both of these two feedback loops are based on PD control.

From an electrical point of view the robot is controlled with distributed digital signal processors, connected by an EtherCat network to a centralized PC running a real-time RTAI Linux kernel. This communication system introduces a 1ms delay from the Linux machine to the actuator DSPs and back. Each DSP controls a single actuator, and they do not communicate directly with each other. Power is delivered through a tether from an external source.

7.1.2 Open-Loop Locomotion using Joint Position Controllers

Although the studies about single series elastic actuator proved that accurate and stable control of SEAs is possible [67], achieving accurate position tracking for legged robots with series elastic actuators (SEAs) remains as an open question. The compliant element in SEAs make robot legs quite "soft" and induces sagging errors due to gravity. This problem is usually solved using integral control gains, but we avoided use of integral control since it is known to complicate the stability of the whole robot system. Using high controller gains helps to reduce this gravity-induced error to some degree. However, if the delay between state sensing and controller command is larger than around 1ms, it dramatically degrades the performance of force feedback and can even cause the system to become unstable. The high level control system on our robot has been unable to reduce this delay below 7ms thus far, but the distributed low level system has a 1ms delay. To avoid this latency problem, feedback loops are not adopted in our high level controllers. Instead, we rely on increasing joint position feedback PD gains K_q while reducing joint torque feedback PD gains K_τ in the distributed system as shown in Fig. 7.1. The controller gains are manually tuned. This study aims to implement a modified phase space planner with an open loop control strategy at the high level and this simple, joint level feedback strategy at the low level.

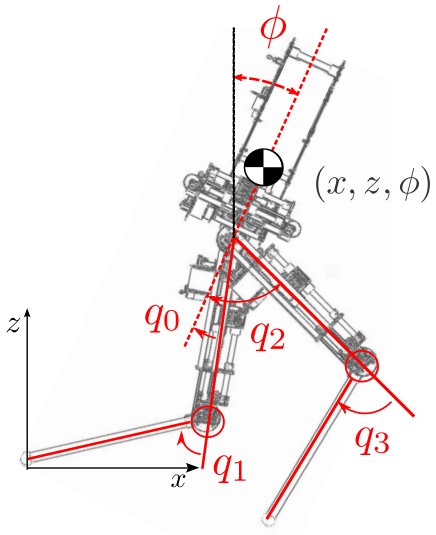


Figure 7.2: **2D configuration of Hume.** Each leg of Hume has three actuated joints. For planar motion, only the hip and knee flexion/extension joints are involved in the inverse kinematics. Hip abduction/adduction joints are position controlled to keep the legs in plane. We use a local coordinate system attached to the stance foot contact point.

7.1.3 Inverse Kinematics

In order to generate joint trajectories at the high level, we employed inverse kinematics based on a known foot location and a planned center of mass location. As will be explained, the center of mass location for this calculation was empirically modified to somewhat offset the effect of the joint sagging in the low level controller. Our inverse kinematics were solved analytically as follows. The cartesian space position vector is defined as $\mathbf{x} = (x_{com}, z_{com}, x_{foot_{sw}}, z_{foot_{sw}}, \phi)^T$, where (x_{com}, z_{com}) are the horizontal and vertical center of mass (CoM) positions, $(x_{foot_{sw}}, z_{foot_{sw}})$ are the horizontal and vertical swing foot positions, and ϕ is the torso orientation with respect to vertical. The configuration vector in joint space is defined as $\mathbf{q} = (q_0, q_1, q_2, q_3, \phi)^T$, where (q_0, q_1) are the stance leg knee and hip joints respectively and (q_2, q_3) are the swing leg knee and hip joints respectively (Fig. 7.2). Here the CoM positions (x_{com}, z_{com}) are defined in a local coordinate with the origin at the stance foot $(x_{foot_{st}}, z_{foot_{st}}) = (0, 0)$.

Then, we derive the following inverse kinematic equations.

$$q_0 = \sin^{-1}(\frac{L_1^2 - L_2^2 + x_{com}^2 + z_{com}^2}{2L_1\sqrt{x_{com}^2 + z_{com}^2}}) - \tan^{-1}(\frac{x_{com}}{z_{com}}) - q_1 + \phi - \frac{\pi}{2} \tag{7.1}$$

$$q_1 = -\cos^{-1}\left(\frac{(x_{com}^2 + z_{com}^2 - L_1^2 - L_2^2)}{2L_1L_2}\right) \tag{7.2}$$

$$q_2 = \cos^{-1}\left(\frac{a^2 + b^2 + L_2^2 - L_1^2}{2L_2\sqrt{a^2 + b^2}}\right) + \phi + \tan^{-1}(\frac{a}{b}) \tag{7.3}$$

$$q_3 = \cos^{-1}\left(\frac{a^2 + b^2 - L_1^2 - L_2^2}{2l_1l_2}\right) + 2\phi - 2q_2 \tag{7.4}$$

Where $a = x_{foot_{sw}} - x_{com}$, $b = z_{com} - z_{foot_{sw}}$ for notational convenience.

The phase space planner we used to accomplish walking motion is modeled on the one proposed in [113]. However, major adjustments were made to compensate for the inaccurate trajectory following of the low level controller and simple inverted pendulum model. Additionally, as previously mentioned, we were unable to implement a high level controller due to stability problems resulting from our large software-induced latency. This resulted in our use of a joint level position feedback controller which allows the robot to support its weight stably at the cost of high Cartesian position error. In this section, we present three adjustments to our planner which allowed more accurate prediction of the robot's behavior despite these limitations.

7.1.4 Inverted Pendulum with a Sliding Bar

Since we use a slider rocker linkage to keep our robot in plane, the constrained robot system behaves like the articulated bodies of [19] rather than a simpler rigid body as assumed in [113]. The connection between the robot's body and the rocker link, with its constrained slider, means that the combined system does not technically have a center of mass. However, since the rotational joint between the rocker link and the body is very close to the robot's original center of mass, the center of rotation due to an applied force changes only slightly when the direction of the force changes. To keep the model simple we neglect this difference and assume that a center of mass exists. We do, however, model disparate inertia and gravitational forces in the vertical and horizontal directions. With this modification in place we arrive at our updated equation for

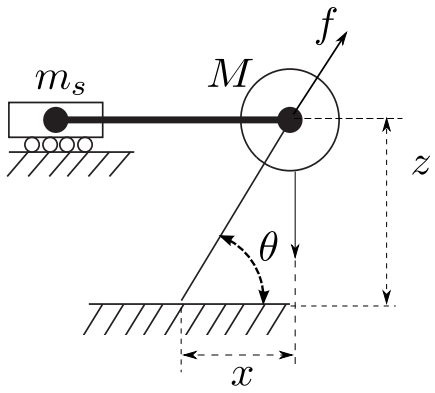


Figure 7.3: **Inverted pendulum with a sliding bar.** The Inverted pendulum model is used to model Hume's dynamics. It has a point foot contact and massless leg. The slider linkage is modeled as an extra mass m_s, which modifies the pendulum dynamics by a scaling factor. More details refer to Equation (7.5).

forward acceleration, following the nomenclature of Fig. 7.3:

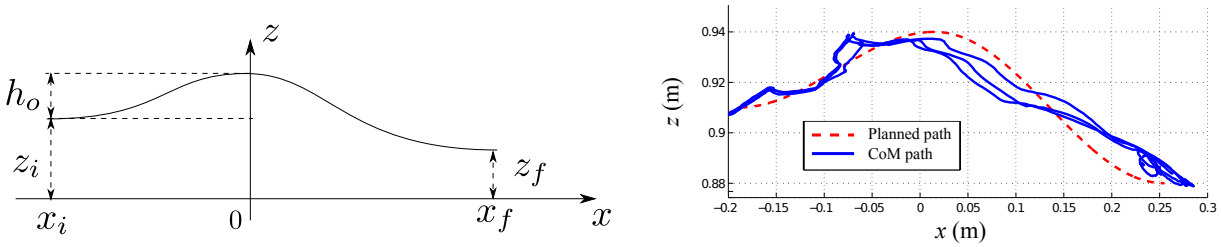
$$\begin{aligned} f\sin\theta &= Mg + M\ddot{z} \\ f\cos\theta &= (M+m_s)\ddot{x} \\ \tan\theta &= \frac{z}{x} \\ \therefore \ddot{x} &= \frac{Mx}{(M+m_s)z}(g+\ddot{z}) \end{aligned} \tag{7.5}$$

Here f and M are the reaction force and robot mass, respectively. As you can see, the ratio $\frac{M}{M+m_s}$ modifies the familiar dynamic equation of a pendulum constrained to a height surface.

Additionally, while we assume a CoM exists, we use an empirically determined estimate for its location. We found this estimate by moving the body towards the forward tipping point in dual support, and iteratively moving the CoM estimate towards the line extending vertically from the forward foot at the instant tipping began. This method produced an approximation which is slightly behind our robot manufacturer's CoM estimate for the body link.

7.1.5 CoM Path Modification to Account for Swing Foot Landing Impulse

One of the advantages of phase space planning is that we can plan for any continuous CoM height surface. Knowing that our controller behaves badly in the presence of large impacts, we designed a height surface which reduces the impact of landing. To do this we smoothly reduce



(a) CoM height surface design

(b) Experiment results of three trials

Figure 7.4: **CoM path.** (a) Smooth CoM height surface is designed. (b) Experimental data of CoM height

the CoM height at the end of the stepping motion, resulting in the path shown in Fig. 7.4. This CoM path is composed of two sinusoidal parts, which meet at the peak height with first order continuity. The sinusoidal parts are derived from the height surface parameters. The first part is found

$$\begin{aligned} z &= z_i + \frac{1}{2}h_o(1 - \cos(\frac{x}{-x_i}\pi)) \\ a &= \frac{1}{2}h_o\Big(\frac{\pi}{-x_i}\Big)\sin(\frac{x - x_i}{-x_i}\pi) \\ b &= \frac{1}{2}h_o\Big(\frac{\pi}{x_f}\Big)^2\cos(\frac{x - x_i}{-x_i}\pi)) \end{aligned} \tag{7.6}$$

And the second

$$\begin{aligned} z &= z_f + \frac{1}{2}(z_i + h_o - z_f)(1 + \cos(\frac{x}{x_f}\pi)) \\ a &= -\frac{1}{2}(z_i + h_o - z_f)\Big(\frac{\pi}{x_f}\Big)\sin(\frac{x}{x_f}\pi) \\ b &= -\frac{1}{2}(z_i + h_o - z_f)\Big(\frac{\pi}{x_f}\Big)^2\cos(\frac{x}{x_f}\pi) \end{aligned} \tag{7.7}$$

The height path is a function of horizontal position. This is converted to a feed-forward position trajectory by first determining the horizontal position as a function of time. This calculation is performed recursively as a discrete integration. Using $R_m = \frac{M}{M+m_s}$ as the ratio of effective masses and Δt as the integration time step, the following equations describe the iteration n of

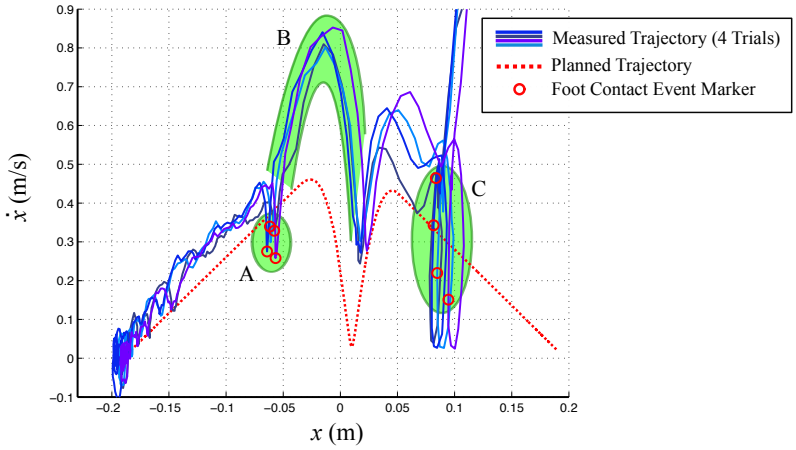


Figure 7.5: **CoM trajectories for separate single step experiments without the post-lift-off speed boost.** Swing phase starts in area A, and ends in area C. Area B represents a period of high body tilt rate. At the landing point, there is impact from the ground, and body velocity is dramatically increased. The measured data disobeys the rules of phase space plots to some extent because the joint velocities are filtered before they are use to compute the center of mass velocity and thus it is an imperfect derivative of the center of mass position. Foot contact events in area C immediately follow an extremely high acceleration foot halting motion. Data after landing is omitted, since the forward kinematics do not take into account the observed lifting of the stance foot during impact. The robot rocks considerably after landing.

the integration procedure.

$$A = \frac{x}{z} R_m (g + b\dot{x}^2) \tag{7.8}$$

$$B = 1 - R_m \frac{x}{z} a \tag{7.9}$$

$$\ddot{x}_{n+1} = \frac{A}{B} \tag{7.10}$$

$$\dot{x}_{n+1} = \dot{x}_n + \ddot{x}_n \Delta t \tag{7.11}$$

$$x_{n+1} = x_n + \dot{x}_n \Delta t + 0.5 \ddot{x}_n \Delta t^2 \tag{7.12}$$

Here, a, b are the values defined in Eq. (7.6) and Eq. (7.7).

7.1.6 Post-Lift-Off Speed Boost

We experimentally found that after lifting the stance foot, the phase space data displayed an unexpected acceleration when we successfully initiated a step. Fig. 7.5 shows this acceleration

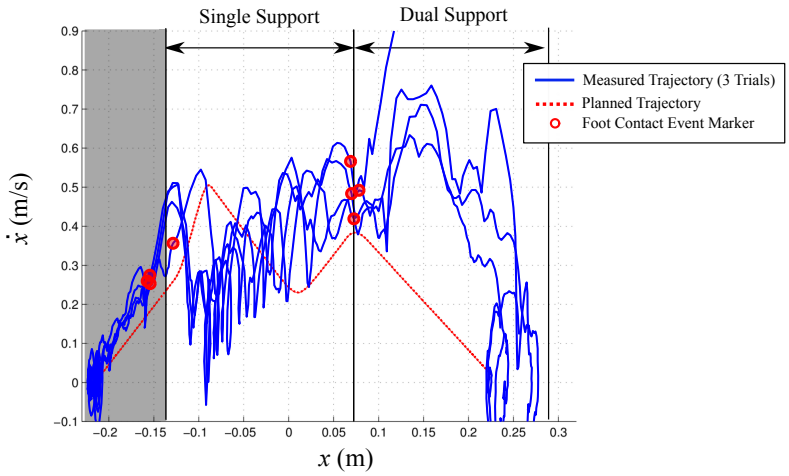


Figure 7.6: **CoM x Phase Space Path of Single Step Test.** The red line indicates the planned path and the blue lines represent three separate trials. The red circles indicate the time when the foot is left and landed. The shaded area highlights the initial dual support phase. After lifting the foot, the robot follows inverted pendulum dynamics. After foot touchdown, the CoM velocity smoothly decreases.

in area B. Later analysis showed this acceleration to be correlated with angular acceleration of the body. As originally designed, the planner creates trajectory based on a constant body pitch, however in our experimental setup the pitch deviates considerably from our intended pitch, and it helps to boost body velocity. In a similar study, Jerry Pratt slightly perturbed the robot with a push to initiate step motion [74]. In our case, the speed boost which is presented by a sinusoidal segment in phase space is inserted into the phase space plan at the point of foot liftoff. As a temporary, ad-hoc method of computing this boost, the post boost phase space path is literally the second part of the original phase space path translated in both velocity and position.

7.1.7 Experiment: Single, Double, and Triple Step With an Obstacle

We performed three types of experiments on the Hume robot: single step, double step, and triple step with an obstacle. In the first two types the floor is flat and level.

Single Step

In this experiment, the motion consists of three parts - dual support forward acceleration of the body, a quick swing leg motion during single support, and a steady deceleration after landing

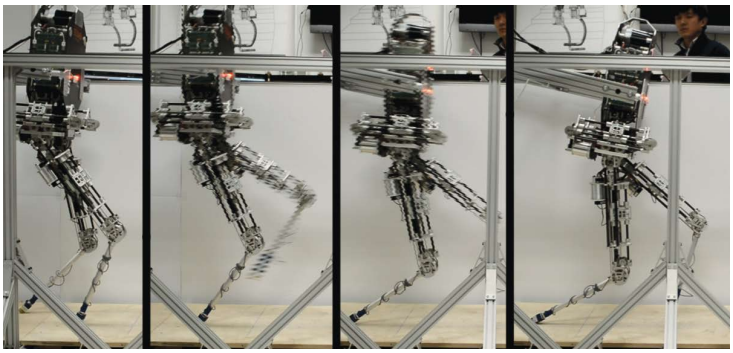


Figure 7.7: **One step test.** Images progress from left to right.

(Fig. 7.7).

As can be seen in Fig.7.6, the planned CoM path has been modified to include the lift-off speed boost and the real data demonstrates a very similar behavior when the single support phase starts. With this boost we allow the robot to more closely follow inverted pendulum dynamics in single support, compensating for the sudden tilting caused by the unmodeled motion of swing leg at lifting time. At the end of single support, the swing foot lands very close to the intended time determined by the planner. Given our simple planner, the impact occurring due to foot touchdown is not avoided, and causes a rotation of the body. This causes the high CoM velocity seen after landing in the phase space. Despite this rotation effect, the overall motion is smooth and the CoM path converges to zero velocity with slight torso rotation continuing to be highly visible in the phase space.

Another important point which we have to be careful about is the velocity of the swing leg. When the velocity of a swing leg is too large, the dynamics of the swing leg, which are not considered in the inverted pendulum model, cannot be ignored. Empirically, so far as hip joint velocity remains below 5 rad/s (Fig. 7.8) with knee joint velocity below 10 rad/s, the robot deviates acceptably little from inverted pendulum dynamics.

Double Step

In the one step test, we show that our techniques are capable of handling our real robot dynamics. However the walking is simplified by the dual contact at the end. With this result, we attempted to make robot not only step but also walk. The CoM phase paths of two step walking tests (Fig. 7.9) are shown in Fig. 7.10 and show that actual behavior is similar to the planned trajectory even though there is velocity noise caused by wobbling motions.

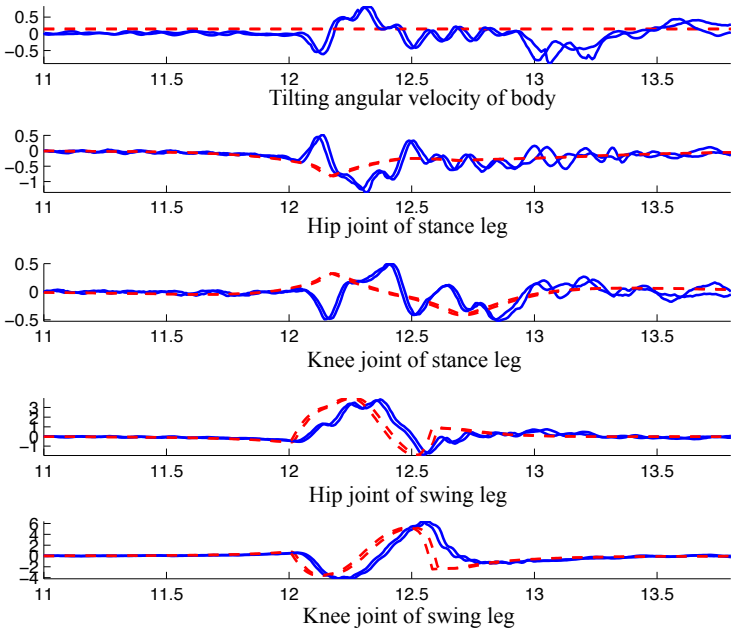


Figure 7.8: **Angular velocity of the body and the joints during the one step test.** x-axis is time (sec) and y-axis is angular velocity (rad/sec). The red dotted lines presents planned velocity trajectories and blue lines are experimental data from two trials. The angular velocity of the hip joint and the knee joint of the swing leg is bounded within 5 rad/s and 10 rad/s during motion, respectively.

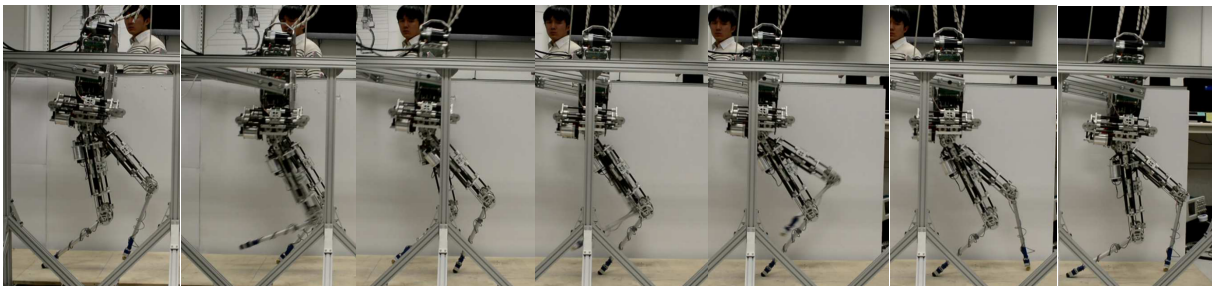


Figure 7.9: **Two step walk.** This video shows the side view of Hume's two step walk.

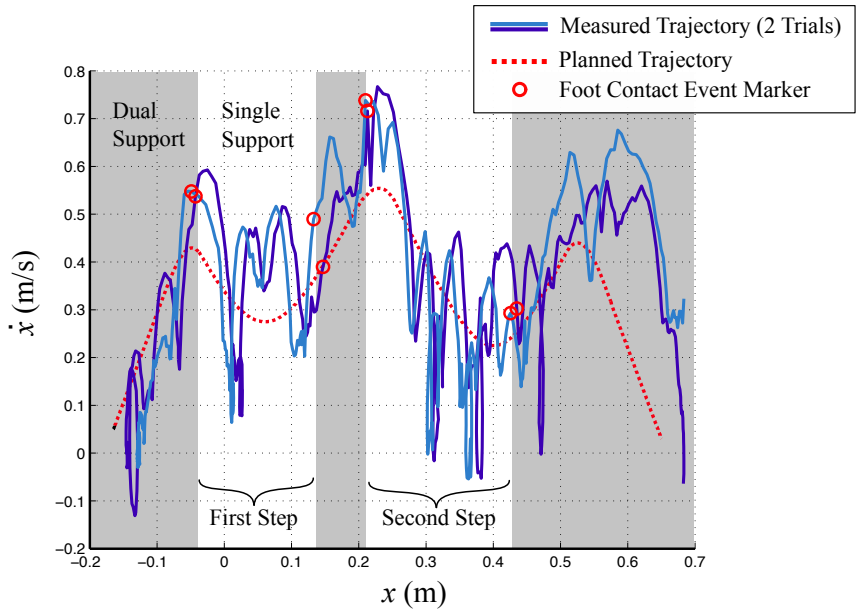


Figure 7.10: **Phase space for CoM x in the two step test.** There is a small dual contact region between steps to improve stability.

Three Steps on non-flat surface

Since our ultimate goal is making Hume walk on rough terrain, it is important to see whether Hume can step on and off of small obstacles. In this experiment, an 8 cm tall wooden block is used as an obstacle and its position is known a priori by the planner. Snapshots of this experiment are presented in the Fig. 7.11. The emergency support cable remained slack throughout the motion.

7.2 Planar Undirected Walking with Online Re-planning Method

The undirected walking experiment is designed to test the balancing ability of our system, testing both the continuous time feedback of WBOSC (Section 2.1) implementation and the discrete time feedback from the footstep planner. In this experiment, Hume continually steps forward or backward in order to remain upright despite its inherently unstable dynamics.

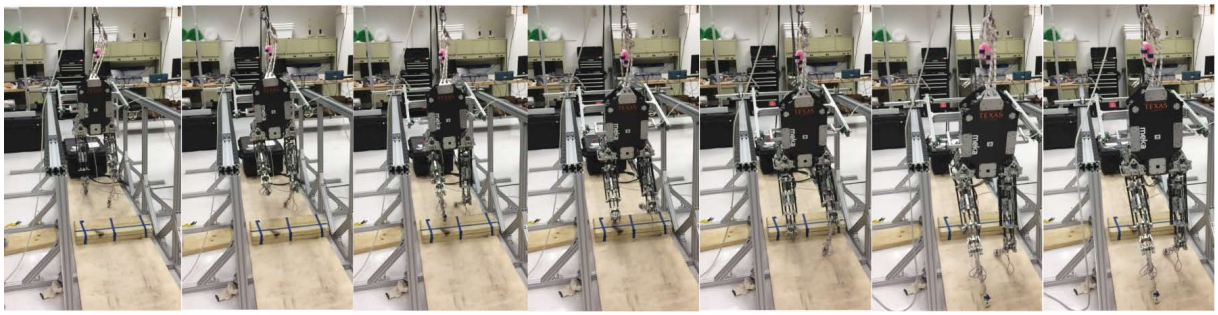


Figure 7.11: **Three step walk over an obstacle.** Hume successfully takes three steps and surmounts an obstacle. Here the planner is fully aware of the obstacle a priori.

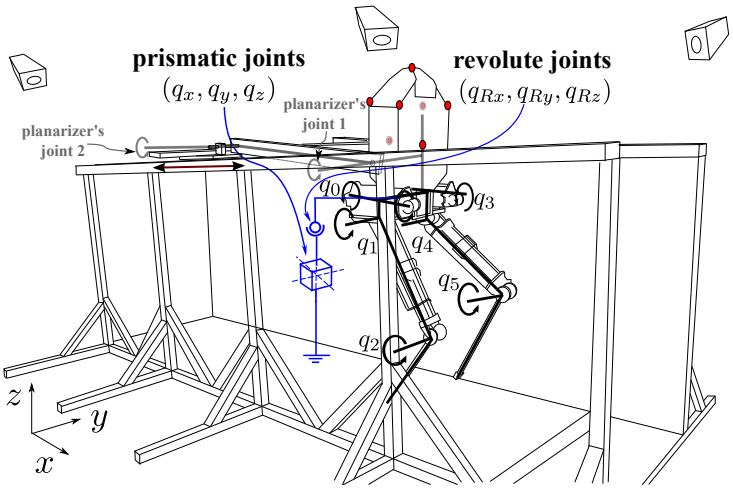


Figure 7.12: **Hume Robot Kinematics.** As when attached to the planarizing linkage. Blue schematics describe the floating base joints, while black describes the kinematics of the six leg joints and three planarizer joints. The planarizer kinematics are not included in the generalized joint vector, since they are not part of the robot's model. The locations of the LED tracking markers identified by the PhaseSpace Impulse Motion Capture system are shown as red dots.

7.2.1 Hardware Setup

The biggest change in hardware setup is the inclusion of a motion capture system to estimate robot's pose. Our poses estimation setup combines an overhead motion capture system with a low cost IMU. This sensing modality contrasts the setup found in [82] which takes advantage of its highly accurate IMU sensor. A relatively inexpensive Microstrain 3DM-GX3-25-OEM inertial measurement unit on our robot's torso measures angular velocity and linear acceleration, which is used in the state estimator. Additionally, the robot has a full overhead PhaseSpace Impulse motion capture system that gives it global coordinate information about seven uniquely identifiable LED tracking devices mounted rigidly to the torso. The PhaseSpace system produces a data stream at 480 Hz and communicates to the Linux Control PC via a custom UDP protocol. There is approximately 5 ms of delay in the feedback data. It accomplishes this using a system of eight high speed sensors mounted on the ceiling above the robot, and a proprietary software package to fuse their readings into a single estimate for the three dimensional position of each marker. On each update, the system reports the location of as many of the uniquely identifiable LED markers as it can see in Fig. 7.12.

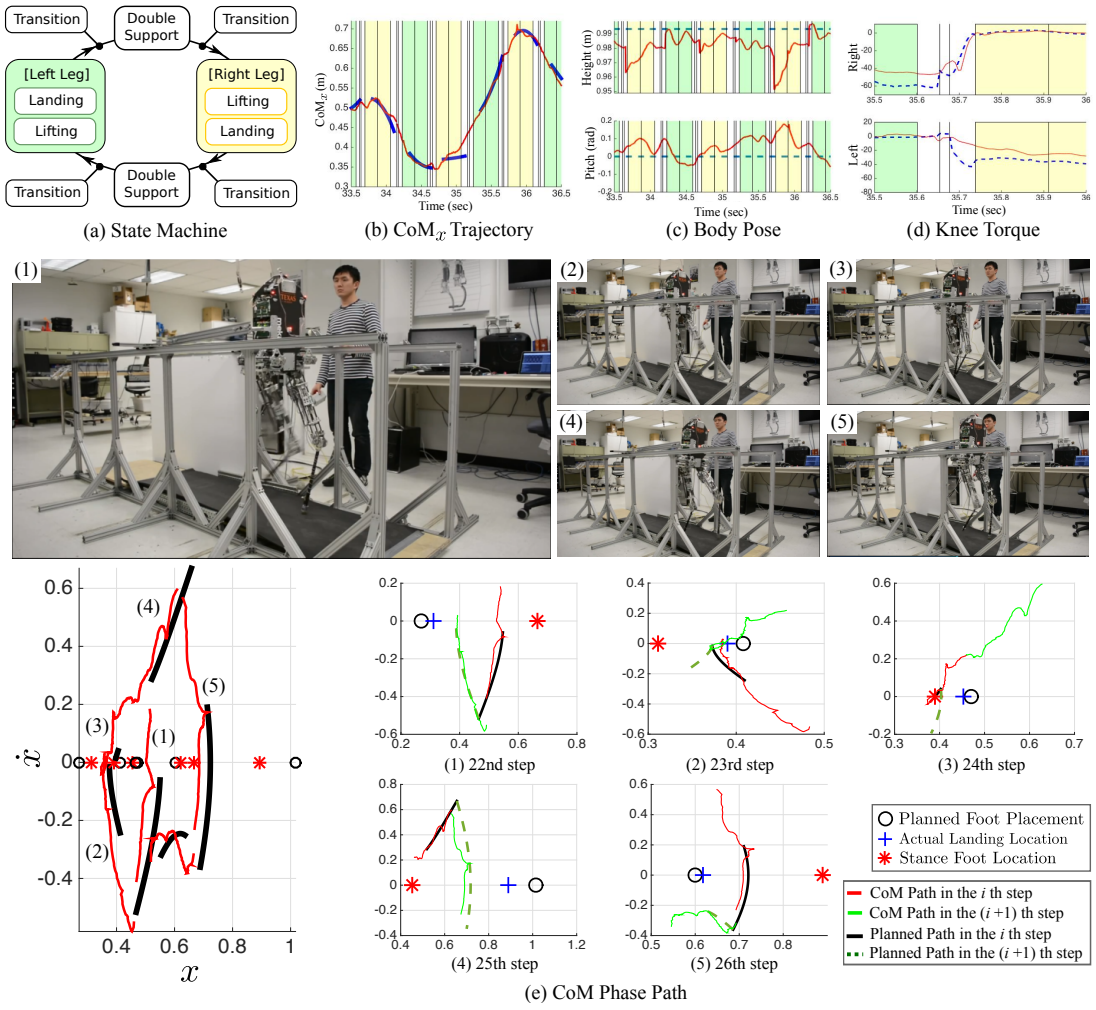
7.2.2 Experimental Result

The abduction/adduction joint of the hip, unlike the other experiments, was fixed using joint level position control to simplify the problem. The experimental setup, as well as the data from this experiment, are shown in Fig. 7.13.

Since the abduction/adduction joint is locked, $\mathbf{x}^d_{\text{task}}$ does not include roll motion control, and is thus $[\text{CoM}_z,\ q_{Ry}]^T$. In single support, it becomes $[\text{CoM}_z,\ q_{Ry},\ foot_x,\ foot_z]^T$. Additionally, since the dual support period is only $0.079s$ long, internal force feedback control is also disabled. Transitions are implemented identically to the way they were implemented in the stepping test, and as described in Sec. 2.2.1.

The difficulty of this challenge, when constrained to the sagittal plane, is highly dependent on the amount of time the robot spends in dual contact, which is naturally stable. We spend an almost trivial amount of time in dual support, limited primarily by the speed at which our contact transitions can proceed. The experimental velocity data used in Fig. 7.13 are filtered by a steady state PIP model based observer. This filtered data are used only by the planner, and are not fed back by the WBOSC implementation.

A critical test of our planning methodology is whether or not the simplified model we used to approximate the zero dynamics of the controlled system are an accurate predictor of the



(a) State Machine (b) CoM$_x$ Trajectory (c) Body Pose (d) Knee Torque

(e) CoM Phase Path

Figure 7.13: **Undirectional walking.** Subfig. (b) shows the x-directional trajectory of the CoM. The blue line represents actual data while the red lines indicate the trajectories expected by the planner. Subfig. (e) displays undirected walking. The central plot focuses on steps 23-27, of which 23-26 are expanded into individual step planning plots. In each step planning plot, a red line marks the actual CoM path up to the switching state, and a green line continues the trajectory after the switch. The robot's initial stance foot in the step planning plot is denoted with a ✱, the planned second footstep with a black circle, and the achieved second stance foot location with a blue cross. Therefore a green line in the ith step plot is the same path as the red line in the $(i+1)$th step plot. A black line and a dark green dotted line are, respectively, the PIP model's predicted paths before and after the switching state.

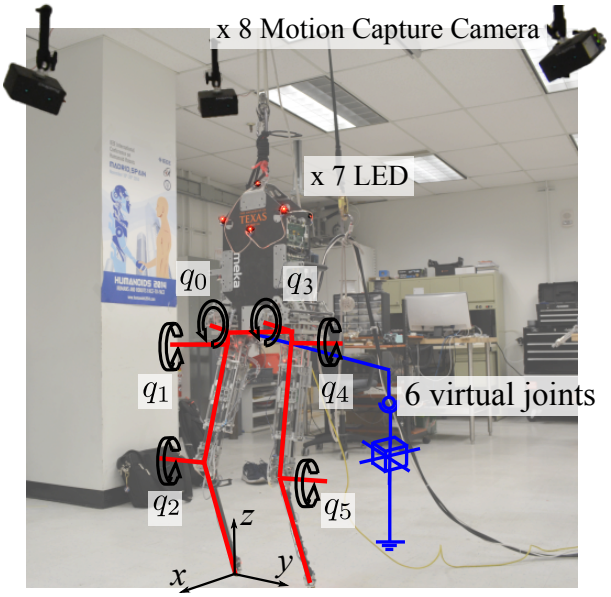


Figure 7.14: **Hume Robot Kinematics.** Blue schematics describe the floating base joints, while black describes the kinematics of the six leg joints. The locations of the LED tracking markers identified by the PhaseSpace Impulse Motion Capture system are shown as red dots.

future center of mass state. And demonstrated by Fig. 7.13, the model predicts the continued evolution of the current step accurately, and the evolution of the second step with a much larger propensity to deviate widely. This is because the landing event is a major source of uncertainty. Not only is the controller performing a transition maneuver at this point, but the landing error of the foot is not insignificant relative to the length of an average step in both space and time.

7.3 Untethered Balancing of a Point-Foot Biped Robot, Hume

Finally, we detach any tethering on Hume demonstrating untethered balancing. The hardware setup is similar to the setup presented in the previous sections except for the abduction motors, which are replaced by the same motors used in the hip and knee joints.

7.3.1 Feedback Controller Implementation

At the implementation level, WBOSC works well provided that communication latencies are sufficiently small. Achieving a 0.667 ms latency requires significant software optimization. We

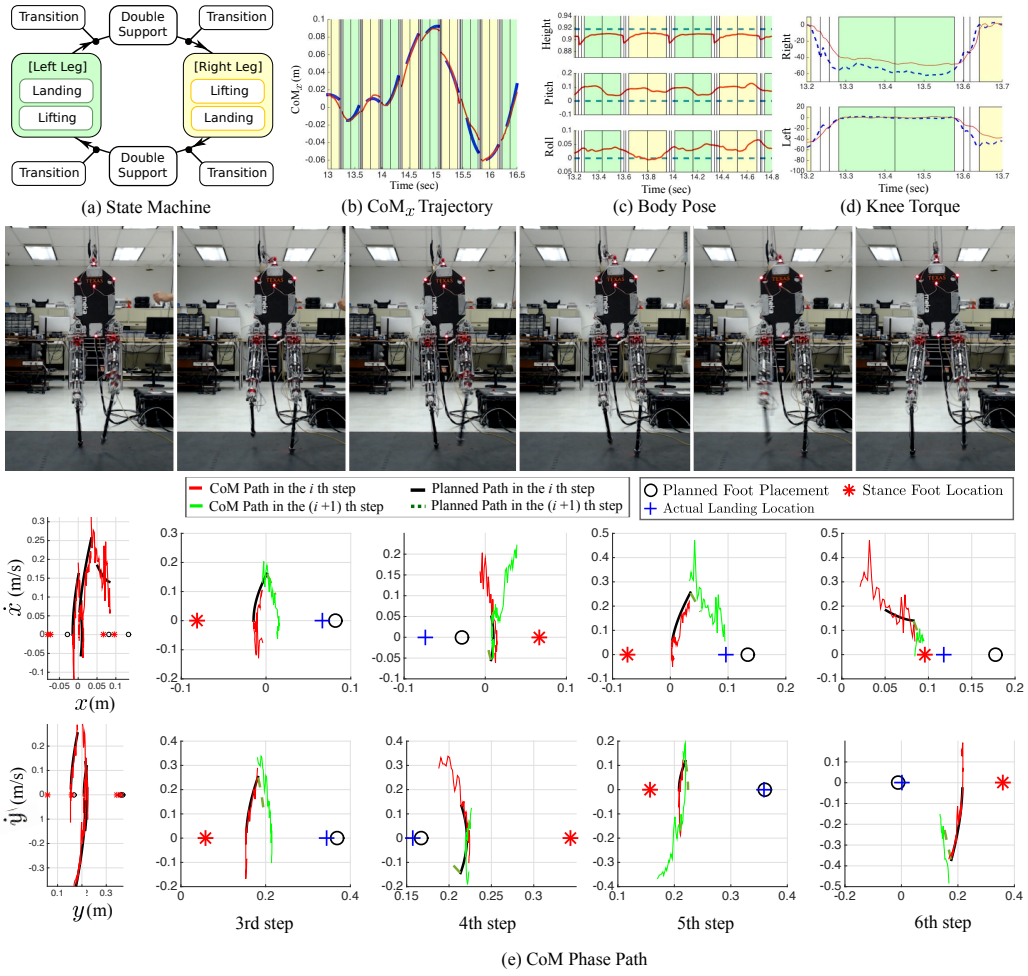
modified Meka's firmware to ensure our controller operated within a real-time context, and to incorporate it into a hierarchical chain structure that ensures minimum latency for stacked control systems.

The feedback control system is split into six joint level controllers and a centralized high level controller (see Fig. 5.1). This forms a distributed control system where the joint controllers focus on high speed actuator dynamics while the centralized controller focuses on overall system dynamics. Yet the high-level feedback is necessary to create the coupling between joints implied by operational space impedance tasks as well as regulating the internal forces between multi-contact supports. In SEA control, we kept Meka's joint torque controller, which is based on passivity as described in [103] (shown on the lower right corner of Fig. 5.1).

7.3.2 Experimental Result

To start the experiment, Hume is briefly supported while it rises to the desired height for balancing. Once it reaches its starting height, the experimenter balances the robot carefully and lets it go as it takes its first step. Once free, the robot continuously steps until it falls over. There is a harness rope, slack when the robot is at its starting height, which catches it if it falls to prevent major damage. The power and Ethernet tether hang slack from another rope.

The motion follows a time scripted state machine, shown in Fig. 7.15(a). Since the states are symmetric with respect to the supporting leg being either right or left, states are categorized in two different compound tasks with left and right single support having symmetric structures. The WBOSC compound task, $\mathbf{x}_{\text{task}}$, differs between dual support and single support phases of the stepping state machine. In dual support, the compound task coordinates are $[\text{CoM}_z,\ q_{Rz},\ q_{Ry},\ q_{Rx}]^T$, where CoM_z represents the height of the center of mass. q_{Rz}, q_{Ry} and q_{Rx} are body yaw, pitch and roll angles, respectively. Those coordinates are controlled via the acceleration input of WBOSC, u_{task}, shown in Equation (2.11) and via PD or PID control laws. In single support, the compound task is $[\text{CoM}_z,\ q_{Ry},\ q_{Rx},\ foot_x,\ foot_y,\ foot_z]^T$. The desired height is set to the initial height when Hume begins to step and the body orientation is set to be straight up. The desired foot trajectory for the lifting phase consists on first reaching a predefined z height while keeping x and y constant. Then, a 3rd order polynomial is used to generate the desired (x, y, z) landing trajectory. Since Hume's feet are points, in single support phase it is not possible to control the yaw motion, q_{Rz}. To compensate for this deficiency, we use the brief time that the robot spends in dual support to correct for it. In the balancing experiment we do not use internal force feedback control to reduce complexity of the sequencing process. Since the standing surfaces are flat, internal force regulation is not needed to keep the contacts



(a) State Machine (b) CoM$_x$ Trajectory (c) Body Pose (d) Knee Torque

(e) CoM Phase Path

Figure 7.15: **Unsupported Dynamic Balancing Experiment:** Hume accomplishes 18 steps of unsupported dynamic balancing. (a) State machine. (b) and (c) show the x and y-directional trajectory of the CoM. The blue line represents actual data while the red lines indicate the trajectories estimated by the planner. (d) Position and orientation task tracking: sensor data in red and desired values, dotted, in blue. Height refers to CoM height. (e) The snapshots and data of steps 3-6: The phase paths of steps 3-6 are expanded into individual step planning plots. For each step, a red line marks the actual CoM path up to the switching state, and a green line continues the trajectory after the switch. The robot's initial stance foot in the step planning plot is denoted with a ✱, the planned second footstep with a black circle, and the achieved second stance foot location with a blue cross. Therefore, a green line in the ith step plot is the same path as the red line in the $(i+1)$th step plot. A black line and a dark green dotted line are, respectively, the PIP model's predicted paths before and after the switching state.

Position Gain

Dual Support				
	CoM_z	q_{Rz}	q_{Ry}	q_{Rx}
$K_x(1/\mathrm{s}^2)$	270.0	100.0	200.0	210
$I_x(1/\mathrm{s}^3)$	20.0	0.0	20.0	35.0
$D_x(1/\mathrm{s})$	10.0	0.0	15.0	10.0

Single Support					
CoM_z	q_{Ry}	q_{Rx}	$foot_x$	$foot_y$	$foot_z$
270.0	200.0	210.0	220.0	237.0	400.0
20.0	20.0	35.0	35.0	40.0	35.0
10.0	15.0	10.0	35.0	40.0	60.0

Torque Gain

	Stance Leg		
	Both Abduction	Both Hip	Both Knee
$K_{P,\tau}(\mathrm{N\,m\,rad/s})$	2.84	43	22
$K_{I,\tau}(\mathrm{N\,m\,rad/s})$	0.15	0	0

Swing Leg				
Both Abduction	Right Hip	Left Hip	Right Knee	Left Knee
3.09	58.57	32.70	49.75	26.1
0.19	5.86	3.48	6.85	4.15

Table 7.1: **Gain Set for the Unsupported Dynamic Balancing Experiment**

Transition	Lifting	Landing	Dual Support
0.024 (s)	0.145 (s)	0.15 (s)	0.016 (s)
κ_x, κ_y	t'_x, t'_y	λ_x	λ_y
0.4, 0.4	0.185, 0.18	0.7, -0.88	0.69, -0.81

Table 7.2: **Planner Parameters for the Unsupported Dynamic Balancing Experiment**

stable. All control parameters for the single and dual support phases, and for the stance and swing legs are shown in Table 7.1. The parameters of the planner used in this experiment are shown in Table 7.2.

In Fig. 7.15(b) and (c), The x and y directional CoM trajectories (red) are superimposed on the one-step predicted path by our planner (blue). Although predicting CoM path for multiple steps is difficult, computing the CoM path for a single step using the PIP model closely approximates the actual CoM motion. The orientation error is bounded by 0.05 rad (Fig. 7.15(d)). Given the model disturbances and the impacts, this error is reasonable small to validate the controller's performance. Fig. 7.15(e) shows snapshots and phase paths for this experiment. The phase space data is corrupted by high frequency noise from the IMU sensor signal and the joint encoders that combined to compute CoM velocity.

Transition and gain scheduling techniques also play an important role by smoothing the motion and tracking the tasks. In Fig. 7.15(d), the commanded torque (blue) changes from 0 to $-60\,\mathrm{N\,m}$ without significant jerk due to our contact transition technique. When the right leg switches to a stance leg (green background), around $20\,\mathrm{N\,m}$ of torque tracking error appears. This is expected because we detune the low-level torque controller to achieve stiffer position control by the WBOSC controller. The controller corrects yaw error during the dual support phase. Additionally, Hume incurs a significant bending of the stance leg which results on uncertainty of the position of the CoM with respect to the stance foot. Despite all these sources of error our robot was able to dynamically balance unsupported for 18 steps using its point contacts.

7.3.3 Discussion

The central focus of the study in this section is on formulating a WBOSC and a new balance planner to achieve unsupported dynamic balancing of a point foot biped robot without passive feet. The underactuated nature of point foot bipeds forces us to "give up" on the x and y

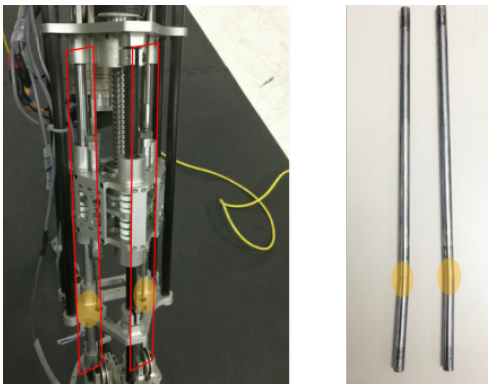


Figure 7.16: **Bent Steel Tube.** We found that the steel tubes in thigh linkages are bent a month later. This shows how much forces are posed on the structure.

components of CoM motion, leaving those aspects to evolve naturally while the WBOSC controls the remaining degrees of freedom. These liberated degrees of freedom are still controlled, but must be controlled indirectly, on a step by step basis, by choosing the footstep locations. By formulating a WBOSC and a phase-space-based foot placement planner we make Hume, our 6-DoF point-foot biped robot with series elastic actuators, balance on rough terrain when constrained to move in the robot's sagittal plane and take 18 steps on flat terrain when unsupported. In addition, our framework also demonstrates its ability to step over a 7cm obstacle in simulation. Achieving these capabilities required advancements in WBOSC that push its performance boundaries to the next level by stabilizing highly dynamic biped robots and showing the benefits of employing feedback control of internal forces.

An important issue has been the integration of WBOSC with joint level torque controllers. To obtain good performance, torque control at the joint level has been sought. The reason is that joint torque control reduces the effect of the natural dynamics of the actuator, i.e. stiction, dynamic friction, non-linearities, thus yielding excellent force behaviors for the actuator. However, using torque-feedback control nested within a position feedback controller needs careful understanding of its stability as a function of loop latencies [112]. Torque controllers substantially decrease the effect of natural friction of the actuators, forcing the task position controllers to provide damping via velocity feedback. However, velocity feedback is very sensitive to loop latencies as was analyzed in detail in [111]. In order to achieve stiff task position control, which is desirable for some tasks to reject modeling disturbances, system latencies both due to the WBC and to communications, need to be minimized. In the case of Hume, we have greatly optimized all control computations and embedded communications to decrease the servo round

trip, including communications, to 0.667 ms.

Additionally, when there is a need to deliver stiff position tracking control, it is desirable to detune or reduce torque gains. This effect was reported in [8] and more recently analyzed in depth in our paper [112]. In essence, we prioritize position accuracy over torque accuracy when position tracking is essential. To compensate for the detuned torque controllers, when our biped balanced on the split terrain, we added a force feedback loop that regulated the internal force from joint torque sensor. This feedback loop mitigated the effects of trading off motion accuracy versus torque accuracy. The key advantage is that increasing internal force control gains does not affect the task position control performance whereas increasing torque gains reduces task position performance.

Finally, exact state measurement has been crucial to maneuvering with the highest possible accuracy given the hardware limitations of our low cost IMU. However, many successful legged robots, such as Atlas, ATRIAS, and the MIT Cheetah 2 robot, use an optical IMU called KVH Industries Model 1750, which has a minuscule bias error, $0.000\,55°/s$. This is not the case in our experimental setup, as we used a low cost and much less accurate MEMS IMU, i.e. the Microstrain 3DM-GX3-25-OEM, with a bias error 500 times worse than the previous one, i.e. $0.25\,°/s$. In other words, our IMU's orientation estimate drifts away quickly. In practice, the drift appears to be much faster than the specification. To compensate for this discrepancy, we devised a sensor fusion approach by combining the IMU with visual data from our motion capture system. Nonetheless, the overall pose sensing system is far less accurate and speedy than using the high end IMU. In the future we plan on replacing our current IMU with the high end version KVH IMU and also redesigning part of the leg joints to increase mechanical rigidity. The mechanical rigidity must be well addressed because the test poses severe stresses on robot's structure as we can see in Fig. 7.16. With those improvements we believe that we will substantially increase the accuracy of the foot positioning, enabling faster and more robust locomotion behaviors.

Chapter 8

Concluding Remarks

Throughout this dissertation, we have built robust whole-body control architecture integrating joint-level controllers, high-level whole-body motion coordinators, and locomotion planners. In this final chapter, we summarize the key points of our work and provide recommendations for hardware experiments.

8.1 Summary of Results and Discussion

We have developed new WBC formulations and new algorithms to address various issues in WBC implementation. Not only the real-time feedback controllers but also locomotion controllers (planners) have been investigated. Fortunately, we have had a chance to test our algorithms with various physical systems, which provides us a comprehensive understanding of the practical problems. We also released our open-source software including all methods presented in this dissertation. Lastly, we would like to share the lessons we learned from our hardware tests.

8.1.1 Enhancement of Whole-Body Control

We have developed a new whole-body controller capable of managing prioritized task executions, inequality constraints, and slip conditions. Also, we have formulated new tasks, which are the centroidal momentum task and the capture point task. In terms of implementation, we have devised, analyzed, and experimentally validated various control methods such as sensor-based internal force feedback control (Section 5.1.2), torque control using disturbance observer and motor velocity feedback (Section 5.3), and collocated feedback control (Section 5.4). We have tested our algorithms with multiple systems: Hume, NAO, Draco P1, and Mercury. The implementations with multiple systems provide convincing evidence of the transferability of WBC.

8.1.2 Locomotion Planners for Agile and Robust Walking

We have developed a new foot placement planner for point-foot biped systems to maintain their balance with continuous stepping. The stability of the planner is theoretically proved, and simulation and experimental results support it. Another important development is a locomotion planner trained through a reinforcement learning method. The training results are encoded in neural networks, so we can instantaneously obtain optimal walking patterns whenever we need. This reinforcement learning-based planner provides fast re-planning, which significantly robustifies locomotion.

8.1.3 Open-source Software

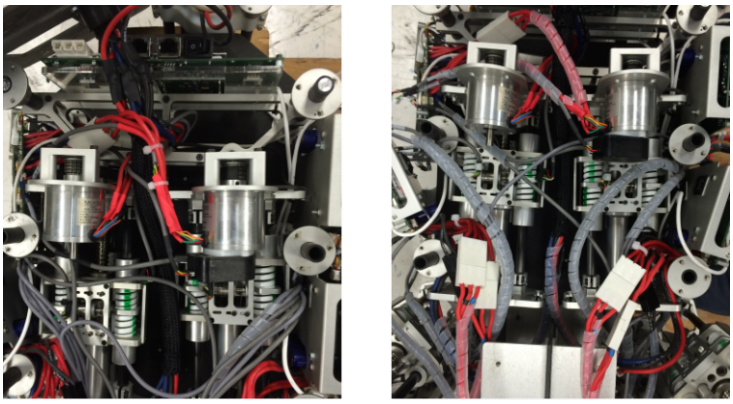
After many iterations, we released our dynamic control software package, "Sejong Dynamic Control Toolkit." It is open-source, so anyone can access and download the code at `https://github.com/dhkim0821/Sejong_Dynamic_Control_Toolkit`. The program is written with C++ and includes all WBCs presented in this dissertation. The primary dynamics engine is RBDL[1], but with an additional function we made, which is the time-derivative of Jacobians. The software package also has a dynamic simulator (a modified version of the open-source library, srLib[2]).

One benefit of the package is a small dependency on external packages. OpenGL, which is used in the dynamic simulator, is the only external library required, and all the other packages are self-contained. Therefore, users can simply compile and run the program without a cumbersome installation process. This small dependency is also beneficial in the usage of our software on other platforms. For example, we do not need to worry about the ROS version since there is no dependency with ROS in our package.

8.1.4 Small But Important Details in Experiments

Many critical issues of hardware experiments are actively discussed in various literature, but I would like to leave my personal opinions and recommendations in this section. Indeed, many lessons obtained from our real-world experiments are not suitable to be published in academic papers, since most of them are not theoretically meaningful. As a student who has gone through

[1]Rigid Body Dynamics Libary. `https://rbdl.bitbucket.io`
[2]Seoul National University Robotics Library. `http://robotics.snu.ac.kr/srlib/`



(a) Tangled cables (b) Organized cables

Figure 8.1: **Organization of Tangled Cables.** (a) When we open the back of Hume's torso, we could see tangled cables and some of them are pinched when actuators move. (b) The cables are organized and covered with a plastic wrapper to avoid scratches caused by moving components.

the novice mistakes, I would like to point out several practical issues which are not explained in the main contents of this dissertation.

First of all, I would like to emphasize maintenance issues in hardware. In our experiments, the robots have to experience high impacts, so the external disturbances damage hardware components. The most annoying elements are connectors and cables. Many of Hume's cables are tangled and pinched inside its torso (Fig. 8.1). I would like to say that it is a miracle that every actuator and every sensor in your robot correctly operates at the same time, although operation is the most basic requirement for research. However, while doing rough experiments multiple times, the connectors are easily loosened, and cables can be damaged or pinched. A single cable or connector issue can ruin entire systems. In the hardware upgrade of Hume (Mercury), we gave special cared about this issue and fabricated cases of micro-controllers to hold all input-output cables. At the same time, we made multiple walls to check sensor signals for easy detection of damaged components. This additional work significantly reduced the development time by keeping Mercury's electronics operational.

Secondly, debugging tools must be well organized and easy to use. One of the central works of my Ph.D. is developing handy debugging tools. The works are represented by data saving programs, management of data repositories, visualization of the data, objective-oriented programming, and well-organized hardware tool boxes (including hex-key sets). The statement that I strongly believe is "The idea is cheap, but data are expensive." So far, this sentence

has always been correct because I spent significantly more time to experimentally validate algorithms than to invent, devise, and test the algorithms in a simulation. Therefore, after idea generations, hundreds of days are needed to obtain meaningful results in the real world, and the major portion of the time will be dedicated to debugging. To win in this long and tedious game, we need to produce high-grade debugging tools. If there are graduate students who hesitate to spend their time to make plotting tools better, I would recommend that they devote their effort to write the programs. The time spent will eventually reduce the cost to achieve the data that they want.

The third issue I would like to point out is sensor calibration. Unsurprisingly, Hume's joint encoders kept rotating during tests. I had to re-calibrate knee joint encoders, which are near to the impact points (feet), once per 20 trials. Since knowing the exact state is critical to successful locomotion, any small bias in sensor signals can cause a test to fail. Most commercial robots have auto-calibration function and calibrate joint position whenever the systems start (e.g., NAO robots always start from the same posture for calibration). However, I overlooked the benefit of devising an auto-calibration program when I first worked with Hume. After our lab colleague made a program to calibrate every joint position at the same time, I could speed up the experiment progress significantly.

8.2 Future Work

More agility. We plan to challenge biped running with our upgraded biped robot, Mercury. We also have a plan to demonstrate jumping with the fully-assembled Draco P1 robot. Next year, we are going to have a new biped, Draco P2. All joints of the new biped will be viscoelastic liquid cooled actuators, so we believe that we can demonstrate agile locomotion while carrying large payloads.

Smarter locomotion. I started studying machine learning around the end of my Ph.D., so there is still a vast area to explore. Recently, some of our lab colleagues start investigating machine learning as their Ph.D. topic, and I am collaborating with them to employ useful techniques for locomotion planning. We desire to see significant synergy, as we saw in our robust locomotion planner trained through reinforcement learning.

Dynamic Motion Involving Multi-Contacts. During my Ph.D., I did not have a chance to extensively test robots with arms; therefore, it was hard to demonstrate multi-contact scenarios in physical systems, although the developed algorithms are suitable to control dynamic multi-contact motions. Indeed, we had planned to build a robot with an arm as depicted in

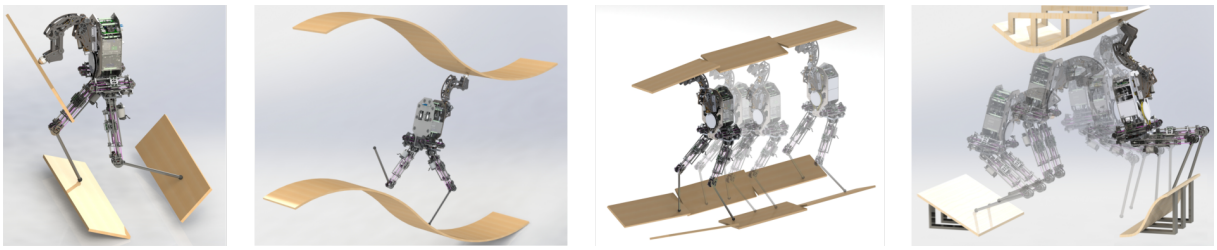


Figure 8.2: **Multi-Contact behaviors** The listed pictures are conceptual illustrations of dynamic motions involving multi-contacts.

Fig. 8.2. We desire to demonstrate the various multi-contact behaviors that are conceptually illustrated in Fig. 8.2.

8.3 Summary of Publications

Although some of the works presented in this dissertation is not yet published, here is the list of some publications generated during this thesis.

Peer reviewed jounal paper:

· D. Kim, Y. Zhao, G. Thomas, B. R. Fernandez, and L. Sentis, "Stabilizing Series-Elastic Point-Foot Bipeds Using Whole-Body Operational Space Control," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 13621379, 2016.

Peer reviewed conference papers:

1. D. Kim, O. Campbell, J. Ahn, L. Sentis, and N. Paine, "Investigations of Viscoelastic Liquid Cooled Actuators Applied for Dynamic Motion Control of Legged Systems," IEEE-RAS 17th International Conference on Humanoid Robots (Humanoids), 2017

2. J. Liu, Y. Zhao, D. Kim, O. Khatib, L. Sentis, "Locomotion Control of Three Dimensional Passive-Foot Biped Robot Based on Whole Body Operational Space Framework," IEEE International Conference on Robotics and Biomimetics (ROBIO), 2017

3. D. Kim, S. J. Jorgensen, P. Stone, and L. Sentis, "Dynamic behaviors on the NAO robot with closed-loop whole body operational space control," IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), 2016, pp. 11211128.

4. D. Kim, G. Thomas, and L. Sentis, "A method for dynamically balancing a point foot robot," IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), 2015, pp. 901907.

5. Y. Zhao, D. Kim, G. Thomas, and L. Sentis, "Hybrid multi-contact dynamics for wedge jumping locomotion behaviors," the Proceedings of the 18th International Conference on Hybrid Systems Computation and Control (HSCC), 2015.

6. D. Kim, G. Thomas, and L. Sentis, "Continuous Cyclic Stepping on 3D Point-Foot Biped Robots Via Constant Time to Velocity Reversal," The 13th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 2014.

7. D. Kim, Y. Zhao, G. Thomas, and L. Sentis, "Empirical Modifications to a Phase Space Planner Which Compensates for Low Stiffness Actuation in a Planar, Point-Foot, Biped Robot," ASME Dynamic Systems and Control Conference (DSCC), 2014, p. V001T11A001.

8. Y. Zhao, D. Kim, B. Fernandez, and L. Sentis, "Phase space planning and robust control for data-driven locomotion behaviors," IEEE-RAS 13th International Conference on Humanoid Robots (Humanoids), 2013, pp. 8087.

Preprinted papers:

1. D. Kim, J. Lee, and L. Sentis, "Robust Dynamic Locomotion via Reinforcement Learning and Novel Whole Body Controller," arXiv.org, vol. cs.RO. 07-Aug-2017.

2. S. J. Jorgensen, O. Campbell, T. Llado, D. Kim, J. Ahn, and L. Sentis, "Exploring Model Predictive Control to Generate Optimal Control Policies for HRI Dynamical Systems," arXiv.org, vol. cs.HC. 13-Jan-2017.

3. D. Kim, Y. Zhao, G. Thomas, and L. Sentis, "Assessing Whole-Body Operational Space Control in a Point-Foot Series Elastic Biped: Balance on Split Terrain and Undirected Walking," arXiv.org, vol. cs.RO. p. 2855, 12-Jan-2015.

Appendices

Appendix A

Quaternion

One of the difficult apects of quaternion representation is that there are multiple ways to describe orientation and operations between the orientations. We clarify the convention used in this paper here to avoid the confusion. Rotation is represented by a rotation axis ($\mathbf{v}$) and with a rotation amount (θ) about the axis. In this appendix, we construct a quaternion as

$$\begin{aligned}\mathbf{q} =& [\ q_w,\ q_x,\ q_y,\ q_z\ ] \\ =& [\ \cos(\frac{\theta}{2}),\ \frac{\mathbf{v}}{|\mathbf{v}|}\sin(\frac{\theta}{2})\ ].\end{aligned} \tag{A.1}$$

The rotation matrix according to the same rotation is defined by

$$\boldsymbol{R}[\mathbf{q}] = \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2q_xq_y-2q_zq_w & 2q_xq_z+2q_yq_w \\ 2q_xq_y+2q_zq_z & 1-2q_x^2-2q_z^2 & 2q_yq_z-2q_xq_w \\ 2q_xq_z-2q_yq_w & 2q_yq_z+2q_xq_w & 1-2q_x^2-2q_y^2 \end{bmatrix} \tag{A.2}$$

With the definition, we obtain the relationship $\mathbf{q} \otimes Q[\mathbf{v}] \otimes \mathbf{q}^* = Q[\boldsymbol{R}[\mathbf{q}]\mathbf{v}]$, where $\otimes$ is the Hamiltonian multiplication and $Q[\mathbf{x}]$ represents the mapping from a three dimensional vector to a pure imaginary quaternion.

Appendix B

Prismatic Inverted Pendulum Dynamics

In the proposed planner, numerical integration starts with $\ddot{z}(x(n),\ ,y(n))$ as described in Fig. 4.13. The PIP model can be expressed as the differential equation

$$\ddot{x} = \frac{g+\ddot{z}}{z}(x - x_p),$$
$$\ddot{y} = \frac{g+\ddot{z}}{z}(y - y_p). \tag{B.1}$$

Accounting for z being a function of x, y, the height surface,

$$\frac{dz}{dt} = \frac{\partial z}{\partial x}\dot{x} + \frac{\partial z}{\partial y}\dot{y}, \tag{B.2}$$

$$\frac{d^2z}{dt^2} = \frac{d}{dt}\Big(\frac{\partial z}{\partial x}\Big)\dot{x} + \frac{\partial z}{\partial x}\ddot{x} + \frac{d}{dt}\Big(\frac{\partial z}{\partial y}\Big)\dot{y} + \frac{\partial z}{\partial y}\ddot{y}, \tag{B.3}$$

$$\ddot{z} = \frac{\partial^2 z}{\partial x^2}\dot{x}^2 + \frac{\partial z}{\partial x}\ddot{x} + \frac{\partial^2 z}{\partial y^2}\dot{y}^2 + \frac{\partial z}{\partial y}\ddot{y} + 2\frac{\partial^2 z}{\partial x \partial y}\dot{x}\dot{y}. \tag{B.4}$$

By plugging Equation (B.1) into Equation (B.4), we obtain,

$$\ddot{z} = \frac{\partial^2 z}{\partial x^2}\dot{x}^2 + \frac{\partial z}{\partial x}\frac{g+\ddot{z}}{z}(x - x_p) + \frac{\partial^2 z}{\partial y^2}\dot{y}^2 + \frac{\partial z}{\partial y}\frac{g+\ddot{z}}{z}(y - y_p) + 2\frac{\partial^2 z}{\partial x \partial y}\dot{x}\dot{y}, \tag{B.5}$$

$$\Big(1 - \frac{g}{z}\big(\frac{\partial z}{\partial x}(x - x_p) + \frac{\partial z}{\partial y}(y - y_p)\big)\Big)\ddot{z} = \tag{B.6}$$

$$\frac{\partial^2 z}{\partial x^2}\dot{x}^2 + \frac{\partial z}{\partial x}\frac{g}{z}(x - x_p) + \frac{\partial^2 z}{\partial y^2}\dot{y}^2 + \frac{\partial z}{\partial y}\frac{g}{z}(y - y_p) + 2\frac{\partial^2 z}{\partial x \partial y}\dot{x}\dot{y}, \tag{B.7}$$

$$\ddot{z} = \frac{\frac{\partial^2 z}{\partial x^2}\dot{x}^2 + \frac{\partial z}{\partial x}\frac{g}{z}(x - x_p) + \frac{\partial^2 z}{\partial y^2}\dot{y}^2 + \frac{\partial z}{\partial y}\frac{g}{z}(y - y_p) + 2\frac{\partial^2 z}{\partial x \partial y}\dot{x}\dot{y}}{1 - \frac{g}{z}\big(\frac{\partial z}{\partial x}(x - x_p) + \frac{\partial z}{\partial y}(y - y_p)\big)} \tag{B.8}$$

Appendix C

Viscoelastic Material Characterization

In this section, we shortly introduce the elastomers we have analyzed and empirically test to choose the most proper material for viscoelastic liquid cooled actuators. We put this section in the appendix because the tests and analyses are done by Dr. Nick Paine, who is CTO of Apptronik, and my role is just documenting it. The detail analysis can be found in [43].

The primary driver for using elastomers instead of metal springs is to benefit from their intrinsic damping properties. However, the mechanical properties of viscoelastic materials can be difficult to predict, thus making the design of an actuator based on these materials a challenging endeavor. The most challenging aspect of incorporating elastomers into the structural path of an actuator is in estimating or modeling their complex mechanical properties. Elastomers possess both hysteresis and strain-dependent stress, which result in nonlinear force displacement characteristics. Additionally, elastomers also exhibit time-varying stress-relaxation effects when exposed to a constant load. The result of this effect is a gradual reduction of restoration forces when operating under a load. A third challenge when using elastomers in compression is compression set. This phenomenon occurs when elastomers are subjected to compressive loads over long periods of time. An elastomer that has been compressed will exhibit a shorter free-length than an uncompressed elastomer. Compression set is a common failure mode for o-rings, and in our application, it could lead to actuator backlash if not accounted for properly.

To address these various engineering challenges we designed experiments to empirically measure the following four properties of our viscoelastic springs: 1) force versus displacement, 2) stress relaxation, 3) compression set, and 4) frequency response, which will be used to characterize each material's effective viscous damping. We selected and tested the seven candidate materials that are listed in Table C.1.

C.1 Force versus displacement

In the design of compliant actuation, it is essential to know how much a spring will compress given an applied force. This displacement determines the required sensitivity of a spring-

Materials	Hardness (A)	Tensile Strength (kPa)	Material Cost ($)
Abrasion-resistance polyurethane	90	37920	19.40
Fabric-reinforced silicone	70	8960	29.08
Ultra-strength oil-resistance Buna-N	90	24130	51.47
Viton Fluoroelastomer	75	10340	105.62
Abrasion-resistance polyurethane	80	43780	19.40
High-strength weather resistance EPDM	80	10340	35.28
High-temperature silicone	90	5170	29.41

Table C.1: Viscoelastic Material Candidates

deflection sensor and also affects mechanical aspects of the actuator such as usable actuator range of motion and clearance to other components due to Poisson ratio expansion. In this experiment, we identify the force versus displacement curves for the various elastomer springs. Note that there is a disagreement between our empirical measurements and the analytic model relating stiffness to hardness, i.e. the Gent's relation shown in [76]. This mismatch arises because in our experiments the materials are preloaded whereas the analytical models assume unloaded materials.

C.2 Stress relaxation

Stress-relaxation is an undesirable property in compliant actuators for two reasons. First, the time-varying force degrades the quality of the compliant material as a force sensor. When a material with significant stress-relaxation properties is used, the only way to accurately estimate actuator force based on deflection data is to model the effect and then pass deflection data through this model to obtain a force estimate. This model introduces complexity and more room for error. The second reason stress-relaxation can be problematic is that it can lead to the loss of contact forces in compression-based spring structures.

The experiment for stress relaxation is conducted as follow: 1) enforce a desired displacement

to a material, 2) record the force data over time from the load cell, 3) subtract the initially measured force from all of the force data. Empirically measured stress-relaxation properties for each of the materials represent force offsets as time goes under the same displacement enforced. Note that each material shows different initial force due to the different stiffness and each initial force data is subtracted in the plot.

C.3 Compression set

Compression set is the reduction in length of an elastomer after prolonged compression. The drawback of using materials with compression set in compliant actuation is that the materials must be installed with larger amounts of preload forces to avoid the material sliding out of place during usage. To measure this property, we measured each elastomers free length both before and after the elastomer was placed in the preloaded testbed.

C.4 Dynamic response

In regards to compliant actuation, the primary benefit of using an elastomer spring is its viscous properties, which can characterize the dynamic response of an actuator in series with such a component. To perform this experiment, we generate motor current to track an exponential chirp signal, testing frequencies between 0.001Hz and 200Hz. Given the input-output relation of the system, we can fit a second order transfer function to the experimental data to obtain an estimate of the system's viscous properties. However, this measure also includes the viscoelastic testbed's ballscrew drive train friction. To quantify the elastomer spring damping independent of the damping of the testbed drive train, the latter (8000 N s/m) was first characterized using a metal die spring, and then subtracted from subsequent tests of the elastomer springs to obtain estimates for the viscous properties of the elastomer materials.

C.5 Selection of Polyurethane 90A

In addition, a variety of other experiments were conducted to strengthen our analysis and are summarized in Table C.2. Based on these results, Polyurethane 90A appears to be a strong candidate for viscoelastic actuators based on its high linearity (0.992), low compression set (2%), low creep (15%), and reasonably high damping (16000 N s/m). It is also the cheapest of the materials and comes in the largest variety of hardnesses and sizes.

Materials	Compression set (%)	Linearity (R-square)	Linear stiffness (N/mm)	Material damping (N s/m)	Creep (%)
Spring steel	0	0.996	860.8	0	0
Polyurethane 90A	2	0.992	8109	16000	15.3
Reinforced silicone 70A	2.7	0.978	57570	242000	
Buna-N 90A	2.8	0.975	11270	29000	25
Viton 75A	4	0.963	2430	9000	30.14
Polyurethane 80A	4.5	0.993	2266	4000	16.8
EPDM 80A	6.48	0.939	6499	16000	23.4
Silicone 90A		0.983	12460	37000	10.7

Table C.2: Viscoelastic Material Experiment Results

Appendix D

Efficiency Analysis of VLCA

The ratio of mechanical power output (force $\times$ velocity) to electrical power input (voltage $\times$ current) is widely used as a metric to characterize the efficiency of actuators, but it is informative to accurately decompose this efficiency measure into the multiple contributions of the major components of an actuator. For example, different electric motors show different efficiencies with respect to the input current and voltage. To correctly characterize drivetrain efficiency, we present both the ratio of mechanical power output versus the electric motor's power and mechanical power output versus the power supply's power input, all shown in Fig. D.1.

Fig. D.1 explains the power flow from the power supply to the robot joint. Input current (I_b) and voltage (V_b) are measured in the micro-controllers and the product of those two yields the input power from the power supply. $\dot{\theta}_m$ is measured by the quadrature encoder connected to the motor's axis (Fig. 5.7(f)) and τ_m is computed from $k_\tau i_m$ with i_m measured in the micro-controller. Joint velocity is also computed based on the quadrature encoder of the motor to avoid errors emanating from any backlash at the joint. The torque (τ_k) is computed from projecting the load cell data across the linkage's effective moment arm.

In this test, the leg lifts a 23kg load using five different durations to observe efficiency over a range of different speeds and torques. The results are presented in Fig. D.1 with the description of three different power measures. The sensed torque data measured by a load cell is noisy; therefore, we compute the average of the drivetrain efficiency for a clearer comparison. The averages are the integrations of efficiency divided by the time durations. Here we only integrate efficiency while the mechanical power is positive, to prevent confounding our results by incorporating the work done by gravity.

The experimental results show that the drivetrain efficiency is approximately 0.93, which means that we lose only a small amount of power in the drivetrain and most of the torque from the motor is delivered to the joint. This high efficiency indicates only minor drivetrain friction, which is beneficial for dynamics-based motion controllers.

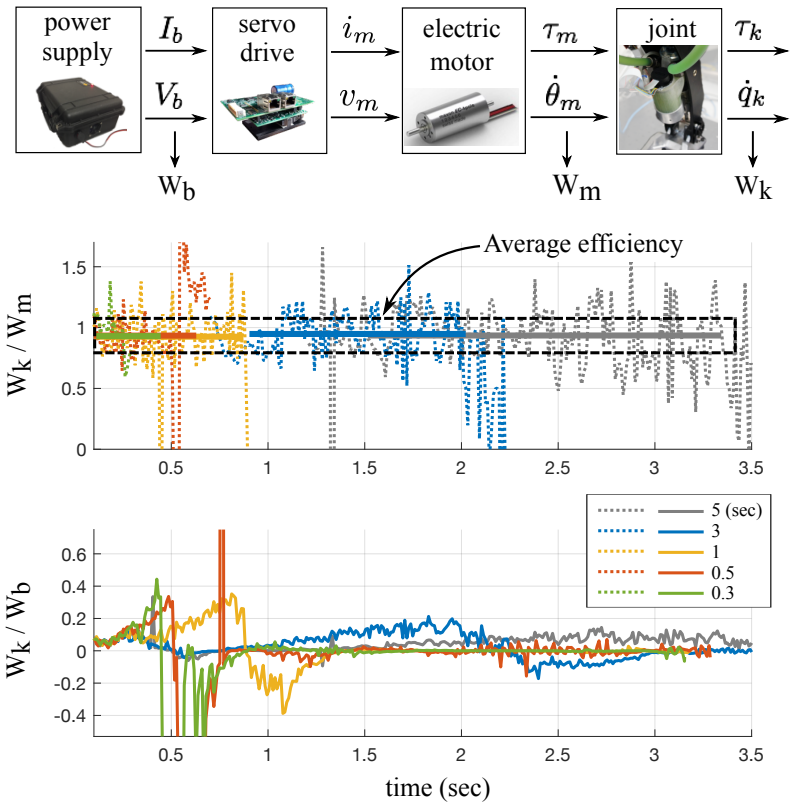


Figure D.1: **Efficiency Analysis of The Ankle Actuator.** Efficiencies of mechanical system using electrical power has 3 steps from a power supply to robot joints. The graph shows the ratio of the mechanical power of the ankle joint and the motor power and the ratio of the joint power and power supply's input power.

Appendix E

Body Orientation Estimation Using IMU and Motion Capture Data

The body orientation estimation used in Hume's dynamic balance experiment (Section 7.3) is designed and implemented by Thomas Gray. This section is the detailed explanation of his work, which has not been described in other works. The main idea of the estimation is combining the motion capture data and IMU signal to overcome the low-quality signal of IMU and slow update rate of the motion capture system. The controller needs a body orientation estimate every servo cycle, 0.667 ms, yet the motion capture system updates at only 480 Hz, occasionally fails to track a subset of the markers, and has a processing delay. When the motion capture position update arrives, the new best estimate of the orientation at the instant in the past corresponding to the delayed sensor data is found. We maintain a list of recent IMU measurements, and calculate a new estimate of the current robots orientation by integrating the angular velocities.

We use least squares to minimize the distance between motion capture LED positions $\hat{y}_i^k \in \mathbb{R}^3$ and predicted motion capture LED position $\tilde{y}_i^k \in \mathbb{R}^3$ for $i = 1, \ldots, n$, where n is typically 7, but decreases when LEDs are blocked, and where k represents the time in the past associated with the delayed motion capture data. Our model predicts motion capture LED locations based on an affine transformation of a default pattern $\tilde{y}_i^k = x^k + A^k z_i^k$ where $T^k = \{x^k \in \mathbb{R}^3, A^k \in \mathbb{R}^{3\times3}\}$ is the affine transform at time k and the default pattern, $z_i \in \mathbb{R}^3$, represents the LEDs in a known frame. The pattern origin is the geometric center of the LED position, $\Sigma_{i=1}^7 e_j z_i = 0$ for $j = x, y, z$. Each affine transform includes both a linear translation and rotation term. Our estimation problem is linear in the individual components of this affine transform, which is why we use it. However, this linearity comes at a disadvantage: an affine transform can represent both physically realistic rotation and non-physical skewing and scaling of the pattern. Since the physical reality will always bias the estimation problem towards valid rigid body transforms, we can safely assume that these transforms can be converted to a physical one later. We find the

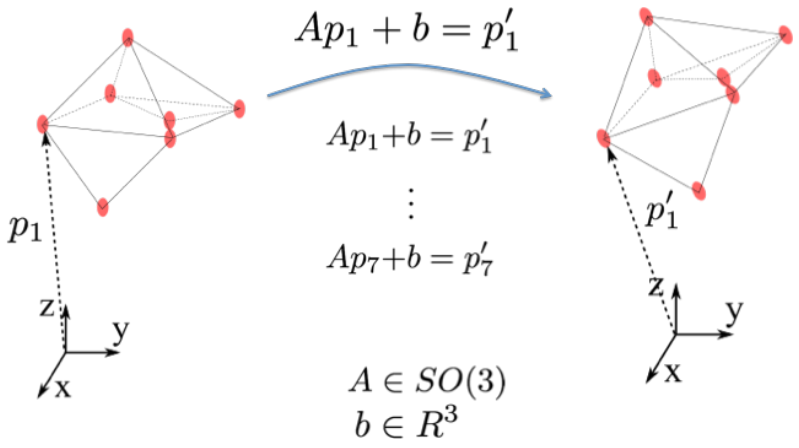


Figure E.1: **Orientation Estimation Using MoCap.** The positions of attached LED sensors are used to estimate A (orientation) and b (position)

best fit affine transform as follows:

$$\theta^k = \begin{pmatrix} x^k \\ \text{vec}(A^k) \end{pmatrix}, \qquad \tilde{\mathbf{y}}^k = \begin{pmatrix} \tilde{y}_1^k \\ \vdots \\ \tilde{y}_7^k \end{pmatrix} = R\theta^k, \tag{E.1}$$

$$R = \begin{pmatrix} I_{3\times3} & e_x z_1^\intercal & e_y z_1^\intercal & e_z z_1^\intercal \\ I_{3\times3} & e_x z_2^\intercal & e_y z_2^\intercal & e_z z_2^\intercal \\ \vdots & \vdots & \vdots & \vdots \\ I_{3\times3} & e_x z_7^\intercal & e_y z_7^\intercal & e_z z_7^\intercal \end{pmatrix}, \tag{E.2}$$

$$K_o \in \mathbb{R}^{n\times7} = \begin{pmatrix} e_0^\intercal \text{ if LED 1 was found} \\ e_1^\intercal \text{ if LED 2 was found} \\ \vdots \\ e_6^\intercal \text{ if LED 7 was found} \end{pmatrix}, \tag{E.3}$$

$$W = \begin{pmatrix} I_{3n\times3n} & 0 \\ 0 & \lambda I_{9\times9} \end{pmatrix}, \tag{E.4}$$

$$R_r \triangleq \begin{pmatrix} (K_o \otimes I_{3\times3})R \\ 0 \quad I_{9\times9} \end{pmatrix}, \tag{E.5}$$

$$\theta^k \triangleq (R_r^\intercal W R_r)^{-1} R_r^\intercal W \begin{pmatrix} (K_o \otimes I_{3\times3})\hat{\mathbf{y}}^k \\ \tilde{\theta}(k|k-p) \end{pmatrix}. \tag{E.6}$$

Here (E.1) describes affine transforms in vector form, and demonstrates the linearity of prediction. The base regressor (E.2) incorporates the default pattern data, but assumes all the LEDs are visible. We use a knockout vector (E.3) of variable size to define which LEDs are to be used in each update. We also employ a regularization term, and a weighting matrix (E.4) parameterized by a variable λ that controls the tradeoff between new rotational information and old data. This results in the full, regularized regressor (E.5), and the best affine transform estimate (E.6).

Note that the regularization to $\tilde{\theta}(k|k-p)$ in (E.6) accounts for dynamics using the IMU data history, starting p steps before k at $k-p$—the index of the last LED position sensor update,

$$\tilde{\theta}(k|k-p) = \hat{\theta}^{k-p} + \sum_{t=k-p}^{k} \begin{pmatrix} 0_3 \\ \mathrm{vec}(\hat{\omega}^t_{\mathrm{IMU}}\times) \end{pmatrix} \Delta t. \tag{E.7}$$

Finally, to attain a valid rigid body transform $\hat{\theta}^k$ from the general affine θ^k, we convert the direction cosine matrix A^k into the closest fit quaternion using the method of [4], and then use that quaternion to generate a new direction cosine matrix $\hat{A}^k$ which is guaranteed to be a rigid body rotation. The original x^k and this new $\hat{A}^k$ form the rigid body transform $\hat{\theta}^k$. The algorithm returns $\tilde{\theta}(t,k)$ as the best estimate of the orientation until a new motion capture LED position message is received.

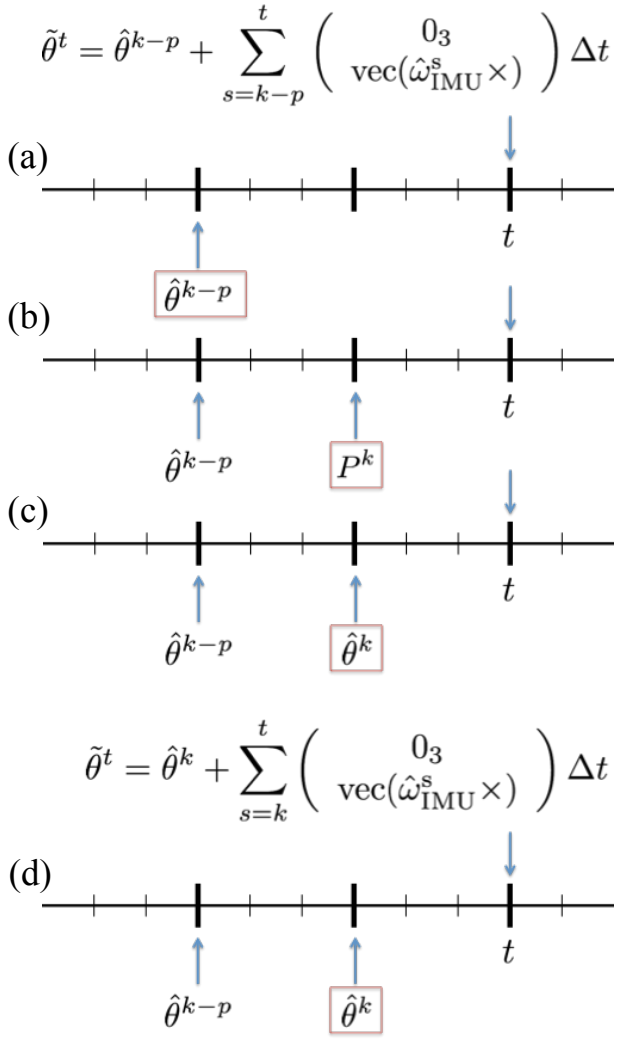


Figure E.2: **Time delay handling.** Since MoCap data are less frequently and slowly updated, (a) the estimator sums the previous estimated value and the acculmated angular velocity of IMU during short period. (b) When there are new available MoCap data, (c) the estimator updates the estimation and shortening the accumulation range.

Bibliography

[1] Kouki Abe, Takahiro Suga, and Yasutaka Fujimoto. Control of a biped robot driven by elastomer-based series elastic actuator. In *2012 12th IEEE International Workshop on Advanced Motion Control (AMC)*, pages 1–6. IEEE, 2012.

[2] Junhyeok Ahn, Orion Campbel, Donghyun Kim, and Luis Sentis. Fast Kinodynamic Bipedal Locomotion Planning with Moving Obstacles. submitted in ICRA, IEEE, 2018.

[3] Jessica Austin, Alexander Schepelmann, and Hartmut Geyer. Control and evaluation of series elastic actuators with nonlinear rubber springs. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6563–6568. IEEE, 2015.

[4] Itzhack Y Bar-Itzhack. New method for extracting the quaternion from a rotation matrix. *Journal of guidance, control, and dynamics*, 23(6):1085–1087, 2000.

[5] Sylvain Bertrand and Jerry Pratt. Momentum-based control framework: application to the humanoid robots atlas and valkyrie. In *IEEE Internationa Conference on Intelligent Robots and Systems, Workshop on Whole-Body Control for Robots in the Real World*, 2014.

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

[7] Michael Blosch, Marco Hutter, Mark A Hoepflinger, Stefan Leutenegger, Christian Gehring, C David Remy, and Roland Siegwart. State Estimation for Legged Robots - Consistent Fusion of Leg Kinematics and IMU. *Robotics Science and Systems 2012*, 2012.

[8] Thiago Boaventura, Gustavo A Medrano-Cerda, Claudio Semini, Jonas Buchli, and Darwin G Caldwell. Stability and performance of the compliance controller of the quadruped robot hyq. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1458–1464, 2013.

[9] Brian G Buss, Amin Ramezani, Kaveh Akbari Hamed, Brent Griffin, Kevin S Galloway, Jessy W Grizzle, et al. Preliminary walking experiments with underactuated 3d bipedal

robot marlo. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2529–2536, 2014.

[10] Enăchescu Călin. Multidimensional function approximation using neural networks.

[11] Stephane Caron, Quang Cuong Pham, and Yoshihiko Nakamura. Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics. In *Robotics: Science and Systems*. Robotics: Science and Systems Foundation, July 2015.

[12] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. Stability of Surface Contacts for Humanoid Robots: Closed-Form Formulae of the Contact Wrench Cone for Rectangular Support Areas. *arXiv.org*, January 2015.

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

[14] Christine Chevallereau, ABBA Gabriel, Yannick Aoustin, Franck Plestan, Eric Westervelt, Carlos Canudas De Wit, Jessy Grizzle, et al. Rabbit: A testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, 2003.

[15] Steven H Collins, Martijn Wisse, and Andy Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001.

[16] Johannes Englsberger, C Ott, M A Roa, Alin Albu-Schaffer, and G Hirzinger. Bipedal walking control based on Capture Point dynamics. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, pages 4420–4427. IEEE, September 2011.

[17] Johannes Englsberger, Christian Ott, and Alin Albu-Schaffer. Three-dimensional bipedal walking control based on divergent component of motion. *IEEE Transactions on Robotics*, 31(2):355–368, 2015.

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

[19] Roy Featherstone. *Rigid body dynamics algorithms*, volume 49. Springer Berlin, 2008.

[20] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G Atkeson. Optimization-based full body control for the darpa robotics challenge. *Journal of Field Robotics*, 32(2):293–312, 2015.

[21] Fabrizio Flacco, Alessandro De Luca, and Oussama Khatib. Motion control of redundant robots under joint constraints: Saturation in the null space. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 285–292. IEEE, 2012.

[22] J Grizzle, A Ramezani, B Buss, B Griffin, K Akbari Hamed, and KS Galloway. Progress on controlling marlo, an atrias-series 3d underactuated bipedal robot. *Dynamic Walking*, 2013.

[23] Jessy W Grizzle, Christine Chevallereau, Aaron D Ames, and Ryan W Sinnet. 3d bipedal robotic walking: models, feedback control, and open problems. In *IFAC Symposium on Nonlinear Control Systems*, volume 2, page 8, 2010.

[24] B. Henze, C. Ott, and M.A. Roa. Posture and balance control for humanoid robots in multi-contact scenarios based on model predictive control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3253–3258, Sept 2014.

[25] Ayonga Hereid, Shishir Kolathaya, Mikhail S Jones, Johnathan Van Why, Jonathan W Hurst, and Aaron D Ames. Dynamic multi-domain bipedal walking with atrias through slip based human-inspired control. In *ACM International Conference on Hybrid Systems: Computation and Control*, pages 263–272, 2014.

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

[27] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti. Momentum Control with Hierarchical Inverse Dynamics on a Torque-Controlled Humanoid. *ArXiv e-prints*, October 2014.

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

[29] Jessica K Hodgins and Marc H Raibert. Biped gymnastics. *The International Journal of Robotics Research*, 9(2):115–128, 1990.

[30] Neville Hogan. Impedance control: An approach to manipulation. In *IEEE American Control Conference*, pages 304–313, 1984.

[31] Marco Hutter, C David Remy, Mark A Hoepflinger, and Roland Siegwart. High compliant series elastic actuation for the robotic leg scarleth. In *Proc. of the International Conference on Climbing and Walking Robots (CLAWAR)*, number EPFL-CONF-175826, 2011.

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

[33] S Hyon, Joshua G Hale, and Gordon Cheng. Full-body compliant human–humanoid interaction: balancing in the presence of unknown external forces. *IEEE Transactions on Robotics*, 23(5):884–898, 2007.

[34] Karl Iagnemma and Jim Overholt. *Special Issue: DARPA Robotics Challenge (DRC)*, volume 32. Journal of Field Robotics, 2015.

[35] Fumiya Iida and Russ Tedrake. Minimalistic control of a compass gait robot in rough terrain. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1985–1990. IEEE, 2009.

[36] Matthew Johnson, Brandon Shrewsbury, Sylvain Bertrand, Tingfan Wu, Daniel Duran, Marshall Floyd, Peter Abeles, Douglas Stephen, Nathan Mertins, Alex Lesman, et al. Team ihmc's lessons learned from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):192–208, 2015.

[37] S Kajita, F Kanehiro, and K Kaneko. Biped walking pattern generation by using preview control of zero-moment point. *International Conference on Robotics and Automation (ICRA)*, 2:1620–1626 vol.2, 2003.

[38] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 239–246. IEEE, 2001.

[39] Oussama Kanoun, Florent Lamiraux, and Pierre-Brice Wieber. Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. *IEEE Transactions on Robotics*, 27(4):785–792, 2011.

[40] Navvab Kashiri, Gustavo A Medrano-Cerda, Nikos G Tsagarakis, Matteo Laffranchi, and Darwin Caldwell. Damping control of variable damping compliant actuators. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 850–856. IEEE, 2015.

[41] R Katoh and M Mori. Control method of biped locomotion giving asymptotic stability of trajectory. *Automatica*, 20(4):405–414, 1984.

[42] Majid Khadiv, Alexander Herzog, S Ali A Moosavian, and Ludovic Righetti. A robust walking controller based on online step location and duration optimization for bipedal locomotion. *arXiv.org*, April 2017.

[43] Donghyun Kim, Junhyeok Ahn, Orion Campbell, Nicholas Paine, and Luis Sentis. Investigations of a Robotic Testbed with Viscoelastic Liquid Cooled Actuators. *arXiv.org*, November 2017.

[44] Donghyun Kim, Junhyeok Ahn, Jaemin Lee, Orion Campbel, and Luis Sentis. Whole-Body Control Incorporating Slip, Inequality Constraints, and Task Hierarchy. submitted in ICRA, IEEE, 2018.

[45] Donghyun Kim, Orion Campbell, Junhyeok Ahn, Luis Sentis, and Nicholas Paine. Investigations of Viscoelastic Liquid Cooled Actuators Applied for Dynamic Motion Control of Legged Systems. In *International Conference on Humanoid Robots (Humanoid)*, 2017.

[46] Donghyun Kim, Steven Jens Jorgensen, Peter Stone, and Luis Sentis. Dynamic behaviors on the NAO robot with closed-loop whole body operational space control. In *16th International Conference on Humanoid Robots (Humanoids)*, pages 1121–1128. IEEE, 2016.

[47] Donghyun Kim, Jaemin Lee, and Luis Sentis. Robust Dynamic Locomotion via Reinforcement Learning and Novel Whole Body Controller. *arXiv.org*, 2017.

[48] Donghyun Kim, Gray Thomas, and Luis Sentis. Continuous cyclic stepping on 3d point-foot biped robots via constant time to velocity reversal. In *IEEE International Conference on Control Automation Robotics & Vision*, 2014.

[49] Donghyun Kim, Ye Zhao, Gray Thomas, Benito R Fernandez, and Luis Sentis. Stabilizing Series-Elastic Point-Foot Bipeds Using Whole-Body Operational Space Control. *IEEE Transactions on Robotics*, 32(6):1362–1379, 2016.

[50] Donghyun Kim, Ye Zhao, Gray Thomas, and Luis Sentis. Empirical Modifications to a Phase Space Planner Which Compensates for Low Stiffness Actuation in a Planar, Point-Foot, Biped Robot. In *the ASME 2014 Dynamic Systems and Control Conference*, page V001T11A001. ASME, 2014.

[51] Donghyun Kim, Ye Zhao, Gray Thomas, and Luis Sentis. Assessing Whole-Body Operational Space Control in a Point-Foot Series Elastic Biped: Balance on Split Terrain and Undirected Walking. *arXiv.org*, page 2855, January 2015.

[52] Junggon Kim. Lie group formulation of articulated rigid body dynamics.

[53] Shishir Kolathaya and Aaron D Ames. Achieving bipedal locomotion on rough terrain through human-inspired control. *Safety, Security, and Rescue Robotics, IEEE International Symposium on*, 2012.

[54] Twan Koolen, Tomas De Boer, John Rebula, Ambarish Goswami, and Jerry Pratt. Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models. *The International Journal of Robotics Research*, 31(9):1094–1113, August 2012.

[55] Twan Koolen, Jesper Smith, Gray Thomas, Sylvain Bertrand, John Carff, Nathan Mertins, Douglas Stephen, Peter Abeles, Johannes Englsberger, Stephen Mccrory, et al. Summary of team ihmc's virtual robotics challenge entry. In *13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 307–314. IEEE, 2013.

[56] Jaemin Lee, Nicolas Mansard, and Jaeheung Park. Intermediate desired value approach for task transition of robots in kinematic control. *IEEE Transactions on Robotics*, 28(6):1260–1277, 2012.

[57] Nicolas Mansard, Oussama Khatib, and Abderrahmane Kheddar. A unified approach to integrate unilateral constraints in the stack of tasks. *Transactions on Robotics*, 25(3):670–685, 2009.

[58] Marcell Missura and Sven Behnke. Online learning of foot placement for balanced bipedal walking. In *14th International Conference on Humanoid Robots (Humanoids)*, pages 322–328. IEEE, 2014.

[59] Michael Mistry, Jun Nakanishi, and Stefan Schaal. Task space control with prioritization for balance and locomotion. In *International Conference on Intelligent Robots (IROS)*, pages 331–338. IEEE, 2007.

[60] Signe Moe, Andrew R Teel, Gianluca Antonelli, and Kristin Y Pettersen. Stability analysis for set-based control within the singularity-robust multiple task-priority inverse kinematics framework. In *54th Annual Conference on Decision and Control (CDC)*, pages 171–178. IEEE, 2015.

[61] Jun Morimoto and Christopher Atkeson. Learning Biped Locomotion. *IEEE Robotics & Automation Magazine*, 14(2):41–51, 2007.

[62] Mohamad Mosadeghzad, Gustavo A Medrano-Cerda, Jody A Saglia, Nikos G Tsagarakis, and Darwin G Caldwell. Comparison of various active impedance control approaches, modeling, implementation, passivity, stability and trade-offs. In *Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on*, pages 342–348. IEEE, 2012.

[63] Ryuma Niiyama, Satoshi Nishikawa, and Yasuo Kuniyoshi. Biomechanical approach to open-loop bipedal running with a musculoskeletal athlete robot. *Advanced Robotics*, 26(3-4):383–398, 2012.

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

[65] C Ott, A Albu-Schaffer, A Kugi, and G Hirzinger. On the Passivity-Based Impedance Control of Flexible Joint Robots. *Robotics, IEEE Transactions on*, 24(2):416–429, 2008.

[66] Nicholas Paine, Joshua S Mehling, James Holley, Nicolaus A Radford, Gwendolyn Johnson, Chien-Liang Fok, and Luis Sentis. Actuator Control for the NASA-JSC Valkyrie Humanoid Robot: A Decoupled Dynamics Approach for Torque Control of Series Elastic Robots. *Journal of Field Robotics*, 32(3):378–396, January 2015.

[67] Nicholas Paine, Sehoon Oh, and Luis Sentis. Design and control considerations for high-performance series elastic actuators. *IEEE/ASME Transactions on Mechatronics*, 19(3):1080–1091, 2014.

[68] Nicholas Paine and Luis Sentis. Design and Comparative Analysis of a Retrofitted Liquid Cooling System for High-Power Actuators. *Actuators*, 4(3):182–202, 2015.

[69] Nicholas Arden Paine. *High-performance Series Elastic Actuation*. PhD thesis, Austin, 2014.

[70] Yongsu Park, Sehoon Oh, and Heeseung Zoe. Dynamic analysis of Reaction Force sensing Series Elastic Actuator as Unlumped two mass system. In *IECON - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 5784–5789. IEEE, 2016.

[71] Gill A Pratt and Matthew M Williamson. Series elastic actuators. In *'Human Robot Interaction and Cooperative Robots', Proceedings. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 399–406. IEEE, 1995.

[72] Gill A Pratt, Matthew M Williamson, Peter Dillworth, Jerry Pratt, and Anne Wright. Stiffness isn't everything. In *Experimental Robotics IV*, pages 253–262. Springer, 1997.

[73] J E Pratt and R Tedrake. Velocity-Based Stability Margins for Fast Bipedal Walking. In *Fast Motions in Biomechanics and Robotics*, pages 299–324. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[74] Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001.

[75] Jerry Pratt, Twan Koolen, Tomas De Boer, John Rebula, Sebastien Cotton, John Carff, Matthew Johnson, and Peter Neuhaus. Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower body humanoid. *The International Journal of Robotics Research*, 31(10):1117–1133, 2012.

[76] E Pucci and G Saccomandi. A note on the Gent model for rubber-like materials. *Rubber chemistry and technology*, 75(5):839–852, 2002.

[77] Nicolaus A Radford, Philip Strawser, Kimberly Hambuchen, Joshua S Mehling, William K Verdeyen, A Stuart Donnan, James Holley, Jairo Sanchez, and et al. Valkyrie: NASA's First Bipedal Humanoid Robot. *Journal of Field Robotics*, 32(3):397–419, May 2015.

[78] M.H. Raibert. *Legged Robots that Balance*, volume 3. MIT Press, Cambridge, Ma., 1986.

[79] Alireza Ramezani, Jonathan W Hurst, Kaveh Akbari Hamed, and Jessy W Grizzle. Performance Analysis and Feedback Control of ATRIAS, A Three-Dimensional Bipedal Robot. *Journal of Dynamic Systems, Measurement, and Control*, 136(2):021012, March 2014.

[80] Oscar E Ramos, Nicolas Mansard, and Philippe Soueres. Whole-body motion integrating the capture point in the operational space inverse dynamics control. In *14th International Conference on Humanoid Robots (Humanoids)*, pages 707–712. IEEE, 2014.

[81] S Rezazadeh and J W Hurst. Toward step-by-step synthesis of stable gaits for under-actuated compliant legged robots. *IEEE-RAS International Conference on Robotics and Automation*, pages 4532–4538, 2015.

[82] Siavash Rezazadeh, Christian Hubicki, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Andy Abate, and Jonathan Hurst. Spring-Mass Walking With ATRIAS in 3D: Robust Gait Control Spanning Zero to 4.3 KPH on a Heavily Underactuated Bipedal Robot. In *Dynamic Systems and Control Conference*. ASME, October 2015.

[83] Nicholas Rotella, Michael Bloesch, Ludovic Righetti, and Stefan Schaal. State estimation for a humanoid robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 952–958. IEEE, 2014.

[84] L. Saab, O. Ramos, N. Mansard, P. Souères, and J-Y. Fourquet. Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Transaction on Robotics*, 29(2):346–362, April 2013.

[85] Luis Sentis. *Synthesis and control of whole-body behaviors in humanoid systems*. PhD thesis, Stanford, 2007.

[86] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(04):505–518, 2005.

[87] Luis Sentis, Jaeheung Park, and Oussama Khatib. Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *IEEE Transactions on robotics*, 26(3):483–501, 2010.

[88] Luis Sentis, Jaeheung Park, and Oussama Khatib. Compliant Control of Multicontact and Center-of-Mass Behaviors in Humanoid Robots. *IEEE Transactions on Robotics*, 26(3):483–501, 2010.

[89] Luis Sentis, Josh Petersen, and Roland Philippsen. Implementation and stability analysis of prioritized whole-body compliant controllers on a wheeled humanoid robot in uneven terrains. *Autonomous Robots*, 35(4):301–319, 2013.

[90] Luis Sentis and Mike Slovich. Motion planning of extreme locomotion maneuvers using multi-contact dynamics and numerical integration. In *11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 760–767. IEEE, 2011.

[91] Bruno Siciliano and J-JE Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pages 1211–1216. IEEE, 1991.

[92] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[93] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and Jessy W Grizzle. A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel. *The International Journal of Robotics Research*, 30(9):1170–1193, 2011.

[94] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and JW Grizzle. Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on mabel. *The International Journal of Robotics Research*, 32(3):324–345, 2013.

[95] Benjamin Stephens. *Push recovery control for force-controlled humanoid robots.* PhD thesis, Carnegie Mellon University, 2011.

[96] Benjamin J Stephens and Christopher G Atkeson. Dynamic balance force control for compliant humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1248–1255, 2010.

[97] T Sugihara. Standing stabilizability and stepping maneuver in planar bipedalism based on the best COM-ZMP regulator. In *Robotics and Automation (ICRA)*, pages 1966–1971, Kobe, 2009. IEEE.

[98] Tomomichi Sugihara, Yoshihiko Nakamura, and Hirochika Inoue. Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1404–1409. IEEE, 2002.

[99] N Sugimoto and J Morimoto. Phase-dependent trajectory optimization for CPG-based biped walking using path integral reinforcement learning. In *13th International Conference on Humanoid Robots (Humanoids)*, pages 255–260. IEEE, 2011.

[100] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, 2011.

[101] Toru Takenaka, Takashi Matsumoto, and Takahide Yoshiike. Real time motion generation and control for biped robot-1st report: Walking gait pattern generation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1084–1091. IEEE, 2009.

[102] R Tedrake, T W Zhang, and H S Seung. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2849–2854 vol.3. IEEE, 2004.

[103] Heike Vallery, Ralf Ekkelenkamp, Herman Van Der Kooij, and Martin Buss. Passive and accurate torque control of series elastic actuators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3534–3538, 2007.

[104] M. Vukobratović and B. Borovac. Zero-moment point: Thirty five years of its life. *International Journal of Humanoid Robots*, 1(1):157–173, mar 2004.

[105] ER Westervelt, Gabriel Buche, and JW Grizzle. Experimental validation of a framework for the design of controllers that induce stable walking in planar bipeds. *The International Journal of Robotics Research*, 23(6):559–582, 2004.

[106] E.R. Westervelt, J.W. Grizzle, C. Chevallereau, J.H. Choi, and B. Morris. *Feebdack control of dynamic bipedal robot locomotion*. CRC Press, 2007.

[107] Eric C Whitman and Christopher G Atkeson. Control of a walking biped using a combination of simple policies. In *9th International Conference on Humanoid Robots (Humanoids)*, pages 520–527. IEEE, 2009.

[108] T Yang, ER Westervelt, A Serrani, and James P Schmiedeler. A framework for the control of stable aperiodic walking in underactuated planar bipeds. *Autonomous Robots*, 27(3):277–290, 2009.

[109] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. In *ACM Transactions on Graphics*, volume 26, page 105, 2007.

[110] Hui-Hua Zhao, Wen-Loong Ma, M.B. Zeagler, and A.D. Ames. Human-inspired multi-contact locomotion with amber2. In *ACM/IEEE International Conference on Cyber-Physical Systems*, April 2014.

[111] Ye Zhao, Nicholas Paine, Kwansuk Kim, and Luis Sentis. Stability and Performance Limits of Latency-Prone Distributed Feedback Controllers. *IEEE Transactions on Industrial Electronics*, 62(11):7151–716, November 2015.

[112] Ye Zhao, Nicholas Paine, and Luis Sentis. Feedback parameter selection for impedance control of series elastic actuators. *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 999–1006, 2014.

[113] Ye Zhao and Luis Sentis. A three dimensional foot placement planner for locomotion in very rough terrains. In *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 726–733. IEEE, 2012.

Vita

Donghyun Kim came to the Human-Centered Robotics Lab in 2012 with a strong interest to experimentally work with highly dynamic legged robots. He started investigating a new technique for dynamic legged locomotion called Phase Space Planning (PSP) which gives analytical information on dynamic stepping transitions. He also started working heavily with a new biped we had built with Meka called Hume. By 2014 he had built and publish a planarizing setup for Hume and made modifications to PSP called Time to Velocity Reversal achieving planar dynamic walking. By 2015 he published an empirical whole body operational space architecture to control body orientation and feet motion on Hume. This was the first WBC on a point foot robot. WBC's are controllers with rely on recursive dynamics and accept a stack of tasks for locomotion and manipulation. This work was subsequently publish in IEEE Transaction on Robotics in 2016. During these times the limitations of Hume came to light. Poor IMU, strong bending of legs and brittle mechanics, poor electronics from Meka. He went on to developed advancements in controls called WBDC, combining QP with priorities for speed. He published research using the Nao robot. During the same year we found Apptronik which built a new embedded nervous system and liquid cooled actuators. We started building a leg testbed for jumping using liquid cooled viscoelastic actuators. Donghyun worked heavily on this leg co-leading a team of students. They worked on WBDC with proprioception and impedance control. This studies have been recently submitted for journal publication. In the meantime we attracted grants to rebuild Hume. Donghyun co-lead a team to do so, redesigning the legs, changing all electronics and using a tactical IMU. He just finished building the new machine this week and he will show initial experiments.

E-mail: dk6587@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.