Terrain-Aware Bipedal Locomotion

by

Grant Gibson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in the University of Michigan
2023

Grant Gibson

grantgib@umich.edu

ORCID iD: 0000-0001-8383-4390

In Loving Memory of my brother,
Brian

Acknowledgments

I want to take a moment to express my deepest gratitude to my advisor, Professor Jessy Grizzle, for providing me with an incredible opportunity to work in the Biped Robotics Lab. Your guidance, pursuit of excellence, patience, encouragement, and belief in me have been invaluable, and I wouldn't have become the researcher I am today without your wisdom and teaching. Being a part of this lab has instilled immense confidence in my ability to impact society positively. I thank my other committee members, Maani Ghaffari, Chad Jenkins, and Ram Vasudevan, for their support and collaboration throughout my time at Michigan.

I am extremely grateful to members of the Biped Robotics Lab: Ross, Omar, Ayonga, Zhenyu, Yukai, Lu, Ray, Bruce, Eva, and Wami, whose relentless drive and enthusiasm made my time at Michigan both productive and enjoyable. A distinct note of gratitude is reserved for Eva, whose infectious personality and meticulous attention to detail have been a continuous source of inspiration. Further, Yukai's generous assistance, especially with Cassie and Digit hardware and software, was pivotal to the achievements of our research. I want to thank Xu, Jennifer, Aayushi, and Gokul for their outstanding work during their Master's degree. Mentoring and collaborating with such a talented group of individuals was a great part of being in the Biped Robotics Lab.

Additionally, I would like to thank Lab4Progress members Alphonsus, Arden, Liz, and Anthony for collaborating on manipulation, locomotion, and computer vision. Specifically to Alphonsus for his efforts on our Kinodynamic Fabrics work on Digit. Your unwavering commitment and passion for robotics have been truly inspiring, pushing me to work even harder.

It has been a privilege to meet amazing friends like Challen, Cynthia, Jon, Justin, and Yves from Michigan and keep old friendships with David, Edgar, Kevin, Milton, Peyton, Vanessa, Yaselly, and Zach. I would also like to thank Liz for always being there for me through the ups and downs. Meeting you has easily been one of my greatest accomplishments in life. To my mother, thank you for all the sacrifices you have made for me. My grandparents have been a huge source of support and encouragement, and I am forever grateful for their presence in my life. I love and value my family more than words can express. Lastly, I want to dedicate this acknowledgment to my late brother Brian, who will always hold a special place in my heart.

TABLE OF CONTENTS

List of Figures

FIGURE

List of Appendices

List of Abbreviations

ALIP	Angular Momentum Linear Inverted Pendulum
CoM	Center of Mass
DARE	Discrete-Time Algebraic Ricatti Equation
DoF	Degree-of-Freedom
HZD	Hybrid Zero Dynamics
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LIP	Linear Inverted Pendulum
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
OSC	Operational Space Control
PD	Proportional-Derivative
QP	Quadratic Program

RC	Radio Control
SLIP	Spring-Loaded Inverted Pendulum
SRB	Single Rigid Body Model
VC	Virtual Constraint

Abstract

Bipedal robots provide a path forward for autonomous systems to operate seamlessly in human-designed environments. They can traverse terrains compatible with wheeled robots and provide improved access to shelves and cabinets out of reach for quadrupeds and other mobile robots. Though their design offers many advantages over other platforms, the consequential challenge of stability remains a glaring hindrance in realizing their true potential. To this end, our work investigates the evolution of algorithmic strategies that enable these robots to traverse various terrains and actively engage with their surroundings dynamically.

To mitigate the unreliability of walking in steep or slippery environments, we first design and test a terrain-aware foot placement locomotion controller (ALIP-MPC) on a 20 Degree-of-Freedom (DoF) Cassie robot. ALIP-MPC displays improved results compared to foot placement methods that disregard terrain information. The controller is validated in simulation and hardware and performs better than other state-of-the-art foot placement methods.

Subsequently, we extend the ALIP-MPC method for a 30-DoF Digit where perception data is used to provide real-time terrain information online. The proposed method is a multi-stage receding horizon algorithm that utilizes properties of the Angular Momentum Linear Inverted Pendulum (ALIP) model for fast execution speeds. Initial results of the fully integrated locomotion controller are shown in simulation and hardware.

Lastly, we broaden the concept of terrain-aware control to encompass interaction with the environment in diverse whole-body tasks. We develop Kinodynamic Fabrics for reactive whole-body control on a 30-DoF Digit robot. This method integrates optimization fabrics within a whole-body nullspace control schema to achieve a range of motions, including balancing and walking.

Chapter 1

Introduction

1.1 Motivation

It's widely acknowledged that robots are playing an increasingly important role in our daily lives. Robots can boost productivity and improve our well-being by taking on repetitive or dangerous tasks. With a plethora of robot types available, from fixed base manipulators to wheeled, legged, aerial, aquatic, and humanoid robots, there's virtually no limit to how robots can be utilized to enhance our lives [13, 14, 15, 16]. In this thesis, we will focus on bipedal robots, legged robots that more closely resemble human morphology. We aim to develop more efficient control algorithms for these robots, increasing their usefulness across various applications.

Compared to wheeled platforms, legged robots are more robust to deviations in terrain. Recently, robots such as the Boston Dynamics® Spot and the ANYbotics® ANYmal have shown many exciting improvements that are impressive [17, 18, 19, 20, 21]. The downside is that they are harder to control from a stability and navigation perspective. This challenge becomes increasingly more complex as the number of legs is reduced.

This begs the question, why would you not use as many legs as possible? Robots mimicking the multi-legged designs of insects have been created and controlled to navigate very rough terrain [22]. There are downsides to having multiple legs for navigation. From a mechanical perspective, more legs imply more mechanical

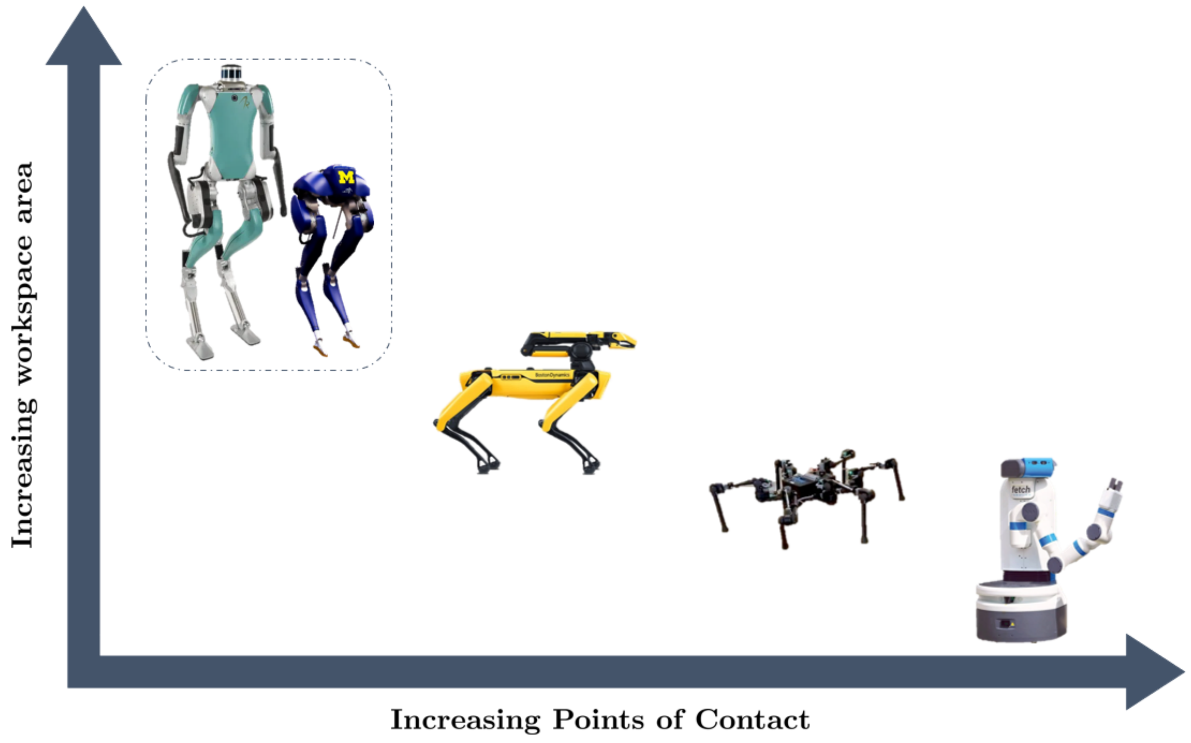


Figure 1.1: Chart comparison for various mobile robot platforms comparing the number of ground end effector contacts to the amount of accessible workspace area.

components that must be designed, fabricated, and maintained. Increased appendages lead to increased weight and volume, which ultimately leads to reduced speed and maneuverability [23]. From a control perspective, increased appendages increase the number of controllable degrees of freedom. Due to the indeterminate solution of the force distribution problem, controlling and designing gaits for highly redundant and over-actuated systems prove challenging [22].

In this thesis, we concentrate on the control of bipedal robots. Their design resembles a human form, offering enhanced workspace manipulation and making them particularly adept at navigating human-centric environments [24]. The height and size of bipedal robots are also more comparable to humans. This design choice inherently makes the control problem more challenging, which can lead to very interesting and exciting advances in control and robotics.

Bipedal robots should be able to navigate our environments and interact with our world. The work presented in this thesis aims to improve these robot capabilities by integrating multiple sensing modalities for improved performance.

1.2 Objectives

The objectives of this dissertation are to:

- Combine reduced-order models with time-varying terrain parameters to develop a robust locomotion controller, termed ALIP-MPC, for bipedal robots,
- Propose and present initial results for a perception-integrated version of the ALIP-MPC for real-time control of bipedal humanoid robots,
- Develop and implement Kinodynamic Fabrics: a motion feedback control framework enabling reactive whole-body control for underactuated humanoid robots and
- Provide open-source code, videos, and results for each method and algorithm presented in this dissertation for simulations and real-world experiments.
  - https://github.com/UMich-BipedLab/cassie_alip_mpc
  - https://github.com/UMich-BipedLab/digit_locomotion_controller
  - https://github.com/UMich-BipedLab/KinodynamicFabrics.jl

1.3 Thesis Structure

Chapter 2 provides relevant information on bipedal locomotion and terrain navigation from software and hardware perspectives. In Chapter 3, we develop and validate a terrain-aware foot placement controller on a Cassie robot, both in simulation and the real world. This work is expanded and integrated with real-time perception data in Chapter 4 to create a locomotion controller for a Digit robot.

Lastly, in Chapter 5, we introduce a novel motion feedback control framework that enables reactive whole-body control for humanoid robots. This framework is also validated on the Digit robot using simulation and hardware. Additionally, we present preliminary results for a Nonlinear Model Predictive Control (NMPC) and Hybrid Zero Dynamics (HZD) stair climbing method for a planar five-link bipedal robot in Appendix A. Further details on what parameters are used in the Digit locomotion controller are provided in Appendix B.

Chapter 2

Background

2.1 Framework for Bipedal Robot Locomotion

All three components of the field of robotics (acting, reasoning, and sensing) are essential for successfully implementing bipedal robot locomotion. These intricately interconnected components form a comprehensive framework for these complex systems (Figure 2.1).

There are two primary sensing modalities: proprioceptive and exteroceptive sensing. Proprioceptive sensors provide information about a robot's internal state and incorporate devices such as joint encoders, gyroscopes, and accelerometers. In contrast, exteroceptive sensing involves gathering data from the robot's external environment and employs devices like Light Detection and Ranging (LiDAR) and RGB-D cameras.

State estimation is a crucial process for robots, enabling them to utilize sensory data to comprehend their position and orientation. This understanding facilitates more accurate and effective interactions with the environment, improving the robot's overall functionality and performance. Recent advancements in state estimation techniques have significantly improved localization across various environments, employing a diverse array of sensing modalities [25, 26, 27]. These advancements contribute to more robust and reliable robot performance and enhance the capabilities of bipedal robots in complex and dynamic environments, such as those encountered in real-world applications.

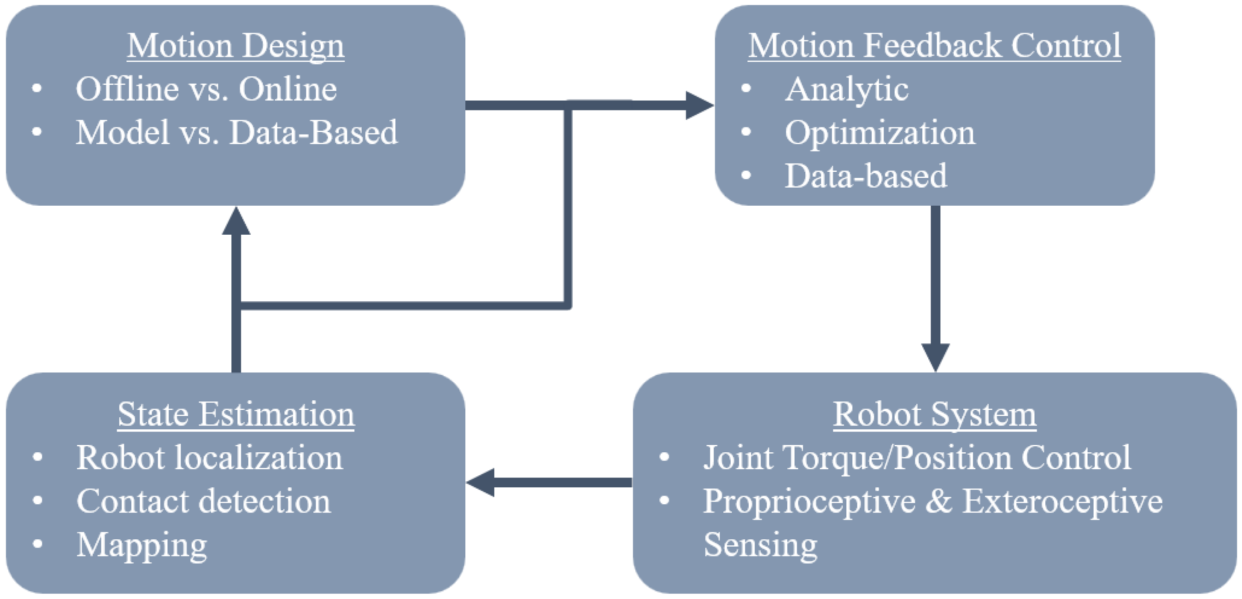


Figure 2.1: Bipedal/Humanoid Robot System Components.

Sensing and state estimation, as emphasized earlier, are crucial components of the bipedal locomotion framework. While these elements are essential for a comprehensive understanding of robot control, this dissertation's primary contribution and focus lie in the framework components of motion design and motion feedback control. Consequently, these aspects, along with the integration of perception for legged locomotion, will be the main subjects of discussion in the remaining sections of this chapter.

2.2 Motion Design

Motion design, also called motion generation or trajectory generation, involves creating trajectories at either the task-space or joint-space level, which a robot must follow to accomplish a specific goal. These trajectories can be custom-designed or computed through optimization techniques, and their dependence on the robot's model may vary. In this section, we discuss the different models used for locomotion and the methods employed in their construction.

2.2.1 Model Complexity

Many different models can be used with varying complexities to design motion trajectories. The kinematics and dynamics of each model type provide advantages and disadvantages that trade-off between accuracy and efficiency.

At one end of the modeling spectrum for bipedal locomotion lies the complete nonlinear hybrid system, which provides the most accurate representation of the robot. The hybrid system consists of various dynamical modes encompassing continuous and discrete dynamics. Guard functions govern transitions between these modes.

In standard bipedal robots, continuous dynamics are represented by the left, right, and double support phases. Discrete dynamics, which often occur instantaneously, are captured by impact dynamics. Guard functions, typically expressed as configuration-related unilateral inequalities, delineate the addition or removal of contact points between the robot and its environment. By incorporating the full spectrum of hybrid dynamics, this comprehensive modeling approach offers a detailed and precise representation of bipedal locomotion, facilitating the design of more effective motion planning and control strategies [28, 4]. The full dynamical model for a bipedal robot can be succinctly represented as

$$\mathcal{H} := \begin{cases} \dot{\mathrm{x}}_i = f(\mathrm{x}_i) + g(\mathrm{x}_i)u & \mathrm{x}_i^- \in \mathcal{S} \\ \mathrm{x}^+ = \Delta(\mathrm{x}^-) & \mathrm{x}_i^- \in \mathcal{S} \end{cases}, \tag{2.1}$$

where i denotes a specific phase of the dynamics (i.e. left stance, double support, etc.), x_i is the state of system in phase i, $f(\mathrm{x}_i)$ is the drift function of the dynamics, $g(\mathrm{x}_i)$ is the control distribution matrix, u is the control input vector, $\mathcal{S}$ is the corresponding guard function for the phase i, Δ is the impact/reset map that either continuously or discontinuously modifies the system state, and $+, -$ represent the states before and after applying the impact/reset map. Figure 2.2 shows an

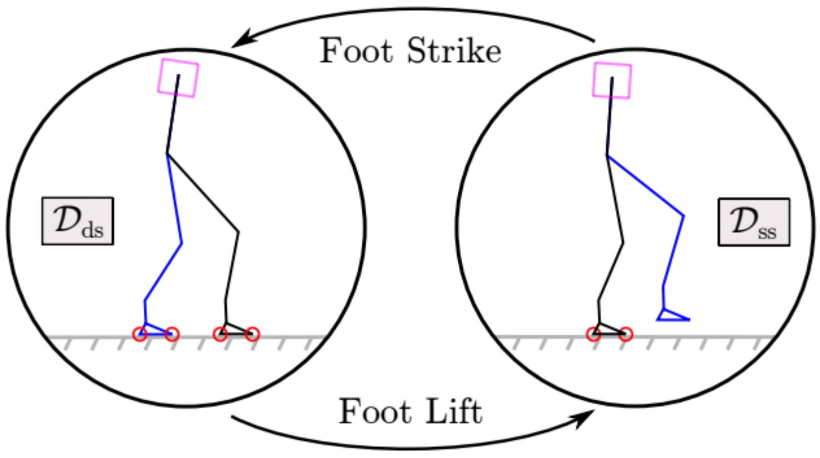


Figure 2.2: Domains graph of 3D walking, where red circles represent foot contact points. The model transitions instantaneously between single and double support phases (assuming the legs are symmetric). Modified from [1].

example of what a hybrid system can look like for a simplified bipedal robot.

The full-order and nonlinear dynamics of bipedal robot models present challenges for control, primarily due to their complexity. Firstly, offline trajectory generation becomes more complex as optimization problems turn nonconvex, inherently making them harder to solve. Furthermore, even with accurate models of all dynamics, potential model inaccuracies still exist. Consequently, many researchers in the field have embraced reduced-order models to represent robot behavior in a lower dimension. These models effectively simplify the control problem while maintaining adequate accuracy.

Examples of successful reduced-order models include the Linear Inverted Pendulum (LIP) [29], the Spring-Loaded Inverted Pendulum (SLIP) [30], the Single Rigid Body Model (SRB) [3], and Virtual Constraint (VC) Models [4, 5], which have been employed for generating dynamic walking patterns. In specific scenarios, impact dynamics can be simplified by making assumptions such as non-slip impacts and inelastic collisions or by approximating the robot's contact with the ground as an instantaneous change in velocity and state. These simplifications help reduce the

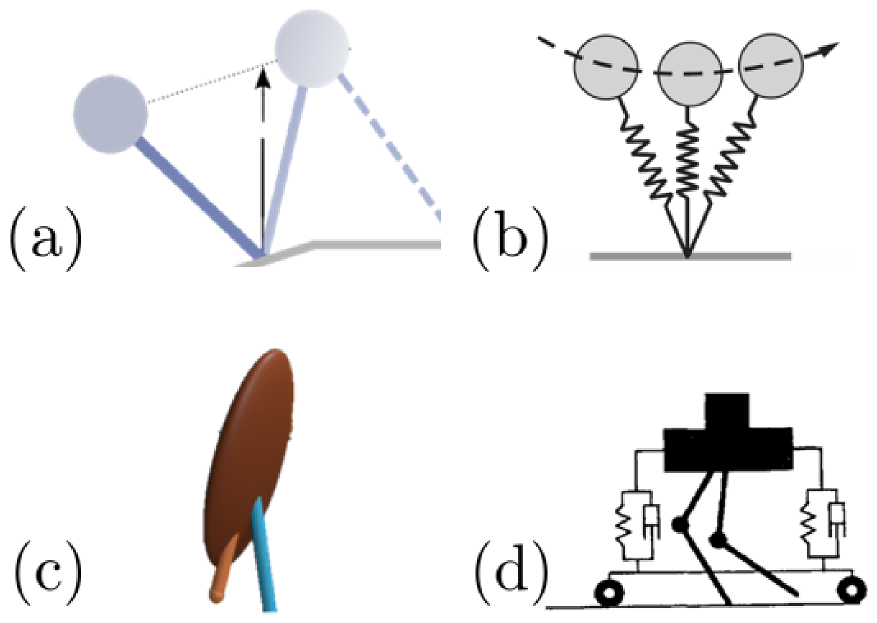


Figure 2.3: Illustrations of some of the most common reduced order models: (a) Linear-Inverted Pendulum, (b) Spring-Loaded Inverted Pendulum [2], (c) Single-Rigid Body [3], (d) Virtual Constraints and Models [4, 5].

complexity of the hybrid system model and facilitate a more manageable control design process. Additional data-based dynamics, termed regressor dynamics, have also been used successfully for various robots [31].

2.2.2 Model-based Design

Model-based motion design can be broadly categorized into manual (hand-designed) and optimization-based. Manual trajectories rely on user expertise to determine the shape and structure of each task or joint path. These hand-designed trajectories are often parameterized by stability-related metrics, such as swing foot location and Center of Mass (CoM) position, as well as constraints related to friction, collisions, and velocity components.

In contrast, optimization-based motion design incorporates stability metrics as cost and constraint functions within optimization problem formulations. Model Predictive Control (MPC) is one widely used optimization method for designing motions. The complexity of the models used in optimization-based design can range from simple single-rigid body dynamics to full 3D nonlinear models. Various adaptations of MPC have been employed to generate trajectories for bipedal and humanoid locomotion, demonstrating the flexibility and efficacy of this approach [32]. HZD is another optimization technique for bipedal gait (or trajectory) design that leverages the full hybrid dynamics of the robot. By computing stabilizing trajectories on the zero dynamics, HZD effectively reduces the state space for control, enabling more efficient motion planning [4, 33].

2.2.3 Data-Driven Design

Biological inspiration can play a role in the development of robot motion design, as it provides insights into the efficient and adaptable movements found in nature. Insect, animal, and human motions have all been utilized as sources of inspiration for creating more effective robotic locomotion [34, 35, 5, 2]. By studying and analyzing these biological systems, researchers can identify essential principles and mechanisms that can be translated into robotic applications.

Data from external vision sensors and attached proprioceptive sensors are employed to reconstruct desired motions that robots can execute, enabling them to mimic the natural movements found in biological systems. This process often involves using machine learning algorithms and optimization techniques to extract and model the key features of the observed motions. By integrating these data-driven motion designs into robotic systems, engineers can create robots with enhanced capabilities, such as improved agility, stability, and adaptability to various environments and tasks. However, a downside of these methods is that they can sometimes lead to overfitting to specific scenarios or tasks, potentially limiting the generalizability and adaptability of the robotic system when encountering novel or unforeseen situations [36, 37].

2.3 Motion Feedback Control

Motion Feedback Control is the calculation of commanded torques sent to the robot to track desired motions. Control can be model-based or data-driven, both with good and bad characteristics.

2.3.1 Model-Based Feedback Control

As described earlier, the various models are often combined with a blend of nonlinear control theory and optimal control techniques to achieve effective bipedal robot control. Analytic nonlinear control approaches include inverse dynamics, partial feedback linearization, and passivity-based control methods. These approaches have been widely used for bipedal control [4, 33, 38, 39]. Furthermore, this dissertation introduces Kinodynamic Fabrics, which leverage geometric mechanics with nonlinear control (presented in Chapter 5).

Operational Space Control (OSC) is another standard method for computing torques that should be applied to the robot to fulfill task requirements and constraints. OSC involves formulating an optimization problem that aims to minimize the tracking error between the desired and actual task-space trajectories while considering system dynamics, joint limits, and other constraints. The optimization process generates the necessary joint torques to achieve the desired task-space motion, ensuring the robot's efficient and smooth operation. By combining different models and control techniques, researchers have been successful in achieving a variety of behaviors in bipedal robots [40, 41, 42, 3, 43, 44, 45]. These diverse strategies showcase the potential of combining model-based and control-based methods for more effective and versatile locomotion solutions in bipedal and humanoid robots.

2.3.2 Data-Driven Feedback Control

Data-driven methods for feedback control vary from highly model-dependent to model-free approaches, each with advantages and disadvantages. Model-free, data-driven methods typically employ reinforcement learning in physics-based simulations to guide the command policy toward high-level goals, such as base velocity tracking without falling over. Physics engines, such as MuJoCo [46] and PyBullet [47], are prominent software tools used to simulate the forward dynamics of robots and collect data for policy optimization. Numerous successful algorithms have been demonstrated in simulation for a variety of legged robots [48, 49, 50].

Model-free methods can have poor performance (i.e., aperiodicity, lack of steady-state convergence, and inconsistent repeatability of tasks) without expert architecture knowledge, making it beneficial to incorporate model-based components for enhanced hardware performance. Reduced-order, single-rigid body, and centroidal dynamics models have been integrated into learning frameworks to improve disturbance rejection and agile movements on Cassie robots [43, 51, 3]. HZD constraints were combined with reinforcement learning to enhance stability convergence of data-driven methods on both Cassie and Digit robots [52, 53].

Despite impressive results achieved by these data-driven methods, they still necessitate precise network architecture and reward policy design, and the controllers are typically tailored to specific tasks. For instance, the stair-climbing controller by Siekmann et al. [43] is not designed or constructed in the same manner as the running controller by Batke et al. [3]. This dissertation concentrates on developing model-based motion design and feedback control algorithms for greater clarity and generality.

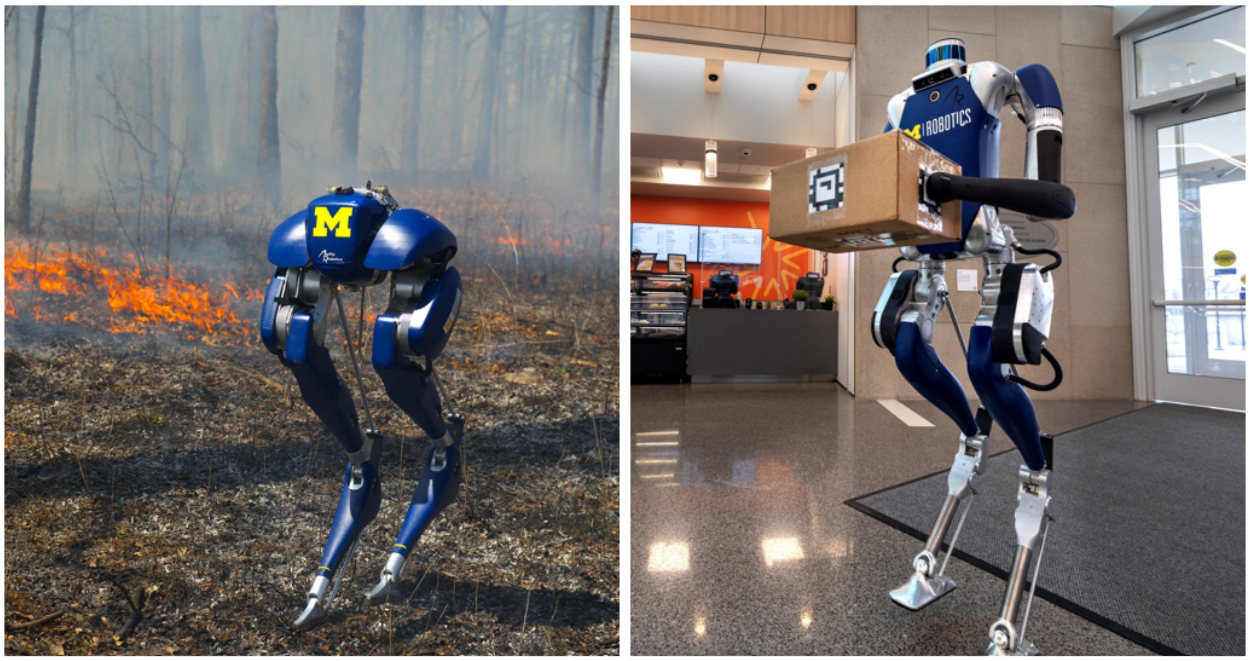


Figure 2.4: Bipedal robots Cassie (left) and Digit (right) made by Agility Robotics [6, 7].

2.4 Robots

The Cassie and Digit robots developed by Agility Robotics are described in this section.

2.4.1 Cassie

Cassie, designed by Agility Robotics, is a 20-Degree-of-Freedom (DoF) bipedal robot capable of dynamically walking and running over rough terrain [6]. The robot's unique morphology features rigid linkages that create closed chains in each leg configuration, constraining the two passive joints highlighted in yellow in Figure 2.4. Fiberglass spring plates embedded in each closed chain provide compliance and enable the robot to absorb impact forces during locomotion. The morphology of Cassie's legs resembles that of a Cassowary, with the hip and knee joints located higher up near the pelvis. As a result, uncommon terminologies such as tarsus and toe represent the final two joints in the kinematic branch of each leg. It is worth noting that Cassie has only one DoF at the end of each leg, limiting its ability to control roll (rotation about the body x-axis) and yaw (rotation about the body z-axis) for each toe.

Cassie's unique design has enabled researchers to investigate various terrain-aware locomotion strategies, including whole-body motion planning, control, and optimization techniques that leverage the robot's passive compliance and dynamic capabilities. Cassie's dynamic abilities have been demonstrated on various terrains, including stairs, ramps, and uneven terrain [40, 41, 42, 3, 43, 44]. These studies have paved the way for developing robust and efficient bipedal locomotion systems that can adapt to real-world environments.

2.4.2 Digit

Digit, designed by Agility Robotics, is a mechanical advancement from the Cassie platform, as it is equipped with additional hardware and sensors that enhance its capabilities (Figure 2.4) [7]. Adding a torso and arms to the existing bipedal structure allows for a greater range of motion and flexibility and increased dexterity for object manipulation. One of Digit's most significant advancements is its incorporation of LiDAR and RGB-D sensors, which provide the robot with real-time, 3D environmental data. This sensor suite enables Digit to perceive its surroundings and adapt its locomotion accordingly, making it an ideal candidate for terrain-aware walking.

Incorporating terrain information into a robot's gait planning is crucial to achieving agile and adaptable bipedal locomotion. Recent research has focused on developing algorithms and controllers that allow robots like Digit to move across various terrains. The exteroceptive sensors on Digit provide valuable terrain information that can be extracted to estimate the location, height, and shape of obstacles and the slope and roughness of the ground. This can then be used to plan more efficient and stable walking trajectories. This terrain information is used to plan more efficient and stable walking trajectories, enabling Digit to adjust its steps and posture in real-time to accommodate unexpected terrain variations.

The research version of Digit (`digit-v3`) has a sensor integration issue that makes the perception data stream unreliable while in the `low-level-api` operation mode. As a solution, additional sensors are mounted and integrated externally into the native API of the robot. For a more complete discussion, refer to Section 4.5.1.

2.5 Perception in Legged Locomotion

Initially, bipedal robot systems were developed using only proprioceptive sensing, as balance and stability were the main challenges to address. The first actuated biped on record, the WL-1, was created at Waseda University in 1966 and relied exclusively on joint encoder and pressure sensors [54, 55]. As hardware and algorithm improvements progressed, the integration of additional sensors, particularly those for perception, became more widespread. Data extracted from these sensors have been utilized for object detection, semantic and terrain mapping, and path planning [56, 57, 58, 59, 60, 61].

Furthermore, the integration of perception sensors in legged locomotion systems has significantly enhanced the robot's interaction with its environment. These cameras, LiDAR, and other vision-based apparatus enable robots to gather valuable information from their surroundings in real-time [61, 62]. This data is subsequently utilized to make informed decisions, such as detecting obstacles, discerning terrain variations, and adapting gait patterns accordingly [52, 63, 56, 57]. The synergy between perception and legged locomotion has ushered in a new era where robots can maintain balance and navigate complex environments with heightened agility and safety. As research in this domain continues to evolve, the fusion of perception and legged locomotion promises to improve bipedal robots' capabilities in many applications.

Chapter 3

Terrain-Aware Foot Placement Controller for Cassie

3.1 Introduction

This chapter contributes to the growing literature on terrain-adaptive locomotion [42]. We design a gait (locomotion) controller that enables an agile bipedal robot, such as Cassie in Fig. 3.1, to traverse terrain as close to a planned velocity as the physical limits of the robot and terrain conditions allow.[1] We assume that the robot is (a) provided a local planar approximation of the terrain, (b) a local friction cone, and (c) a vector field of desired velocity (speed, heading, and yaw rate) as a function of the robot's current pose and velocity. The integral curves of the vector field provide a family of paths that the robot may follow to reach a goal unknown to the local gait controller. These parameters may come from a reactive planner, as in [64, 65], or through a human operator and a Radio Control (RC) transmitter, as is done in this paper.

We make a key simplifying assumption on the terrain: that over robot step-length distances, it can be piecewise approximated by planes, with allowed jumps at the boundaries. This admittedly vague assumption will be made more precise in Sect. 3.2, where we model the center of mass dynamics of the robot. The MPC foot-placement controller plans N robot steps ahead with a terminal cost that assumes the terrain slope and friction cone remain the same beyond the planning

[1]An open-source repository containing code, videos, and results is available at https://github.com/UMich-BipedLab/cassie_alip_mpc.

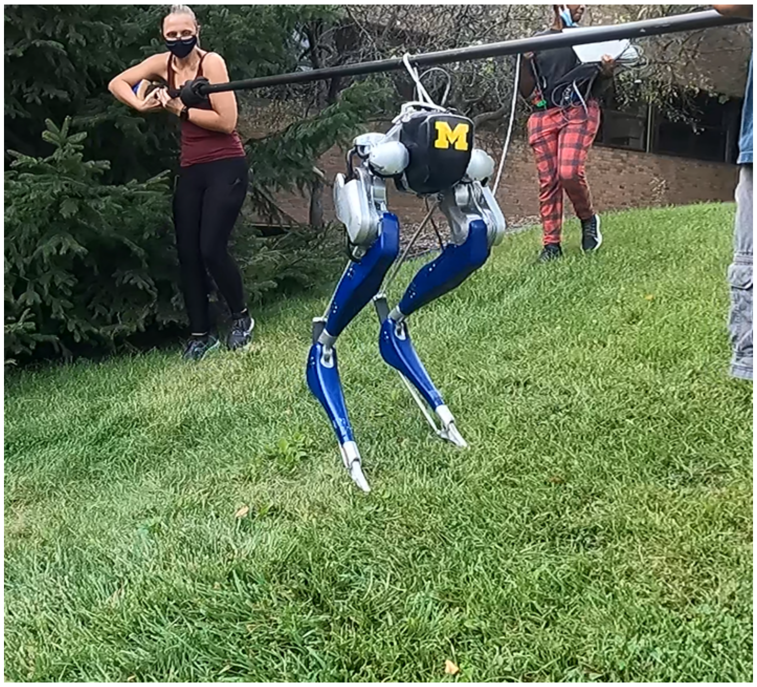


Figure 3.1: Cassie Blue using an ALIP-inspired MPC gait controller to walk sideways up a 22^o incline on wet grass. Lateral walking with Cassie is much more difficult than longitudinal walking due to tight workspace constraints, which are accounted for in our formulation.

horizon. The ability to include workspace constraints (e.g., avoid self-collisions) and an approximate friction cone (e.g., avoid overconfidence on horizontal ground reaction forces) leads to significantly enhanced agility over the original Angular Momentum Linear Inverted Pendulum (ALIP)-based controller introduced in [8]. In addition, including a piecewise linear approximation of the local terrain in the ALIP model enlarges the situations where the approximate zero dynamics analysis and associated stability guarantees in [41] are applicable.

3.1.1 Related Work

Switching Control Based on One-step Ahead Terrain Profile: Terrain-adaptive locomotion of a simulated 3D humanoid is achieved in [66]. First, an offline library that includes five periodic gaits and a set of transition gaits that terminate in a periodic gait is computed. The gaits are parameterized to allow a low-level joint controller to move the robot in a single step from a current pose to a desired final pose, with the desired pose planned in real-time at step initiation as a function of a terrain height map. Similarly, reference [67] first develops a set of feedback controllers for bipedal walking on flat ground, upstairs and downstairs, called motion primitives. Then, a set of feedback controllers is designed that evolve the robot from one motion primitive to another (termed motion transitions). The appropriate controller is selected at step transition.

Terrain Robust: Previous locomotion work has also addressed gait controller design for a specified, finite set of terrain height perturbations [68, 69, 70, 51]. During offline optimization, which could be via parameter optimization or reinforcement learning, a "score" is assigned based on how the closed-loop system (consisting of the controller and robot) responds to a family of terrain profiles. The online controller can only use proprioception (such as Inertial Measurement Unit (IMU) and joint encoder signals) to complete a locomotion task. In particular, the controller is not provided exteroceptive information on terrain profile, as in [66] or [67], for example.

MPC for Foot Placement without Terrain Preview: In [71], the decoupled LIP model dynamics, first introduced in [29], is used to solve a hybrid system-based optimization problem by computing center of pressure trajectories for a specified footfall pattern. These trajectories are computed at the beginning of each domain and are used as inputs to a virtual constraint-based Quadratic Program (QP) to realize joint torque commands. A separate hybrid system MPC approach was also performed in [44] for computing footfalls on a bipedal robot. The footfalls were chosen to minimize errors between the propagated LIP dynamics and a pre-specified reference trajectory.

One Step-Ahead Prediction: This paper designs an MPC controller that blurs the boundary between gait generation and trajectory planning. Our starting point is the one-step ahead gait controller in [8, 41], which bridged the gap between the low-dimensional LIP models in [72, 29, 73, 74, 75, 76] and the method of VCs and HZD in [77, 70, 78]. The main contribution of [8, 41] was to show that when the CoM dynamics of a physical robot are parameterized in terms of the *angular momentum about the contact point instead of linear velocity*, the resulting model is only weakly affected by the angular momentum about the center of mass; in effect, the angular momentum about the contact point acts as a form of "total momentum", accounting for both linear momentum about the contact point and the angular momentum about the center of mass. When a 3D bipedal robot is controlled so that its CoM height is constant, a four-dimensional linear model (referred to as the 3D-ALIP) that is weakly perturbed by angular momentum about the center of mass is extracted. When the perturbation term is dropped, the model simplifies to a pair of decoupled 2D models for the sagittal and frontal planes, respectively; see also the decoupled dynamics of the LIP for comparison [29].

The 2D-ALIP models were subsequently used in [8] to predict angular momenta at the end of the next step as a function of the robot's current angular momenta, position of its center of mass, and the swing-foot position at the end of the current step. When a (dead-beat) foot-placement controller was designed to place the swing foot to match the predicted angular momentum to the desired angular momentum at the end of the next step, the Cassie bipedal robot was able to walk at 2.1 m/s, complete a 90^o turn in 5 steps when walking at 1.0 m/s, and traverse significant slopes [79].

Importantly, *achieving this agile performance on Cassie required a skilled operator for the RC transmitter*, namely, an operator who has an intuitive feel for how rapid changes in commanded speed could result in workspace violations, self-collisions (especially in lateral walking), foot slippages, and who could appropriately adjust foot clearance for locomotion over sloped terrain. This paper transforms the one-step ahead controller in [8] into a multi-step horizon MPC controller. Moreover, the center of mass is allowed to move parallel to the ground, workspace constraints on the legs are included to avoid self-collisions, and finally, the friction cone of the local terrain is incorporated. These contributions prepare the gait controller for integration with perception, mapping, and motion planning components illustrated in [56].

3.1.2 Contributions

We provide the following contributions for enhancing terrain-aware locomotion:

- Provide new insights about the exact CoM dynamics of a bipedal robot using CoM and angular momentum about the contact point as state variables. After applying a constraint to enforce the CoM height to remain a constant distance to the ground, we derive *coupled* dynamics as opposed to the decoupled dynamics generated when using CoM velocity [29]. In [29], the decoupling is exact due to the point contact assumption. However, additional assumptions are needed when using angular momentum. We justify why specific terms can be treated as negligible, which allows us to recover a decoupled linear system about the new state variables.

- Formulate an N-step receding horizon optimization problem that incorporates the 3D-ALIP dynamics and a piecewise linear terrain approximation for computing foot placements. The foot placement solutions, subject to workspace and approximate friction cone constraints, are computed to asymptotically achieve desired periodic trajectories at the end of the planning horizon.

- Create a novel set of virtual constraints and desired trajectories that can be used on a bipedal robot to achieve the desired motion of the 3D-ALIP dynamics for locomotion on piecewise linear terrain. The virtual constraints ensure that the CoM and swing toe remain parallel to the local ground plane,

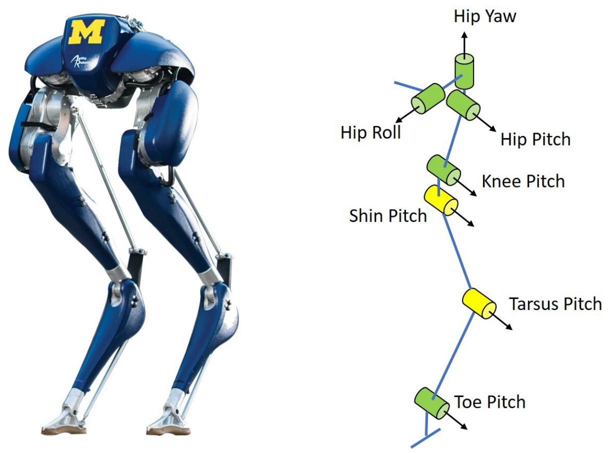


Figure 3.2: The figure presents the Cassie robot's kinematic tree, featuring labeled joints and a photographic representation of Cassie in its real-world form.

extending the work in [8], which assumed a constant ground height.

- Demonstrate the enhanced agility delivered by the control algorithm when implemented on a 20-DoF Cassie robot, in comparison to previous results[8].

3.2 3D Robot Models & Dynamics

3.2.1 Cassie Robot Model

We assume a pinned point contact dynamic model of the form

$$D(q)\ddot{q} + H(q,\dot{q}) = B(q)u, \tag{3.1}$$

with no yaw motion about the stance foot (Cassie has a blade foot). The generalized coordinates $q \in \mathbb{R}^{n_q}$, the vector of motor torques $u \in \mathbb{R}^{n_u}$, and the torque distribution matrix has full column rank. $D(q)$ is the mass inertia matrix, $H(q,\dot{q})$ is the combination of Coriolis, centrifugal, and gravity forces, and B is the torque distribution matrix. For Cassie, $n_q = 15$ due to the blade foot and $n_u = 10$ if the ankle torque on the stance foot is included. This work will set it to zero for simplicity, leaving $n_u = 9$.

The generalized coordinates of the Cassie robot are defined as

$$q = \begin{bmatrix} q_{\text{base}} \\ q_{\text{body}} \end{bmatrix}, \tag{3.2}$$

where

$$q_{\text{base}} = \begin{bmatrix} q_{\text{position,x}} & q_{\text{position,y}} & q_{\text{position,z}} & q_{\text{Euler,yaw}} & q_{\text{Euler,pitch}} & q_{\text{Euler,roll}} \end{bmatrix}^T \tag{3.3}$$

are the floating base coordinates representing the pose of the base link with respect to a fixed inertial frame and

$$q_{\text{body}} = \begin{bmatrix} q_{\text{left,leg}}^T & q_{\text{right,leg}}^T \end{bmatrix}^T. \tag{3.4}$$

are the body coordinates for each arm and leg. The leg coordinates are defined as

$$q_{\text{i,leg}} = \begin{bmatrix} q_{\text{i,HipRoll}} \\ q_{\text{i,HipYaw}} \\ q_{\text{i,HipPitch}} \\ q_{\text{i,Knee}} \\ q_{\text{i,KneeToShin}} \\ q_{\text{i,ShinToTarsus}} \\ q_{\text{i,ToePitch}} \\ q_{\text{i,ToeRoll}} \end{bmatrix} = \begin{bmatrix} q_{\text{iHR}} \\ q_{\text{iHY}} \\ q_{\text{iHP}} \\ q_{\text{iK}} \\ q_{\text{iK2S}} \\ q_{\text{iS2T}} \\ q_{\text{iTP}} \\ q_{\text{iTR}} \end{bmatrix}, \tag{3.5}$$

where $i \in \{\text{left}, \ \text{right}\}$.

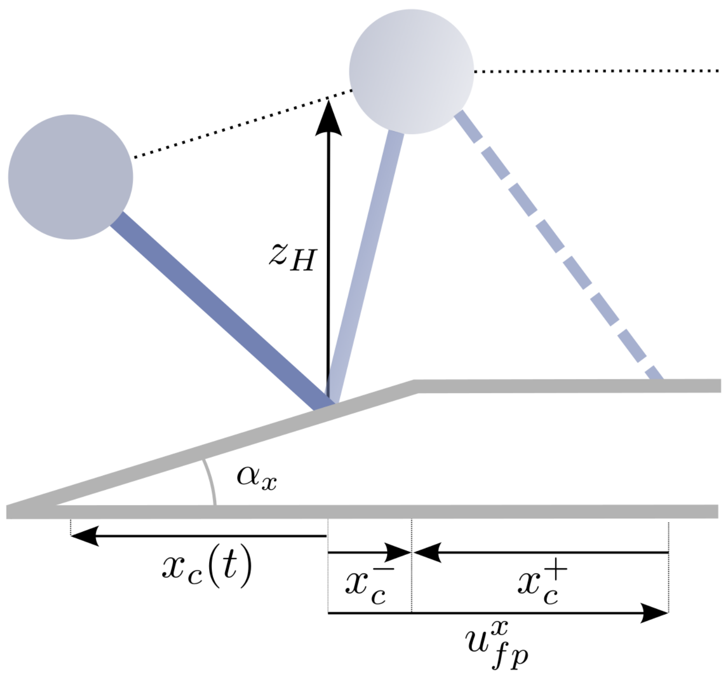


Figure 3.3: The planar linear inverted pendulum is shown. $x_c(t)$ represents the location of the center of mass with respect to the stance foot at time t. The control input u_{fp}^x denotes the foot placement at the end of the current step. The state before and after the instantaneous impact is denoted with a minus (-) and a plus (+) sign, respectively.

3.2.2 Center of Mass Dynamics for Full-Order and Reduced-Order Model

For a 3D robot with a point contact, the dynamics for CoM positions and angular momenta about the contact point can be written as follows,

$$\begin{aligned}
\dot{x}_c &= \frac{L^y}{mz_c} + \frac{\dot{z}_c}{z_c}x_c - \frac{L_c^y}{mz_c} \\
\dot{y}_c &= -\frac{L^x}{mz_c} + \frac{\dot{z}_c}{z_c}y_c + \frac{L_c^x}{mz_c} \\
\dot{L}^x &= -mgy_c \\
\dot{L}^y &= mgx_c.
\end{aligned} \tag{3.6}$$

where m is the mass of the robot, g is the gravitational constant, x_c, y_c, z_c denotes the CoM position with respect to the contact point, $L^{x,y,z}$ denotes the angular momenta about the x, y, z-axes of the contact point, and $L_c^{x,y,z}$ denotes the angular momentum about the CoM. Assuming $p = [x_c,\ \ y_c,\ \ z_c]^T$ and F_g is the gravitational force vector applied to the CoM, the above can be derived from the fact that $L = p \times m\dot{p} + L_c$ and $\dot{L} = p \times F_g$.

Motivated by the CoM height constraint from the LIP model [29], we will later design virtual constraints to impose the following relations on the evolution of the CoM,

$$\begin{aligned}
z_c &= k_x x_c + k_y y_c + z_H \\
\dot{z}_c &= k_x \dot{x}_c + k_y \dot{y}_c,
\end{aligned} \tag{3.7}$$

under which the model becomes

$$\begin{aligned}
\dot{x}_c &= \frac{L^y}{mz_H} + \frac{k_y}{z_H}(x_c\dot{y}_c - y_c\dot{x}_c) - \frac{L^y_c}{mz_H} \\
\dot{y}_c &= -\frac{L^x}{mz_H} - \frac{k_x}{z_H}(x_c\dot{y}_c - y_c\dot{x}_c) + \frac{L^x_c}{mz_H} \\
\dot{L}^x &= -mgy_c \\
\dot{L}^y &= mgx_c.
\end{aligned} \tag{3.8}$$

The slope of the $x-y$ ground plane is represented by $k_x = \tan\alpha_x$ and $k_y = \tan\alpha_y$, respectively. The constant ground height parameter is represented by z_H (as shown in Fig. 3.3). By the equation $L = L_c + [x_c, y_c, z_c]^T \times m \ [\dot{x}_c, \dot{y}_c, \dot{z}_c]^T$, the above can be rewritten to make L^z and L_c explicit,

$$\begin{aligned}
\dot{x}_c &= \frac{L^y}{mz_H} + \frac{k_y}{mz_H}(L^z - L^z_c) - \frac{L^y_c}{mz_H} \\
\dot{y}_c &= -\frac{L^x}{mz_H} - \frac{k_x}{mz_H}(L^z - L^z_c) + \frac{L^x_c}{mz_H} \\
\dot{L}^x &= -mgy_c \\
\dot{L}^y &= mgx_c.
\end{aligned} \tag{3.9}$$

In [8, 41], it has been shown that both L^x_c and L^y_c are small compared to L^x and L^y, respectively. By using L^x and L^y as state variables in place of the CoM velocities, neglecting L_c has only a small effect on the dynamic accuracy during normal walking, even for robots with *heavy* legs.

Next, we make the case that $(L^z - L^z_c)$, which is the same as $(x_c\dot{y}_c - y_c\dot{x}_c)$, can be neglected. Some readers might already believe $(L^z - L^z_c)$ is a small term. For others, there are two ways to look at it intuitively: 1) When a robot is walking purely longitudinally $(y_c = \dot{y}_c = 0)$ or laterally $(x_c = \dot{x}_c = 0)$ the product is zero. For diagonal movement, we can define a new frame aligned with the walking direction, making the y_c and $\dot{y}_c$ terms small. 2) If we project the position vector and velocity vector to a horizontal plane, then $(x_c\dot{y}_c - y_c\dot{x}_c)$ is the cross product of these two projected vectors. Throughout a walking gait, either the angle between the two projected vectors is small, or the magnitude of the projected position vector is small. *Note:* Similar approximations are not needed in [29] because the coupling term $(x_c\ddot{y}_c - y_c\ddot{x}_c)$ can be nullified by the assumption of a point contact (zero torque applied).

As a result of these approximations, we arrive at the dynamics for the 3D-ALIP model with CoM evolving as in (3.7),

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{L}^x \\ \dot{L}^y \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & \frac{1}{mz_H} \\ 0 & 0 & -\frac{1}{mz_H} & 0 \\ 0 & -mg & 0 & 0 \\ mg & 0 & 0 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_c \\ y_c \\ L^x \\ L^y \end{bmatrix}}_{\mathbf{x}}. \tag{3.10}$$

While the model (3.10) is ultimately the same as in [8, 41], these references do not provide a derivation of the model (3.8) and (3.9) nor do they explain the simplifications needed to arrive at (3.10). We reiterate that the state includes the angular momenta rather than the CoM velocities; the benefits of this selection have been highlighted in several related publications [80, 70, 81, 8].

3.2.3 Foot Placement as a Control Variable

The dynamic model (3.10), which describes the evolution of the centroidal dynamics when the robot is in single support, is not affected by the motor torques[2]. So how to control it? As in [82, 83, 81, 8], we use the placement of the end of the swing leg as a step-to-step actuator. Under conservation of angular momentum and (3.7), if $\mathbf{x}^-$ is the solution of (3.10) just before impact, and $\mathbf{x}^+$ is the value

[2]Recall that we are leaving the stance ankle passive in this study.

of the state just after the (instantaneous) impact, then

$$\mathbf{x}^{+} = \mathbf{x}^{-} + B\mathbf{u}_{fp}, \tag{3.11}$$

where

$$B = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}^{T} \tag{3.12}$$

and $\mathbf{u}_{fp}$ is the foot placement (or resultant vector emanating from the current stance contact point to the desired swing foot impact location), and B represents the instantaneous change in coordinate systems to the new stance foot.

We assume that the height of the swing leg is controlled so that the duration of each robot step is fixed at T_s. Hence, during the k-th step of the robot, the height of the swing leg above the ground is regulated to be positive for $(k-1)T_s < t < kT_s$ and zero at kT_s. The relative (x, y) position of the swing leg end at time kT_s is selected to achieve a desired evolution of (3.10) for $t > kT_s$. Fig. 3.3 shows a planar schematic of the x-component of the 3D-ALIP.

Remark: Reduced-order models similar to (3.10) and (3.11) were used in [44] for a foot placement controller based on the LIP model, which is parameterized by CoM velocity instead of angular momentum. Moreover, the models for the impact dynamics are based on the same conservation of angular momentum assumption. Nevertheless, our use of angular momentum is not a "matter of taste"; it is the better state choice for controlling a variety of bipedal robots as explained in [41]: (a) the CoM velocity is more sensitive to motor torque transients because it is relative degree one as opposed to angular momentum, which is relative degree three; (b) linear velocity does not capture the natural interchange between linear momentum and angular momentum about the center of mass; and (c), there are typically large jumps in the velocity variables at impact for fast-moving bipedal robots. To predict this jump, all states of the robot and the impact model need to be known, which makes the online prediction infeasible in practice. Therefore, an assumption of continuity is widely adopted. This is much less of an issue for angular momentum about the contact point because it is invariant to the impulse generated at contact. Hence, even the impact model functions better in the angular momentum coordinates.

3.3 MPC Formulation for Foot Placement Control of 3D-ALIP

Our next goal is to compute desired footholds for the robot to execute to converge to some desired (centroidal) state at the end of each step. In [8], a single $\mathbf{u}_{fp}$ was selected step-to-step to achieve a desired value of the angular momenta in (3.10) at the end of the next step. The values of x_c and y_c were not regulated, and constraints on self-collisions and the friction cone were ignored. The one-step-ahead foot placement method in [8] can be viewed as MPC with a terminal linear equality constraint on a portion of the state vector and no inequality constraints. Here, we will develop an MPC formulation of foot placement control over a multi-step horizon, a quadratic cost to be minimized, and appropriate linear inequality constraints to avoid self-collisions for a solution terrain-aware solution.

We begin by defining

- N_s, the number of (robot) steps of (fixed) duration T_s in the MPC control horizon.

- $\delta t = T_s / N_{\delta t}$ is the sample period for the controller, where $N_{\delta t} > 1$ is an integer.

- $A_{\delta t} := \exp(A\delta t)$, the state transition matrix of (3.10) for a time duration of δt seconds.

- $\mathbf{x}_0$ in the MPC will always be the predicted solution $\widehat{\mathbf{x}}(kT_s, t)$ of (3.10) just before the impact of the current step, based on the measured value of state at time t, that is,

$$\mathbf{x}_0 := \widehat{\mathbf{x}}(kT_s, t) := \exp(A(kT_s - t))\mathbf{x}(t). \tag{3.13}$$

This will allow us to work with a fixed control horizon.

- Next, we define the discrete-time dynamics

$$\mathbf{x}_{i+1} = \begin{cases} A_{\delta t}(\mathbf{x}_i + B\mathbf{u}_{fp,i}), & i = jN_{\delta t}, \\ & 0 \leq j \leq (N_s - 1) \\ A_{\delta t}\mathbf{x}_i, & \texttt{otherwise} \end{cases} \tag{3.14}$$

  for use in our MPC problem, allowing us to place constraints on the intra-step evolution of $x(t)$, that is, its behavior between steps.

- $\mathbf{x}_i^{\text{des}}$ is the desired evolution of the state, and the associated error term is

$$\mathbf{x}_{e,i} := \mathbf{x}_i - \mathbf{x}_i^{\text{des}}. \tag{3.15}$$

3.3.1 MPC Formulation

An N_s-step horizon MPC control problem with quadratic cost and linear constraints can now be formulated as

$$\begin{aligned} &\min_{\mathbf{U}_{fp}} J = \sum_{i=0}^{N_{\delta t}N_s-1} \mathbf{x}_{e,i}^T Q_i \mathbf{x}_{e,i} + \mathbf{x}_{e,N_{\delta t}N_s}^T Q_f \mathbf{x}_{e,N_{\delta t}N_s} \\ &\textbf{subject to} \\ &(3.13), (3.14), \text{ and } (3.15) \\ &\forall\ \mathbf{x}_i \in \mathcal{X} \text{ and } \forall\ \mathbf{u}_{fp,i} \in \mathcal{U}, \end{aligned} \tag{3.16}$$

where $\mathbf{U}_{fp} = \left[\mathbf{u}_{fp,0}, \mathbf{u}_{fp,N_{\delta t}}, \ldots, \mathbf{u}_{fp,N_{\delta t}(N_s-1)}\right]$. The solution returns the optimal foot placement sequence. Only the sequence's first value $\mathbf{u}_{fp,0}$ is applied. The state and control constraint sets, $\mathcal{X}$ and $\mathcal{U}$, respectively, are defined in the following subsection.

3.3.2 Constraint Sets

The state constraint set $\mathcal{X}$ is the union of the mechanical safety set $\mathcal{X}^{\text{mech}}$ and the friction cone safety set $\mathcal{X}^{\text{slip}}$. The foot placement safety set $\mathcal{U}$ is also used to prevent foot collisions, most notably in the lateral direction. The mechanical safety and foot placement safety sets are constructed as box constraints related to the geometrical limitations of the robot. The ground friction cone (based on the 3D-ALIP dynamics) constrains the intra-step CoM positions.

We derive the constraint for x_c, assuming the ground slope is purely in the sagittal direction. This model applies the ground reaction force collinearly through the contact leg because the point mass moves parallel to the ground plane. Taking advantage of the known slope of the terrain, we derive the resultant tangent and normal forces with respect to the ground to be

$$\begin{bmatrix} F_{T_x} \\ F_{N_x} \end{bmatrix} = \begin{bmatrix} \cos\alpha_x & \sin\alpha_x \\ -\sin\alpha_x & \cos\alpha_x \end{bmatrix} \begin{bmatrix} F_x \\ F_z \end{bmatrix}. \tag{3.17}$$

Given the defined motion constraints, we compute the relative force ratios

$$F_{x/z} = \frac{F_x}{F_z} = \frac{x_c}{k_x x_c + z_H} \tag{3.18}$$

and combine this with a Coulomb static friction constraint ($|F_{T_x}| \leq \mu F_{N_x}$) to compute the slip constraint on x_c,

$$|F_{x/z} + k_x| \quad \leq -\mu k_x F_{x/z} + \mu. \tag{3.19}$$

For known k_x and μ (friction coefficient), the explicit constraint is given as

$$|x_c| \;\; \leq \frac{(\mu - k_x) z_H}{1 + k_x^2} = x_c^{\text{slip}}. \tag{3.20}$$

For ground slope purely in the lateral direction and known k_y, a similar constraint on y_c can be derived.

We emphasize that this is not an exact friction slip constraint due to the simplifications of using the 3D-ALIP for ground reaction force estimation. It can, however, be combined with an under-approximation of the friction coefficient μ to enable safer foot placements. For example, multiplying μ by $\frac{1}{\sqrt{2}}$ in (3.20) results in a linearized under-approximation of the Coulumb cone.

3.3.3 Cost Design

The cost function is the sum of a running cost with non-zero weights at step transitions, that is, $Q_i = \mathbf{0}, \forall i \notin \{N_{\delta t}, \ldots, (N_s - 1)N_{\delta t}\}$ and a terminal cost Q_f. Given a desired longitudinal angular momentum $L^{y,\text{des}}$ and step width W, we can use the solutions of (3.10) to compute the desired state of the corresponding 2-step periodic orbit for the corresponding stance foot (by following [8]). With the assumption of conservation of angular momentum about the contact point, we substitute $L^{y,\text{des}} = L^y(0) = L^y(T_s)$, $L^x(0) = -L^x(T_s)$, and $y_c(0) = {}^{W}\!/_{2}$ into the trajectory solutions of (3.10) and solve the resultant linear system of equations. The resultant desired state at each impact is

$$\mathbf{x}_i^{\text{des}} = \begin{bmatrix} \frac{1}{m z_H \ell} \tanh({}^{\ell T_s}\!/_{2}) L^{y,\text{des}} \\ -\frac{1}{2}\sigma W \\ \frac{1}{2}\sigma m H \ell W \tanh({}^{\ell T_s}\!/_{2}) + L^{x,\text{offset}} \\ L^{y,\text{des}} \end{bmatrix}, \tag{3.21}$$

where $\ell = \sqrt{g/z_H}$, $L^{x,\text{offset}}$ is an additional lateral angular momentum term, and σ is +1 for left stance and -1 for right stance. Without $L^{x,\text{offset}}$, the controller will walk nominally with zero lateral velocity.

The terminal cost Q_f is computed as the optimal cost-to-go of a Discrete-Time Algebraic Ricatti Equation (DARE) for a periodic 2-step trajectory including impact, combining (3.10) and (3.11), and ignoring constraints. This selection of terminal cost ensures recursive feasibility via Bellman's principle of optimality [12].

3.4 Virtual Constraints and Foot Placement Implementation on Cassie

The computed foot placement solution is implemented on the 20-DoF bipedal Cassie robot using user-defined designed virtual constraints. As documented in [8], an important feature of the 3D-ALIP model is that the mass of the swing leg and its corresponding momentum are accounted for in L^x and L^y.

Cassie is a 32 kg, 20-DoF biped robot actuated at ten joints. Each leg has seven joints, five of which are actuated while the remaining two are constrained by springs [40]. To achieve a desired foot placement, we must define the control variables and generate their reference trajectories. The nine control variables h and the corresponding references h_d are defined as follows:

$$h = \begin{bmatrix} \text{torso pitch} \\ \text{torso roll} \\ \text{stance hip yaw} \\ \text{swing hip yaw} \\ p^z_{\text{CoM}_{\text{proj}} \to \text{CoM}} \\ p^x_{\text{st} \to \text{sw}} \\ p^y_{\text{st} \to \text{sw}} \\ p^z_{\text{st} \to \text{sw}} \\ \text{absolute swing toe pitch} \end{bmatrix} \tag{3.22}$$

The desired reference trajectories are parametrized by a time-based phase variable $s = {}^{(T_s - t)}/_{T_s}$ where t is the time since the last impact. We set the reference values for torso pitch and torso roll to be zero. To enable turning, one-half of the total desired turn angle $\Delta\psi$ at step end is set as the reference position for both stance and swing yaw motor joints at the end of the current step [8]. The reference absolute swing toe pitch angle is adjusted to align with the terrain slope. $p_{\text{st}\to\text{sw}}$ is the position vector of the swing leg toe relative to the stance leg toe and $p^z_{\text{CoM}_{\text{proj}} \to \text{CoM}}$ represents the constant height parameter z_H, between the center of mass and an *inclined* ground.

$$h_d(s) := \begin{bmatrix} 0 \\ 0 \\ (1-s)h_3^{\text{init}} - s(\frac{1}{2}\Delta\psi) \\ (1-s)h_4^{\text{init}} + s(\frac{1}{2}\Delta\psi) \\ z_H \\ \frac{1}{2}[(1+\cos(\pi s))h_6^{\text{init}} + (1-\cos(\pi s))p_{\text{st}\to\text{sw}}^{x,\text{des}}] \\ \frac{1}{2}[(1+\cos(\pi s))h_7^{\text{init}} + (1-\cos(\pi s))p_{\text{st}\to\text{sw}}^{y,\text{des}}] \\ \beta_1 s^2 + \beta_2 s + \beta_3 \\ k_x \end{bmatrix}. \tag{3.23}$$

Remark: The virtual constraints in [8] regulate the CoM to remain at a constant height with respect to the pinned stance foot and do not account for the CoM height constraint used in this paper. We instead derive a new kinematic relation $p^z_{\text{CoM}_{\text{proj}}\to\text{CoM}}$ that computes the height of the CoM relative to the projected position on the terrain. More explicitly, $p^z_{\text{CoM}_{\text{proj}}\to\text{CoM}} = p^z_{\text{st}\to\text{CoM}} - k_x p^x_{\text{st}\to\text{CoM}} - k_y p^y_{\text{st}\to\text{CoM}}$. The desired swing toe angle is modified to align with the ground.

Outputs, $p^{x,y}_{\text{st}\to\text{sw}}$ are set equal to the MPC foot placement solution $\mathbf{u}_{fp,0}$ described in (3.16). The z component of $p_{\text{st}\to\text{sw}}$ can be easily computed with the knowledge of k_x and k_y. We use sinusoidal references for the x and y components (following [8]) and a parabola for the z component parametrized by the initial and final heights of the swing leg determined by foot placement and the relative time and height of a user-defined step clearance (The β_i parameter derivations are omitted due to space constraints). h_i^{init} denotes the value of each output at the beginning of each new step.

We implement an inverse kinematics and passivity-based control schema to track these constraints on the physical robot (see [8, 41]).

3.5 Simulation Results

This section demonstrates the performance of the ALIP-MPC walking controller in simulation experiments.[3] It also compares the new controller with a previous one-step-ahead (or ALIP-1step) controller implemented on Cassie in [8].

Simulation is used to demonstrate the advantages of the proposed gait ALIP-MPC controller with respect to the ALIP-1step controller in [8]. The 20-DoF simulation model and the ALIP model are identical in both cases, except that the ALIP model used in the MPC controller takes the slope into account, following general motion constraints in [29].

Fig. 3.4 highlights the importance of the ALIP model and modified virtual constraints accounting for slopes in the lateral plane. On terrain with a 5° lateral slope (Fig. 3.4a), the MPC controller achieves an average lateral velocity close to the desired zero velocity reference because it adapts to the slope, while in contrast with the one-step-ahead controller, the robot drifts downhill. The drift is caused by untimely impacts and unbalanced forces on the uphill vs. downhill contacts. The virtual constraints of each controller are designed to zero the vertical velocity of the CoM. With the MPC controller, v_z has only a small oscillation caused by imperfect low-level tracking on the 20-DoF model. Lateral walking at 0.5 m/s is compared in Fig. 3.4b, with similar results. In short, the improvements over [8] allow walking over steeper terrain.

Fig. 3.5 illustrates how changing the prediction horizon's length affects the ALIP-MPC controller's ability to satisfy constraints. The friction parameter is modified online, and we confirm the benefits of using a larger horizon for safer walking when imposing step length restrictions $u_{fp}^{\max}$.

[3]Open-source code, videos, and results can be found at https://github.com/UMich-BipedLab/cassie_alip_mpc.

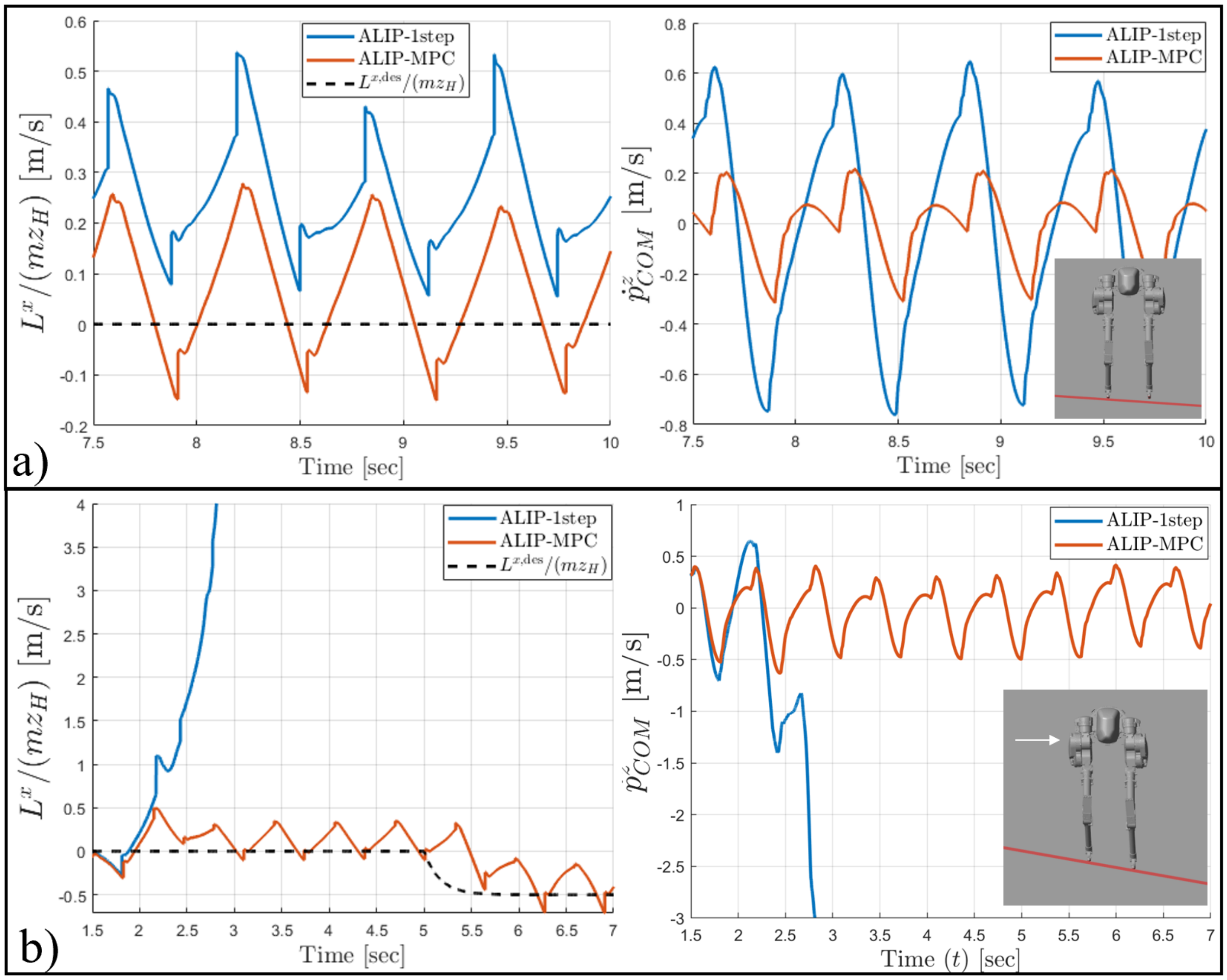


Figure 3.4: Comparison between proposed ALIP-MPC controller and ALIP-1step (one-step-ahead) controller [8] on inclines. (a) The ground has a 5° lateral slope, and Cassie is commanded to walk with zero velocity. The lack of slope information in the ALIP-1step controller leads to an increase in the magnitude of the CoM velocity and increased tracking error of the constant CoM height assumption. In (b), a ground incline of 11° causes the ALIP-1step controller to fail, while the ALIP-MPC controller allows Cassie to walk downhill laterally at approximately 0.5 m/s.

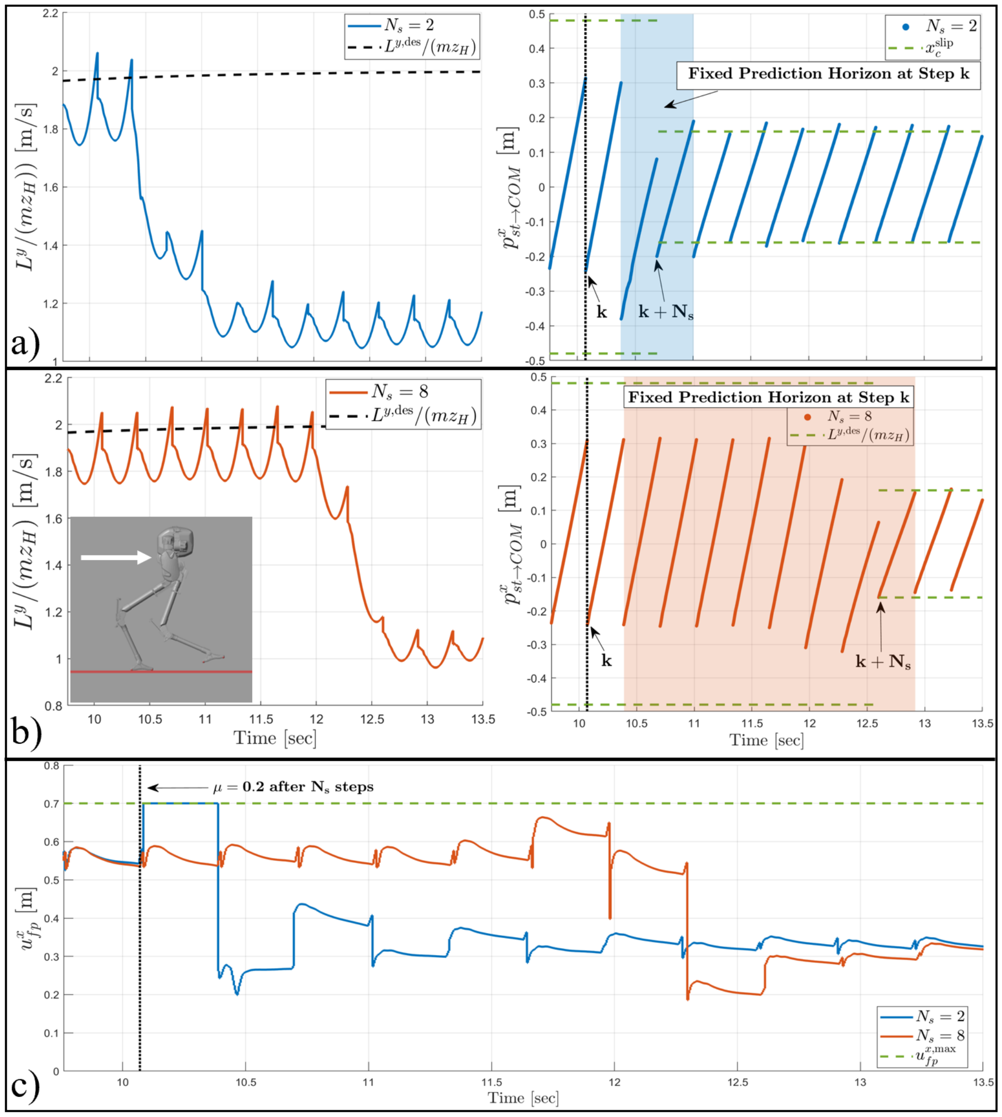


Figure 3.5: Comparison between (a) the 2-step horizon and (b) the 8-step horizon implementation of the proposed ALIP-MPC controller. At $t = 10.1$, both controllers are informed that the friction coefficient will reduce to 0.2 at the end of their horizons. In c), we compare the MPC foot placement solutions calculated during each step for both planning horizon choices. The 2-step version is restricted from extending its swing leg too far, which results in a friction violation at step $k + N_s$. The 8-step version has more time to reduce the robot's velocity to satisfy the constraint and only uses one additional step to affect the change. The slight friction cone violations are from the ALIP approximation, reinforcing the need for an under-approximated friction coefficient.

3.6 ALIP-MPC Benefits

The benefits of including slope incline within the proposed ALIP-MPC controller are twofold. Firstly, the constant height constraint is embedded within the 3D-ALIP model, instead of the flat ground assumption, to more accurately approximate the tangential contact forces. We then formulate an MPC problem with constraints imposed on these forces to compute foot placements, which prevent slipping. Secondly, we modify the virtual constraints of [8] to include slope data. Fig. 3.4 highlights this improvement's benefits in reducing CoM height variation and untimely contacts, leading to improved velocity tracking.

3.7 Experimental Results

The proposed ALIP-MPC controller was coded in `C++` and run on a secondary computer in a Linux environment.[4] The planning horizon of the controller was set to four steps ($N_s = 4$) with a step period of 0.3 seconds and an intra-step time discretization of 10 ms. Foot placement updates were sent over UDP to the primary computer on Cassie at 250 Hz. The resultant QP was code-generated using CasADi and evaluated using a primal-dual active set algorithm [84, 85].

The MPC controller was implemented on Cassie Blue and evaluated in a variety of situations shown in Fig. 3.6, 3.14, and 3.15.

3.7.1 Inclined walking

Cassie walked forward on a treadmill inclined at 6° at a maximum speed of 1.5 m/s and also walked laterally at a maximum speed of 0.5 m/s on a stationary treadmill inclined at 13°. When we tried lateral walking with the one-step ahead controller of [8], which does not actively constrain the workspace of the legs, the

[4]Open-source code, videos, and results can be found at https://github.com/UMich-BipedLab/cassie_alip_mpc.

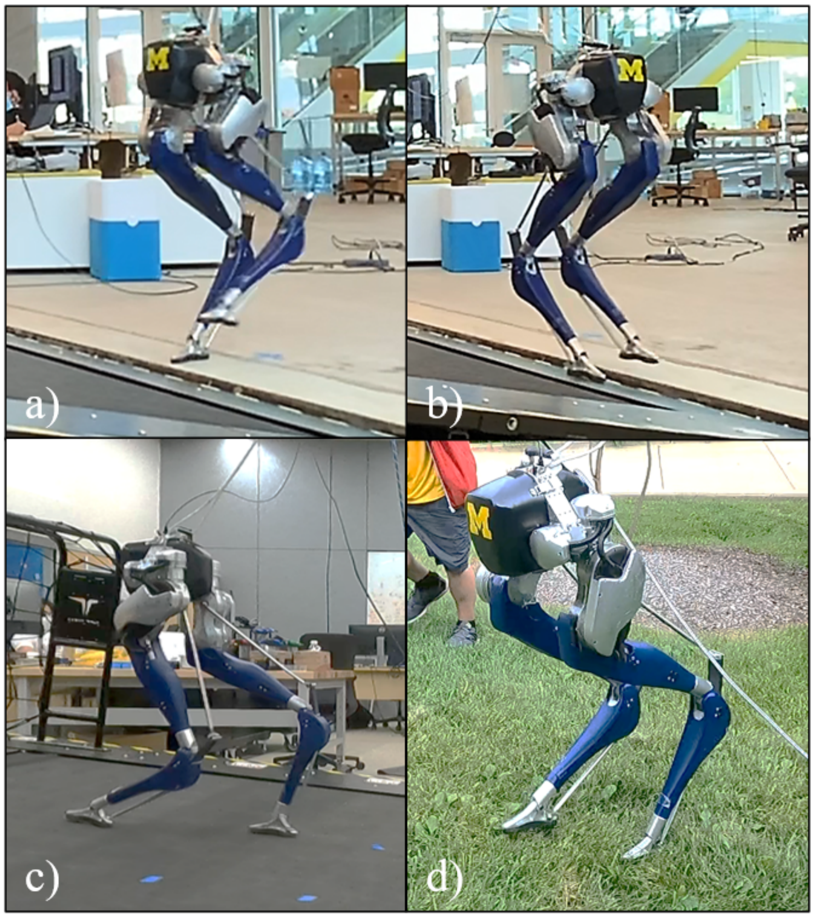


Figure 3.6: Images from various experiments performed with Cassie Blue using the gait controller discussed in this paper. (a) Forward walking at 1.0 m/s with a transition onto a stationary treadmill inclined at 13°. (b) Lateral walking at 0.5 m/s with a transition onto a stationary treadmill inclined at 13°. (c) Forward walking (max 2.1 m/s) on a moving treadmill inclined at 6°. (d) Forward walking at 1 m/s on a wet, grassy slope inclined at 22°. Videos available at [9].

robot tripped and fell. We test the limits of our control method by having Cassie Blue walk up an uneven slope with an estimated average incline of 22°. Cassie successfully walked up the slope in the sagittal direction at 1.0 m/s and the lateral direction at 0.3 m/s. Speed in the lateral direction is inherently slower due to hardware limits on step width.

3.7.2 Transitioning from Flat Ground to an Incline

In this experiment, an operator sent the slope information to Cassie at the transition. Performance relied heavily on the timing and accuracy to which the operator switched the estimated ground slope with respect to the body frame of the robot. While the transitions were not ideal due to operator error, a noticeable improvement in maintaining a constant CoM height with respect to the ground was seen compared to [8]. Future work will remove operator dependence and integrate with perception and [56].

3.7.3 Rapid Changes in Lateral Velocity

To validate the ability of the MPC controller to achieve self-collision constraints, as in Sect. 3.3, we constrain the lateral foot placement solution to remain within the safety set $\mathcal{U}$ for all experiments. As shown in [9], the swing legs avoid collisions when rapidly changing the lateral target velocities.

3.7.4 Avoid Slipping on a Snow-Ice Mixture

Via the RC transmitter, the operator adjusted the assumed friction coefficient to prevent slipping on the snow-ice mixture shown in Fig. 3.14. With poor estimation (or omission) of the x_c^{slip} constraint, the robot slips and falls. This is corrected by underestimating the friction coefficient for successful walking, as shown in Fig. 3.15.

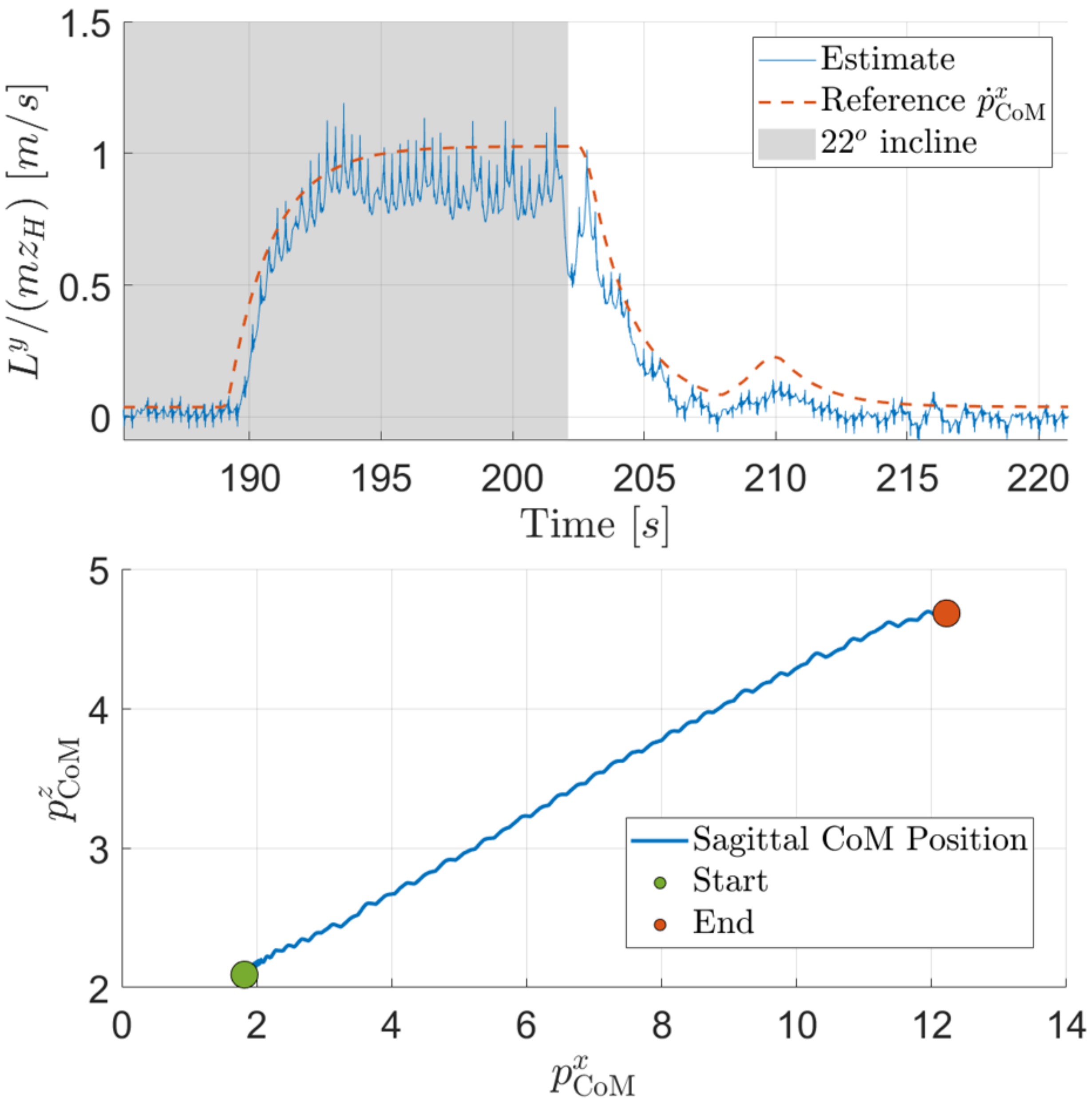
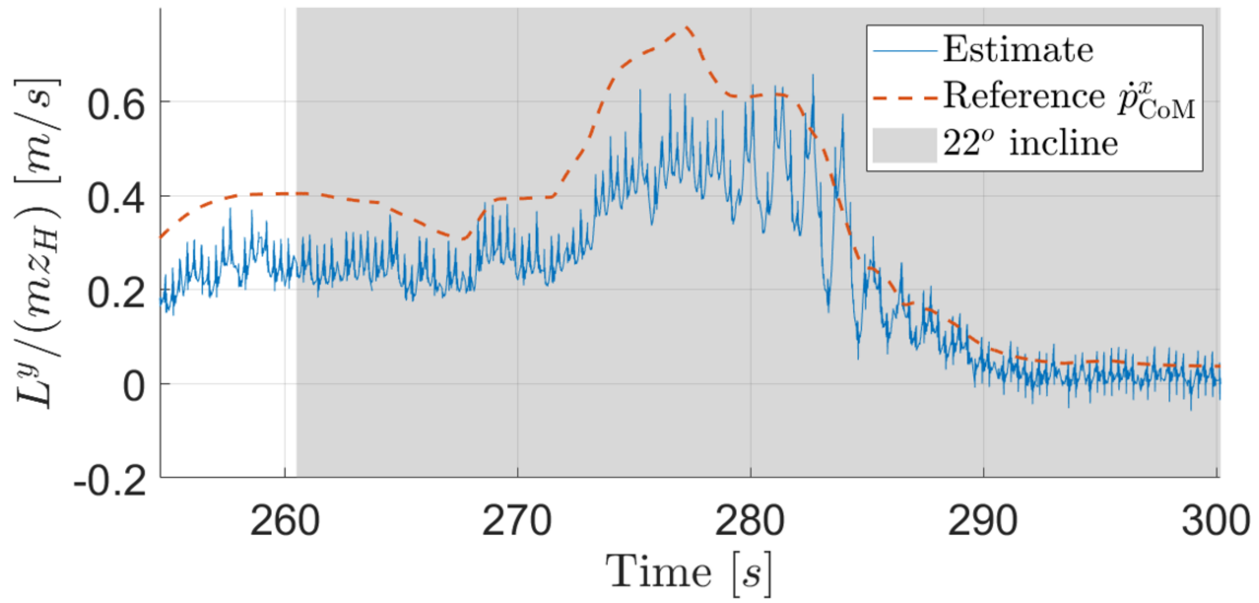


Figure 3.7: Cassie sagittal walking uphill 22° incline wet-grass hill.



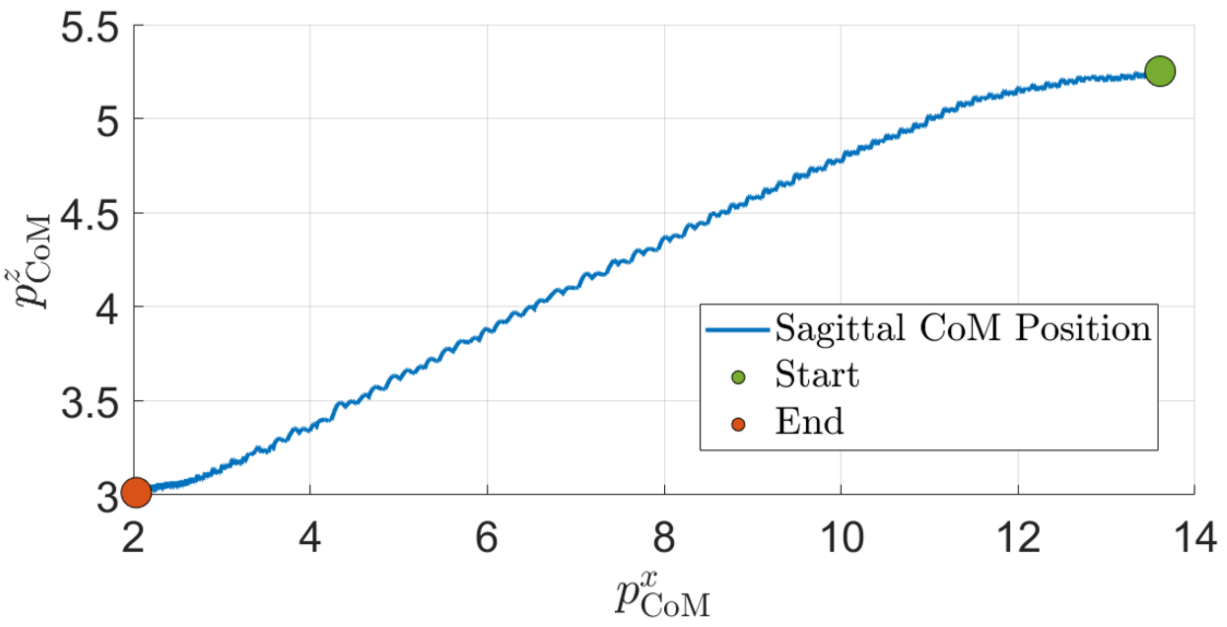
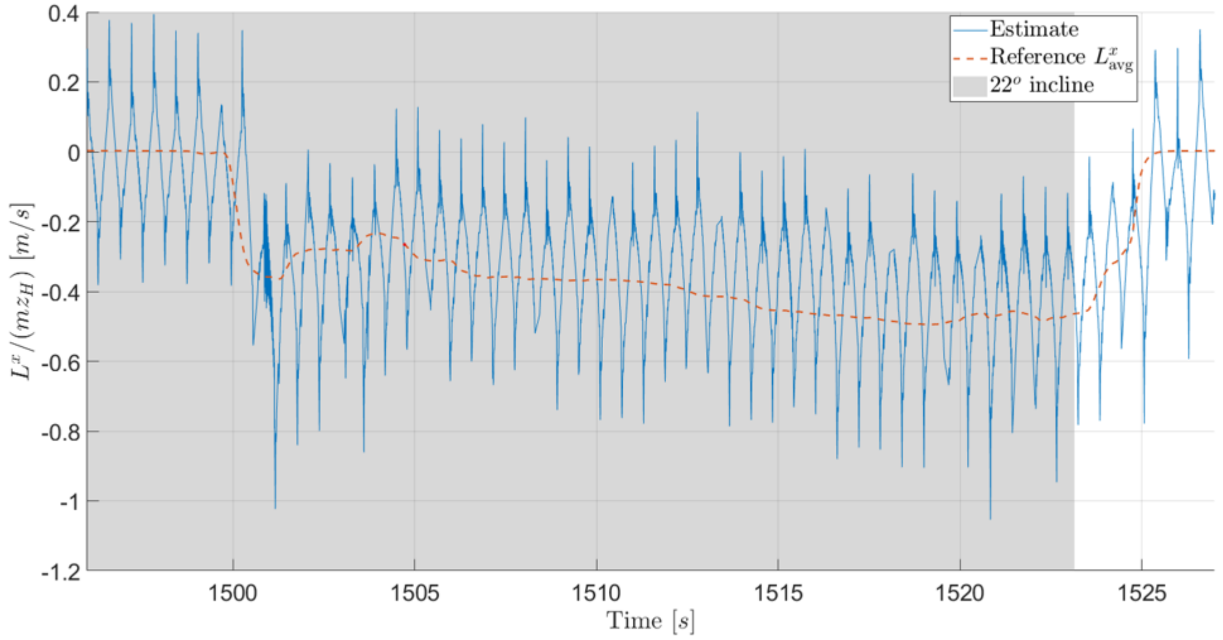


Figure 3.8: Cassie sagittal walking downhill 22° incline wet-grass hill.



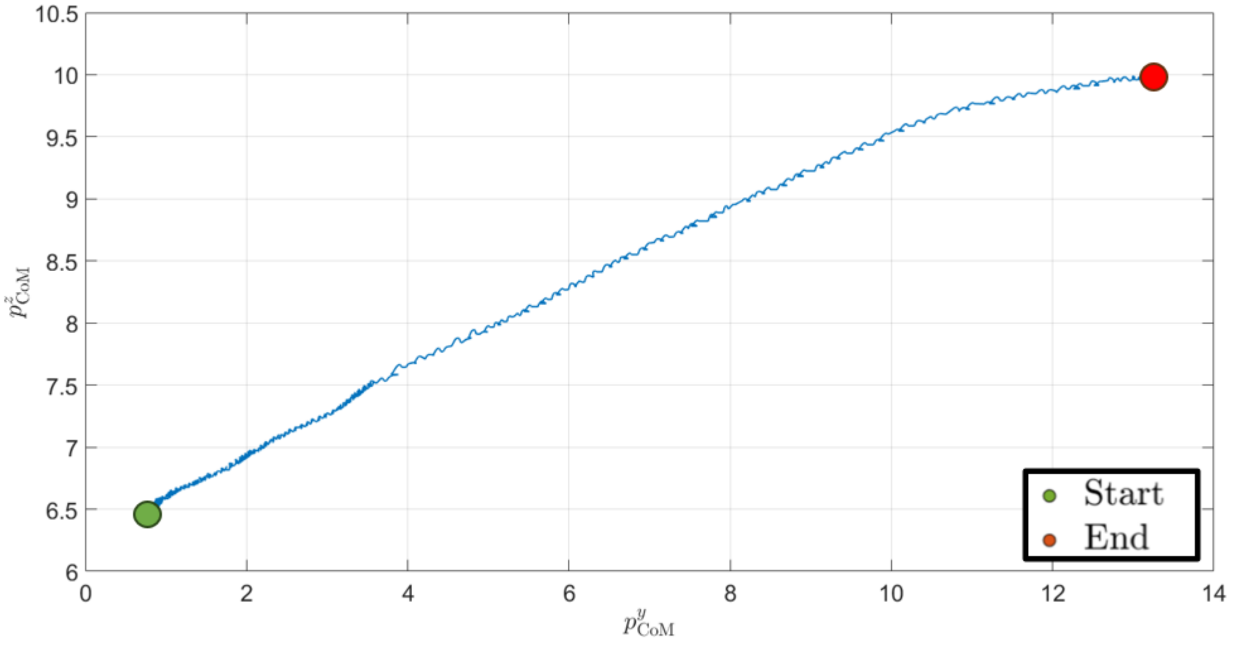


Figure 3.9: Cassie lateral walking uphill 22° incline wet-grass hill.

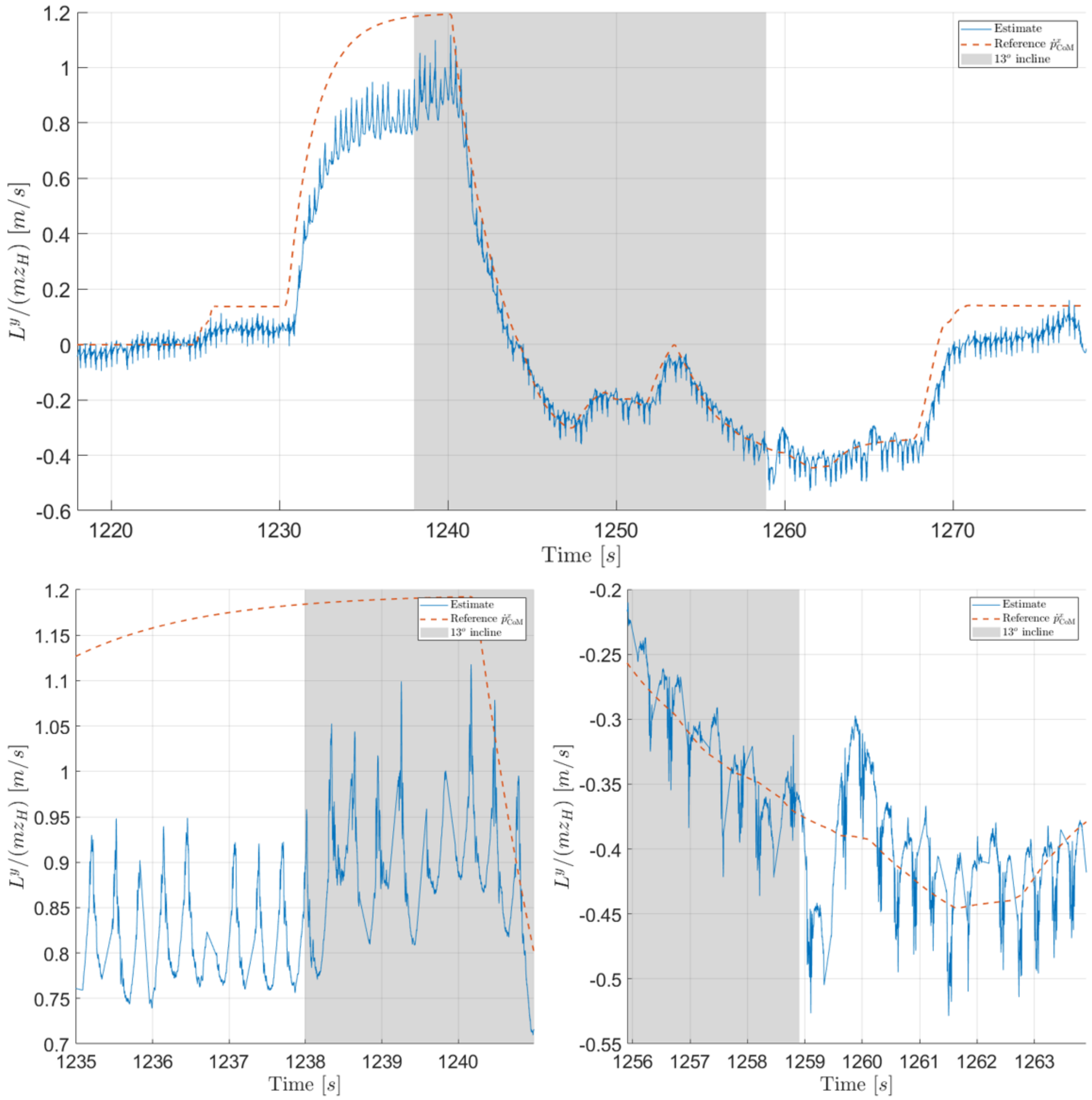


Figure 3.10: Cassie Indoor sagittal walking transition from flat ground to 13° incline. Plot of L^y shown.

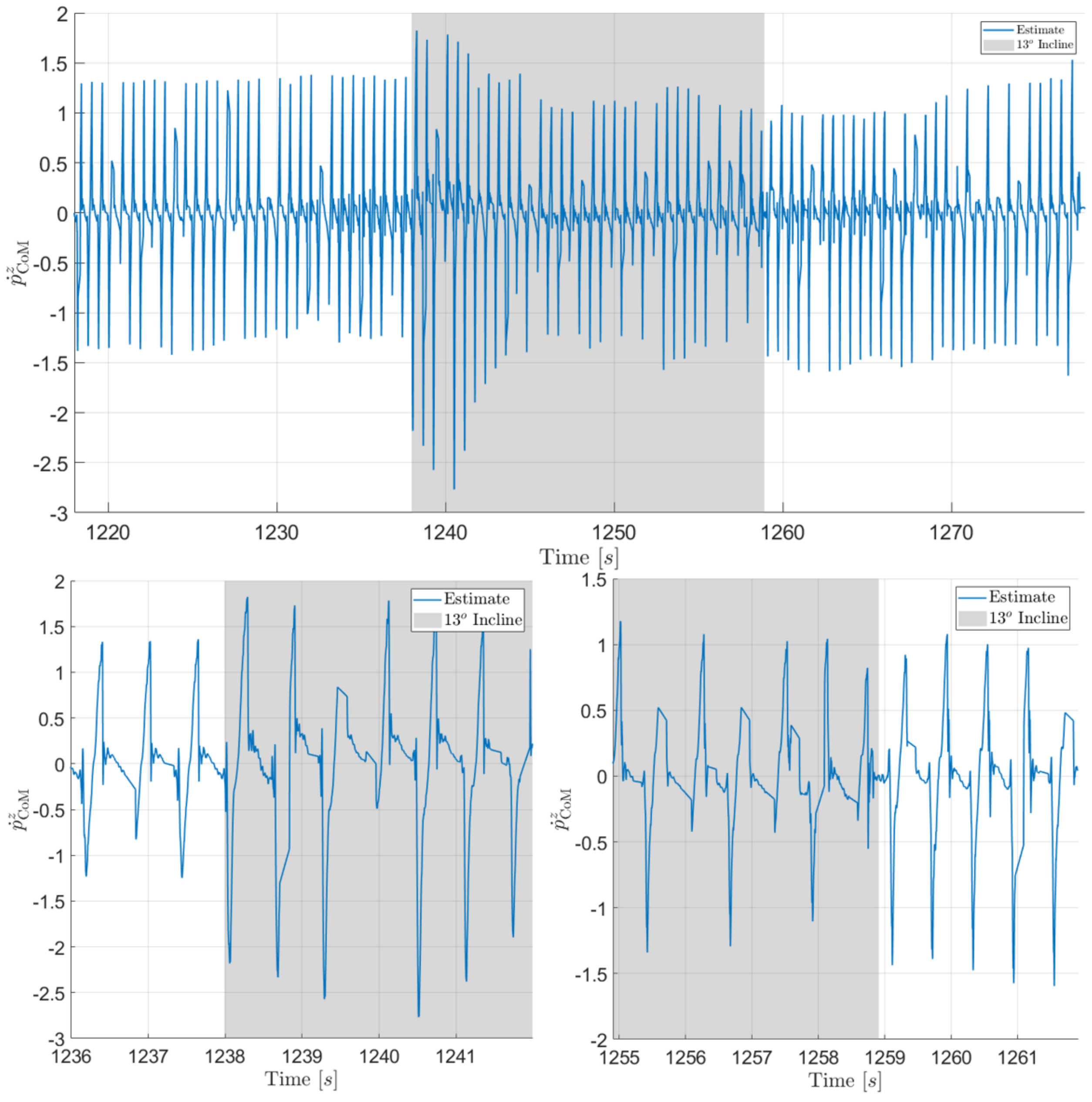


Figure 3.11: Cassie Indoor sagittal walking transition from flat ground to 13° incline. Plot of $\dot{p}^z_{\mathrm{CoM}}$ shown.

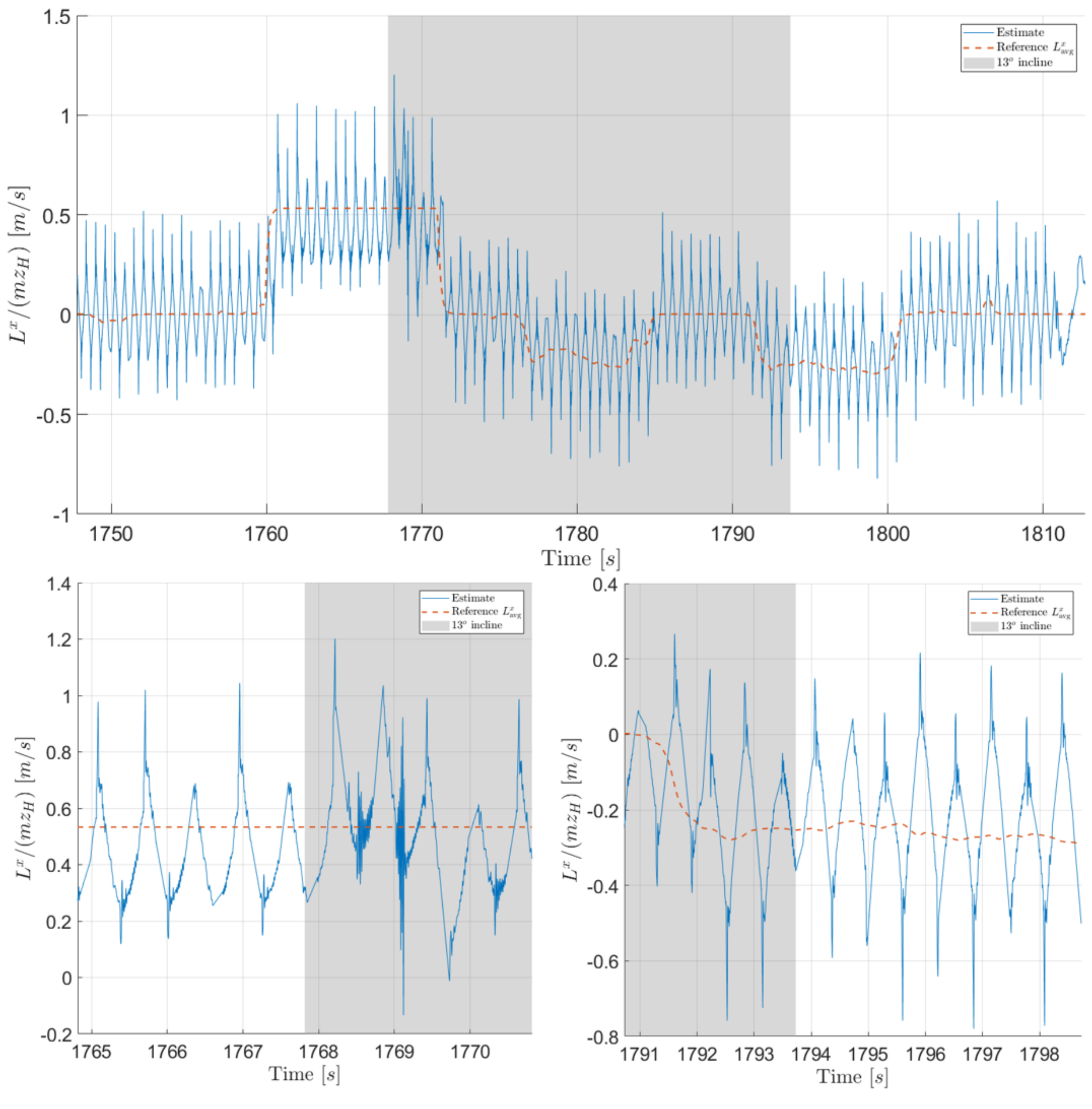


Figure 3.12: Cassie Indoor lateral walking transition from flat ground to 13° incline. Plot of L^x shown.

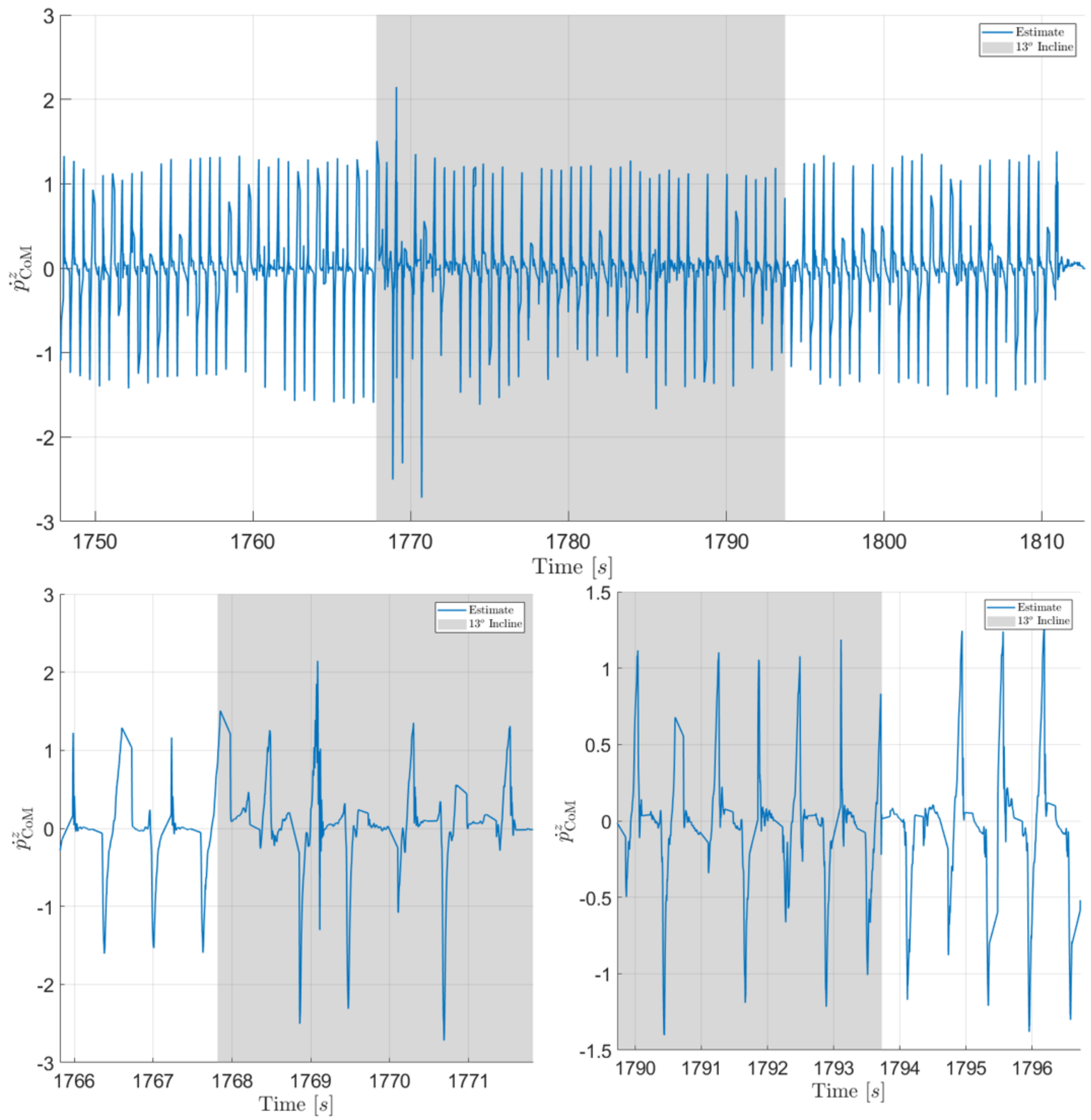


Figure 3.13: Cassie Indoor lateral walking transition from flat ground to 13° incline. Plot of $\dot{p}^z_{\text{CoM}}$ shown.

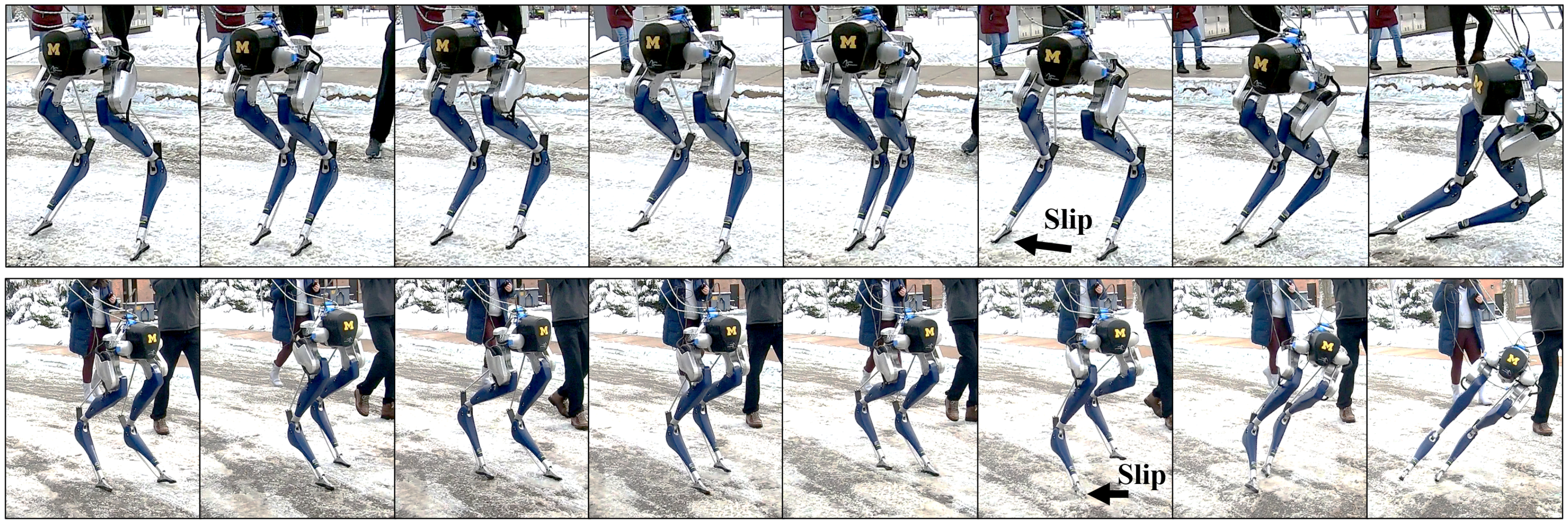


Figure 3.14: Failure of 1 m/s lateral walking on snow due to overestimated coefficient of friction. The friction coefficient parameter in the ALIP-MPC controller was set to $\mu = 1.0$. Cassie slips when the lateral component of the Center of Mass deviates too far from the contact point.

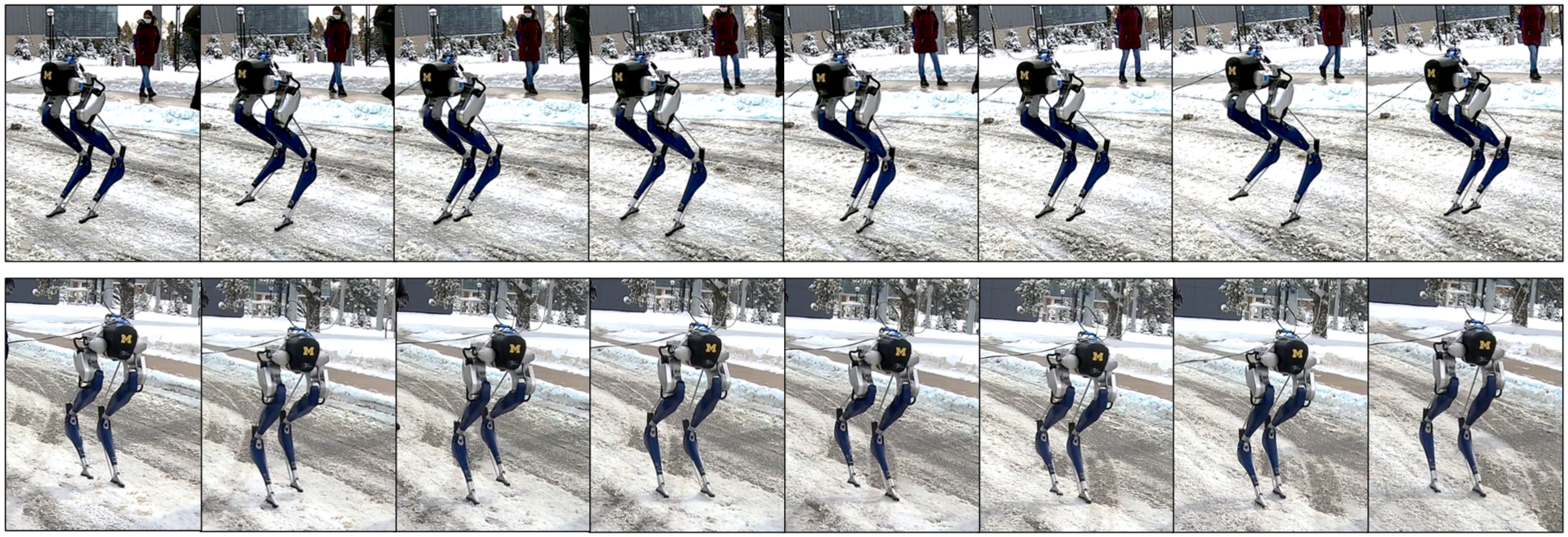


Figure 3.15: "Successful 1.0 m/s lateral walking on mixed snow terrain with an underestimated coefficient of friction. The friction coefficient was set to $\mu = 0.6$. Adopting a more conservative friction estimate restricted the robot's foot placements, ensuring an optimal safe speed is maintained during locomotion.

3.8 Conclusions and Future Work

This paper extended a controller based on a constant CoM height and a one-step ahead prediction of angular momentum [8] in three ways. First, the 3D-ALIP model was derived to allow the robot's center of mass to exhibit piecewise planar motion. Similar to the LIP in [29], the resulting model more closely models a physical robot. Second, a four-step ahead MPC controller was provided, which importantly allowed realistic workspace and terrain-centric constraints in the MPC formulation. Lastly, a novel set of virtual constraints allowed us to experimentally realize the assumed CoM properties on a highly agile, 20-DoF bipedal robot.

Currently, the ALIP-MPC performance depends on the operator's ability to provide a real-time reference of the terrain slope and friction within a local region of the robot. Ideally, this information should be retrieved autonomously from a perception system, as in [79]. When the slope is estimated accurately, the robot is very stable while walking with lateral and longitudinal velocity on sloped ground. In future work, we plan to (a) improve step-to-step smoothness by appending a rate-limiter term to the cost function, (b) look at further relaxing the assumptions on the low-dimensional model (e.g., zero dynamics) to allow nonlinear terms, and (c) integrate the controller with a reactive planner.

Chapter 4

Perception-Integrated Bipedal Locomotion Controller for Humanoid Robots

A locomotion controller is responsible for providing a robot with actuator commands to achieve the goals prescribed by high-level reference commands. These goals encompass tasks such as balancing, manipulation, and walking. Adding exteroceptive sensors, such as LiDAR and RGB-D, to the Digit robot compared to its predecessor, Cassie, enhances its capacity for navigating complex environments. Leveraging these extended capabilities, we aim to improve the stability and performance of these robots for moving in the real world. In this chapter, we describe a novel locomotion algorithm that combines perception-derived terrain information with the ALIP-MPC foot placement and locomotion strategy described in Chapter 3. In addition to walking, locomotion controllers handle additional modes related to dynamic balance and transitioning between modes. This chapter focuses solely on walking, and additional details on balance, mode transition, and parameters can be found in Appendix B. Open-source code and implementations of this locomotion controller are available at https://github.com/UMich-BipedLab/digit_locomotion_controller.

4.1 Humanoid Locomotion System Architecture

The architecture of the humanoid locomotion system involves seamlessly integrating all requisite components—hardware and software—for humanoid robot operation.

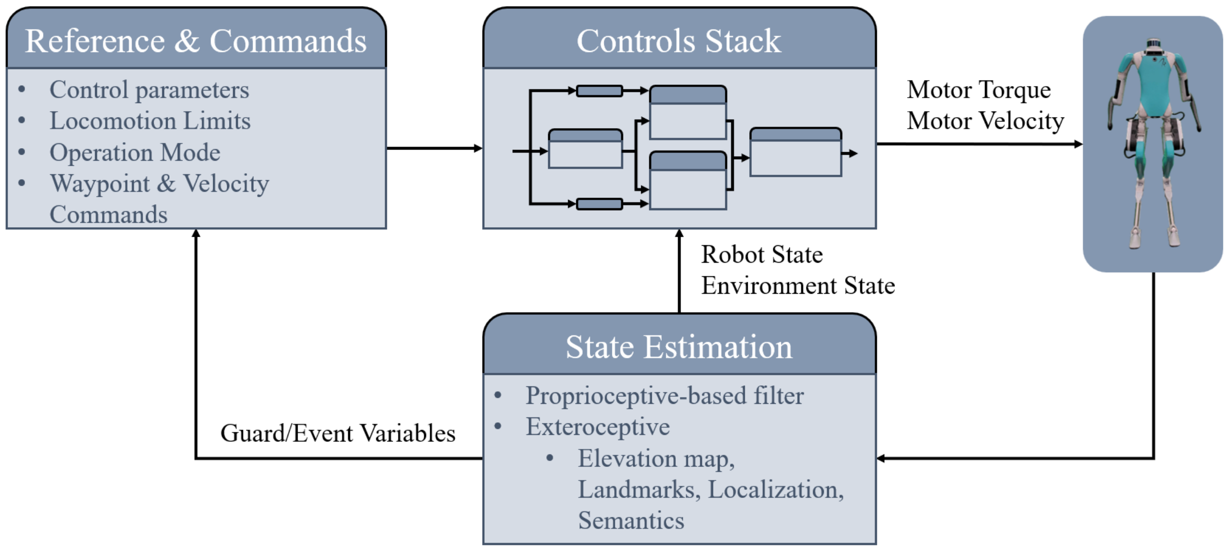


Figure 4.1: This block diagram illustrates the flow of information and control in a humanoid locomotion system. It starts with sensors providing input for estimating both the robot-centric and environmental states. References and commands drive relevant tasks and mode transitions. The control stack combines the estimated state and reference commands to compute motor commands for the actuators, resulting in the realization of locomotion on the physical humanoid robot.

At a higher level, the block diagram encompassing the control structure of these components for implementing diverse locomotion controller modes features shared elements, as depicted in Figure 4.1. The control stack is at the heart of the system, responsible for computing pertinent actuator commands, thus enabling precise robotic response to desired actions.

Figure 4.2 visually represents a control stack's internal flow. In this instance, the interplay between reference, commands, and state variables governs the choice between two locomotion modes: dynamic balance and walking. Upon selecting the pertinent task space objectives, these are meticulously tracked using user-preferred joint velocity or torque resolution methods.

Notable distinctions between balance and walking control manifest in the reference commands and task map segments. While distinct command parameters cater to each operational mode in the reference commands section, the task map section delves into the tasks essential for realizing the desired locomotion behavior.

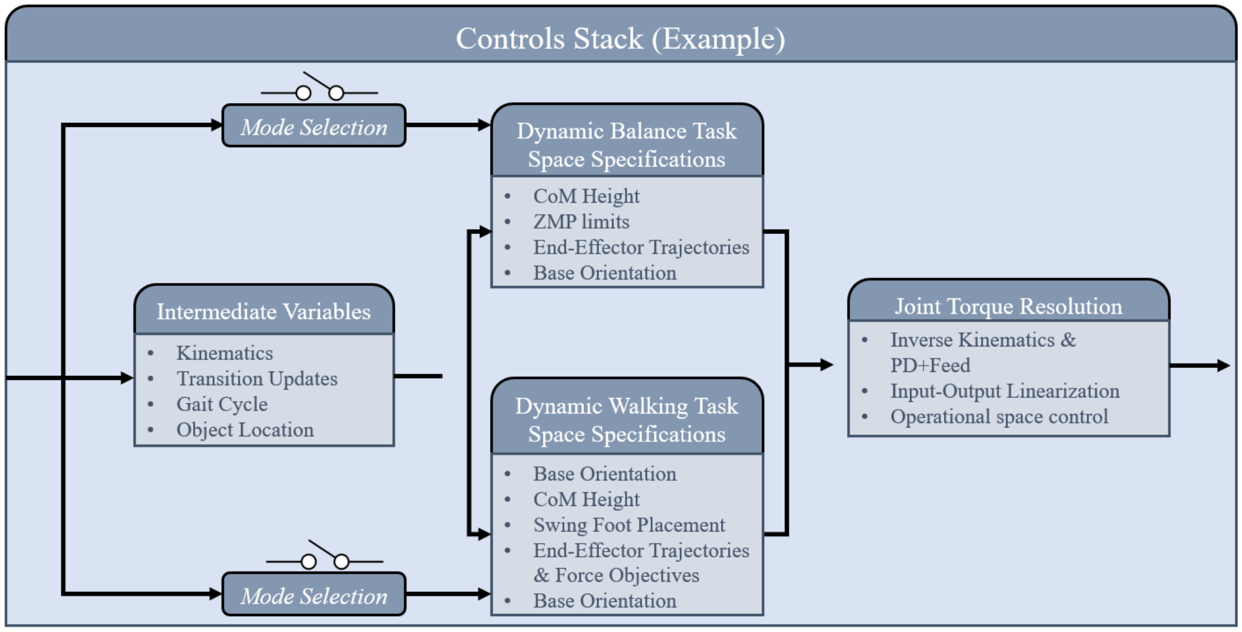


Figure 4.2: Visualizing the Control Stack process, illustrating the progression of state estimation, references, and commands through diverse algorithms to calculate intended motor velocities and torques.

The subsequent sections exclusively concentrate on the walking mode and the corresponding task maps that could materialize. For more insights into other modes or facets of the locomotion system, please refer to Appendix B.

4.2 Digit Robot Model

The kinematic tree of the Digit robot is described in this section for reference for the remainder of this document. The generalized coordinates of the Digit robot are

$$q = \begin{bmatrix} q_{\text{base}} \\ q_{\text{body}} \end{bmatrix}, \tag{4.1}$$

where

$$q_{\text{base}} = \begin{bmatrix} q_{\text{pos,x}} & q_{\text{pos,y}} & q_{\text{pos,z}} & q_{\text{Euler,yaw}} & q_{\text{Euler,pitch}} & q_{\text{Euler,roll}} \end{bmatrix}^T \tag{4.2}$$

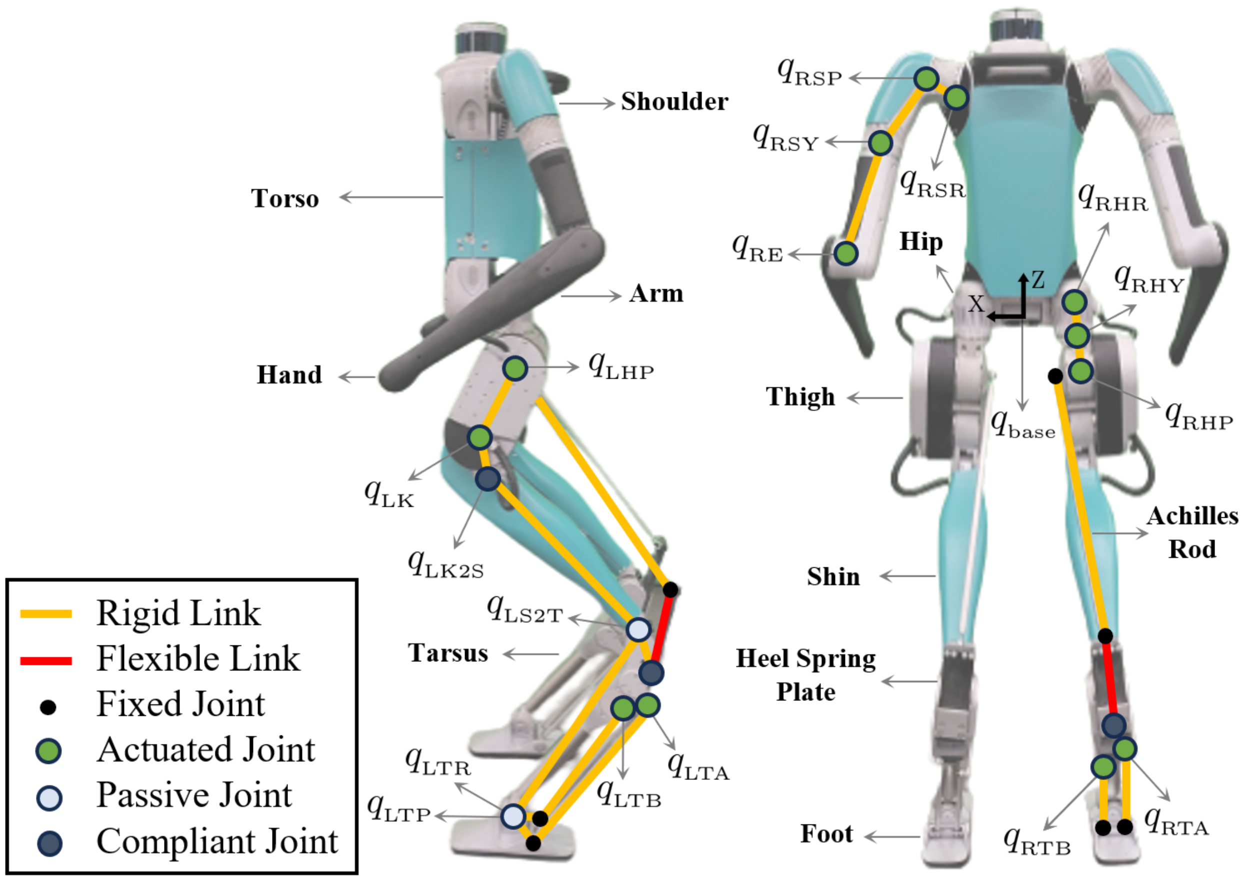


Figure 4.3: The Digit robot is shown from both the side (left) and the back (right). Its kinematic tree is composed of open and closed chains with rigid and flexible links and joints that are actuated, passive, and compliant. The base link of the robot is located at the center of its pelvis.

are the floating base coordinates representing the pose of the base link with respect to a fixed inertial frame and

$$q_{\text{body}} = \begin{bmatrix} q^T_{\text{L,leg}} & q^T_{\text{L,arm}} & q^T_{\text{R,leg}} & q^T_{\text{R,arm}} \end{bmatrix}^T. \tag{4.3}$$

are the body coordinates for each arm and leg. The leg coordinates are defined as

$$q_{\text{i,leg}} = \begin{bmatrix} q_{\text{i,HipRoll}} \\ q_{\text{i,HipYaw}} \\ q_{\text{i,HipPitch}} \\ q_{\text{i,Knee}} \\ q_{\text{i,KneeToShin}} \\ q_{\text{i,ShinToTarsus}} \\ q_{\text{i,ToePitch}} \\ q_{\text{i,ToeRoll}} \end{bmatrix} = \begin{bmatrix} q_{\text{iHR}} \\ q_{\text{iHY}} \\ q_{\text{iHP}} \\ q_{\text{iK}} \\ q_{\text{iK2S}} \\ q_{\text{iS2T}} \\ q_{\text{iTP}} \\ q_{\text{iTR}} \end{bmatrix}, \tag{4.4}$$

where $i \in \{\text{L}, \text{R}\}$ represent the left and right leg, and the arm coordinates are defined as

$$q_{\text{i,arm}} = \begin{bmatrix} q_{\text{i,ShoulderRoll}} \\ q_{\text{i,ShoulderPitch}} \\ q_{\text{i,ShoulderYaw}} \\ q_{\text{i,Elbow}} \end{bmatrix} = \begin{bmatrix} q_{\text{iSR}} \\ q_{\text{iSP}} \\ q_{\text{iSY}} \\ q_{\text{iE}} \end{bmatrix}, \tag{4.5}$$

where $i \in \{\text{left}, \text{right}\}$ represent the left and right arm.

4.3 ALIP-MPC Modifications and Improvements

The walking algorithm employed on Digit extends the method previously described in Chapter 3. Several improvements have been made to the optimization formulation to improve performance and efficiency. For the basic formulation of the MPC

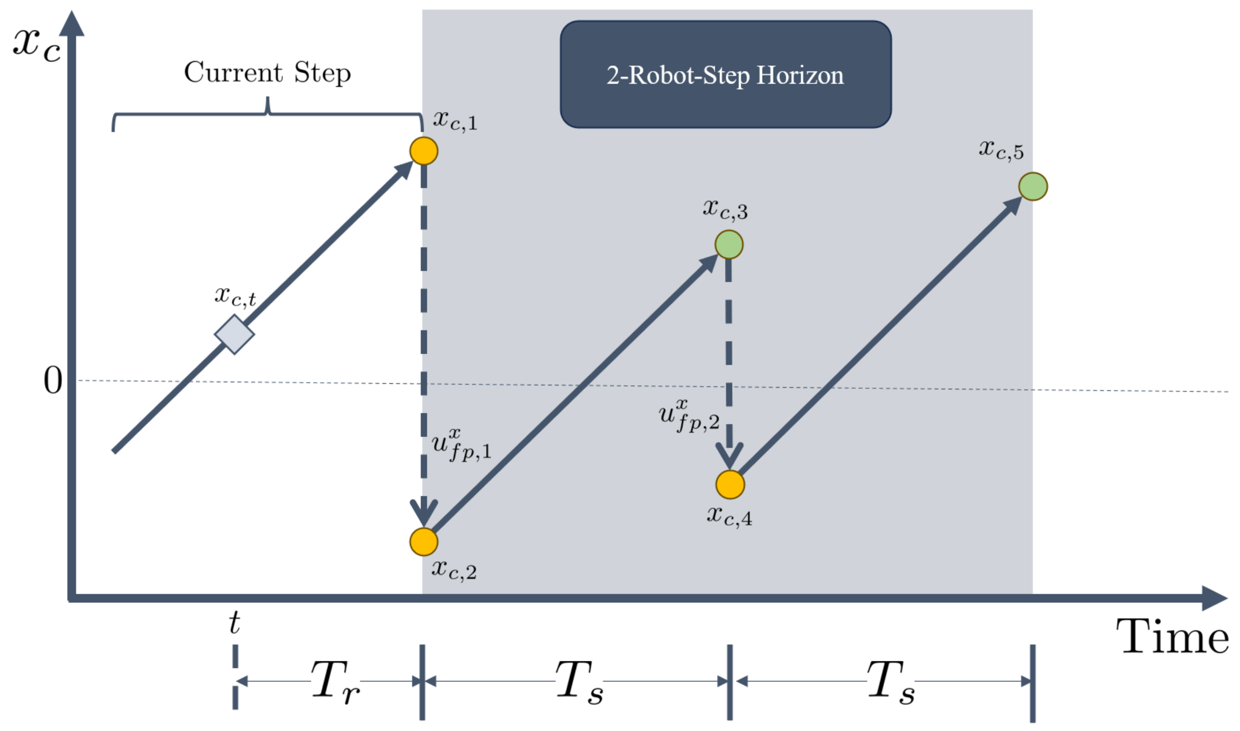


Figure 4.4: A visualization of variables used within the MPC formulation. In this example, the trajectory of the sagittal CoM position of the ALIP model ($x_{\mathrm{c,i}}$) is plotted with optimization variables and parameters for a fixed 2-robot-step horizon. Each step period has a fixed time of T_s. The colored circles are the CoM decision variables ($x_{\mathrm{c,i}}$), and the dashed arrows are the foot placement decision variables ($u_{\mathrm{fp,i}}$). The green circles represent locations where a state cost is computed. The initial condition of the optimization is represented by $x_{c,1}$. It must be pre-computed using the current estimate of the CoM position and time remaining in the current step.

problem, refer to Section 3.3. A simplified yet helpful visualization of the MPC problem is shown in Figure 4.4.

4.3.1 Cost Function

In the previous MPC formulation described in Eq. (3.16), the running cost is only a function of the state. However, this is lacking consistency with respect to the terminal cost derivation. Therefore, we add a term to the running cost such that the magnitude of each control input (foot placement) is penalized. The penalty matrix is then used to derive the terminal cost. The reformulated cost function is

$$\min_{\mathbf{U}_{\text{fp}}} J = \sum_{i=0}^{N_{\delta t}N_s-1} \mathbf{x}_{e,i}^T Q \mathbf{x}_{e,i} + \sum_{j=0}^{N_s-1} \mathbf{u}_{\text{fp,j}}^T R \mathbf{u}_{\text{fp,j}} + \mathbf{x}_{e,N_{\delta t}N_s}^T Q_{\text{term}} \mathbf{x}_{e,N_{\delta t}N_s} \tag{4.6}$$

where R is a positive definite matrix of appropriate size, $u_{\text{fp,j}}$ are foot placement decision variables, and the terminal cost has been renamed as Q_{term}.

The terminal cost is computed as the DARE solution for a N_s periodic ALIP system. The setup and solution for an example 2-robot-step periodic system are described below.

First, we assume the nominal discrete dynamics of the ALIP model are

$$\mathrm{x}_{\mathrm{k+1}} = A\mathrm{x}_{\mathrm{k}}, \tag{4.7}$$

where $k, k+1$ are the current and next state indices, A is the state transition matrix, and the impact dynamics are

$$\mathrm{x}^+ = \mathrm{x}^- + B\mathrm{u}_{\text{fp}}, \tag{4.8}$$

where $+, -$ represent the states pre- and post-impact, B is the impact map, and u_{fp} is the foot placement control input. Next, for a 1-robot-step periodic ALIP system, where each state x_{1p} is the ALIP state value before impact, the discrete dynamics can be written as

$$\begin{aligned} \mathrm{x}_{\text{1p,k+1}} &= A(\mathrm{x}_{\text{1p,k}} + B\mathrm{u}_{\text{fp}}) \\ &= A_{\text{1p}}\mathrm{x}_{\text{1p,k}} + B_{\text{1p}}\mathrm{u}_{\text{fp}}, \end{aligned} \tag{4.9}$$

where $A_{\text{1p}} = A$ is the state matrix, $B_{\text{1p}} = AB$ is the input matrix for the 1-robot-step periodic system, and $\mathrm{u}_{\text{fp}} \in \mathbb{R}$.

Similarly, the 2-robot-step periodic dynamics are written as

$$\mathrm{x}_{\mathrm{2p,k+1}} = A_{\mathrm{2p}}\mathrm{x}_{\mathrm{2p,k}} + B_{\mathrm{2p}}\mathrm{u}_{\mathrm{fp}}, \tag{4.10}$$

where $A_{\mathrm{2p}} = A^2$, $B_{\mathrm{2p}} = [A^2B, \quad AB]$, and $\mathrm{u}_{\mathrm{fp}} \in \mathbb{R}^2$.

With this information, we can now substitute the desired state, input, and cost matrices into the DARE,

$$Q_{\mathrm{term}} = A_{\mathrm{ip}}^T Q_{\mathrm{term}} A_{\mathrm{ip}} - (A_{\mathrm{ip}}^T Q_{\mathrm{term}} B_{\mathrm{ip}})(R + B_{\mathrm{ip}}^T Q_{\mathrm{term}} B_{\mathrm{ip}})^{-1}(B_{\mathrm{ip}}^T Q_{\mathrm{term}} A_{\mathrm{ip}}) + Q, \tag{4.11}$$

where ip is the matrix identifier for the corresponding robot-step horizon. A method for efficiently solving (4.11) can be found in [86].

4.3.2 Impact Map

The impact map matrix from (3.12) assumes that the CoM height location is tracked perfectly at impact. With a robot like Cassie, whose mass is centrally located at the pelvis, this assumption is fair at low speeds ($<$ 2 m/s). However, this assumption breaks down at faster speeds, and a more complex linear matrix should be used for improved results as detailed in [41]. The updated impact map still uses the conservation of angular momentum but now assumes that the vertical CoM velocity is non-negligible. Given the current gains and torque control strategy, Digit's vertical CoM velocity is non-negligible at much lower speeds ($<$ 0.4 m/s). The improved impact map from [41] must also be updated to include locomotion across sloped terrain. Taking these factors into consideration, the updated impact map is given as

$$B = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ k_x v^y_{\text{CoM,eos}} & k_y v^y_{\text{CoM,eos}} - v^z_{\text{CoM,eos}} \\ v^z_{\text{CoM,eos}} - k_x v^x_{\text{CoM,eos}} & -k_y v^x_{\text{CoM,eos}} \end{bmatrix}, \tag{4.12}$$

where k_x, k_y are the slope of about the sagittal and lateral components of the robot and $v^x_{\text{CoM,eos}}, v^y_{\text{CoM,eos}}, v^z_{\text{CoM,eos}}$ are the estimated center of mass velocities at the end of each predicted step (including the current step). This map is used for all implementations of the ALIP-MPC walking algorithm on Digit.

Note: If the tracking performance of the stance knee was improved, it is possible that this updated impact model would be unnecessary for low walking speed.

4.3.3 Slack constraints and Rate Limits

ALIP-MPC feasibility issues may arise When Digit is walking at speeds near the maximum and minimum commanded velocity targets (especially in the sagittal direction). The main reason is the hard constraint imposed on each step's maximum allowable foot placement. As the robot increases speed with smaller robot-step horizons, the predicted CoM position may accelerate such that the maximum foot placement input cannot slow the robot down. We have found that adding additional slack variables to the foot placement constraints, such as

$$u^{\text{x}}_{\text{fp}} \leq u^{\text{x}}_{\text{fp,max}} + \lambda_{\text{fp,x}}, \tag{4.13}$$

can eliminate this issue in practice. An additional term $S_{\text{fp,x}}\lambda^2_{\text{fp,x}}$ is then added to the cost function in (3.16) and (4.6), where $S_{\text{fp,x}} > 0$. Additional constraints and cost penalties can be added to the MPC formulation at the user's preference. We have also considered adding rate limiters to the foot placement inputs where the change in magnitude of subsequent steps is constrained. However, this could reduce the robot's success of push recovery and was therefore not used. A rate limiter constraint could be a good addition to the controller if a separate push recovery foot placement modifier were added after the ALIP-MPC solution was retrieved.

4.3.4 Additional Considerations

Since we assume the optimization size is fixed, a separate solver must be created for each robot-step horizon. A separate MPC formulation is also needed for different stance leg conditions. The lateral foot placement constraints on $u^{\mathrm{y}}_{\mathrm{fp}}$ cannot be expressed for arbitrary initial stance leg conditions without adding discrete decision variables. Therefore, a separate MPC formulation is created for each stance leg for efficiency. The formulations are equivalent except with respect to the lateral foot placement constraints. As an example, for an even robot-step horizon with the current step in left stance, the lateral constraints would be

$$-u^{\mathrm{y}}_{\mathrm{fp,max}} \leq u^{\mathrm{y}}_{\mathrm{fp,n}} \leq -u^{\mathrm{y}}_{\mathrm{fp,min}} u^{\mathrm{y}}_{\mathrm{fp,min}} \leq u^{\mathrm{y}}_{\mathrm{fp,n+1}} \leq u^{\mathrm{y}}_{\mathrm{fp,max}},$$

where N_s is the robot-step horizon, $u^{\mathrm{y}}_{\mathrm{fp,max}}$ is the magnitude of the largest lateral step and $u^{\mathrm{y}}_{\mathrm{fp,min}}$ is the magnitude of the smallest lateral step, and n is every odd natural number less than the robot-step horizon (i.e. $n = 1 : 2 : N_s$). Slack variables are omitted from Eq. (4.14) for conciseness but should be added for feasibility robustness.

Note: Mixed integer optimization solvers could be used to remove the need for separate solvers depending on the stance leg; however, these problems typically take longer to evaluate than standard QPs and are therefore not considered.

4.4 Terrain-Aware Walking *sans* Perception

A similar virtual constraint and inverse kinematics tracking method, as described in Chapter 3, is utilized for the walking controller on the Digit robot. Apart from the virtual constraints defined by Eq. (A.5), additional tasks are incorporated for the 8 arm joints and 2 toe roll joints. A pseudo-inverse inverse kinematics scheme is applied to compute the desired arm joint values based on the desired Cartesian position of each end-effector knob. Each toe has two linear actuators that modify their relative pitch and roll. A data-based quadratic nonlinear regression algorithm maps desired orientation values to motor position values for integration into the virtual constraints. For simplicity, A Proportional-Derivative (PD) controller is used to track all desired joints with feedforward torque added to selected actuators.

4.4.1 Preliminary Simulation & Hardware Results

All results are achieved using the updated ALIP-MPC algorithm discussed in Section 4.3. Figure 4.5 illustrates (in simulation) a comparison between the ALIP-MPC strategy for including terrain information versus assuming the ground is flat. Figures 4.6 and 4.7 showcase the ALIP-MPC performance on actual hardware for a variety of terrain conditions.[1] A table of relevant ALIP-MPC walking parameters and gains are listed in Table 4.1, Table 4.2 and Table 4.3, respectively.

4.5 Mapping and Perception

The purpose of this section is to describe the sensors and methods used to extract terrain information necessary for implementing the terrain-aware foot placement method first described in Section 3.

The terrain-aware walking techniques introduced in Chapters 3 and 4 necessitate

[1] Open-source repository: https://github.com/UMich-BipedLab/digit_locomotion_controller

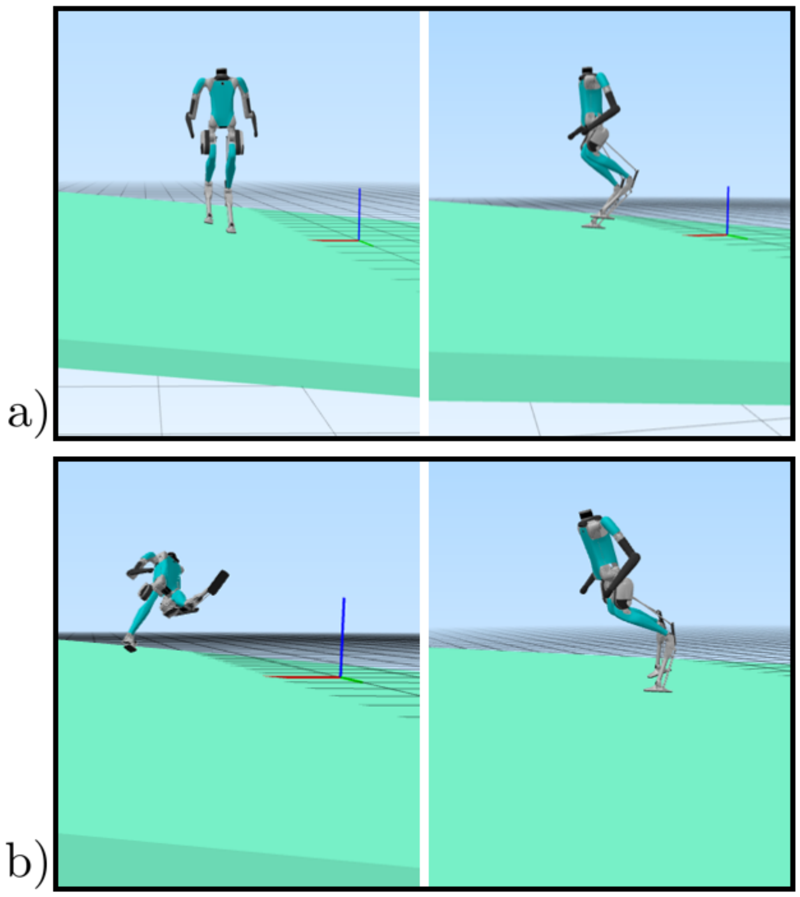


Figure 4.5: Comparison of ALIP-MPC performance for a) known terrain parameters versus b) assumption that the ground is flat. The images on the left column show Digit walking laterally up a 5° inclined plane. The images in the right column show Digit walking forward up the same plane. In both comparisons, the flat ground assumption causes the robot to become unstable and fall due to early impact. The ALIP-MPC (terrain-aware) method remains stable for the same commanded reference velocities. The simulations were performed on the Agility Robotics custom simulator that uses the Mujoco physics engine.

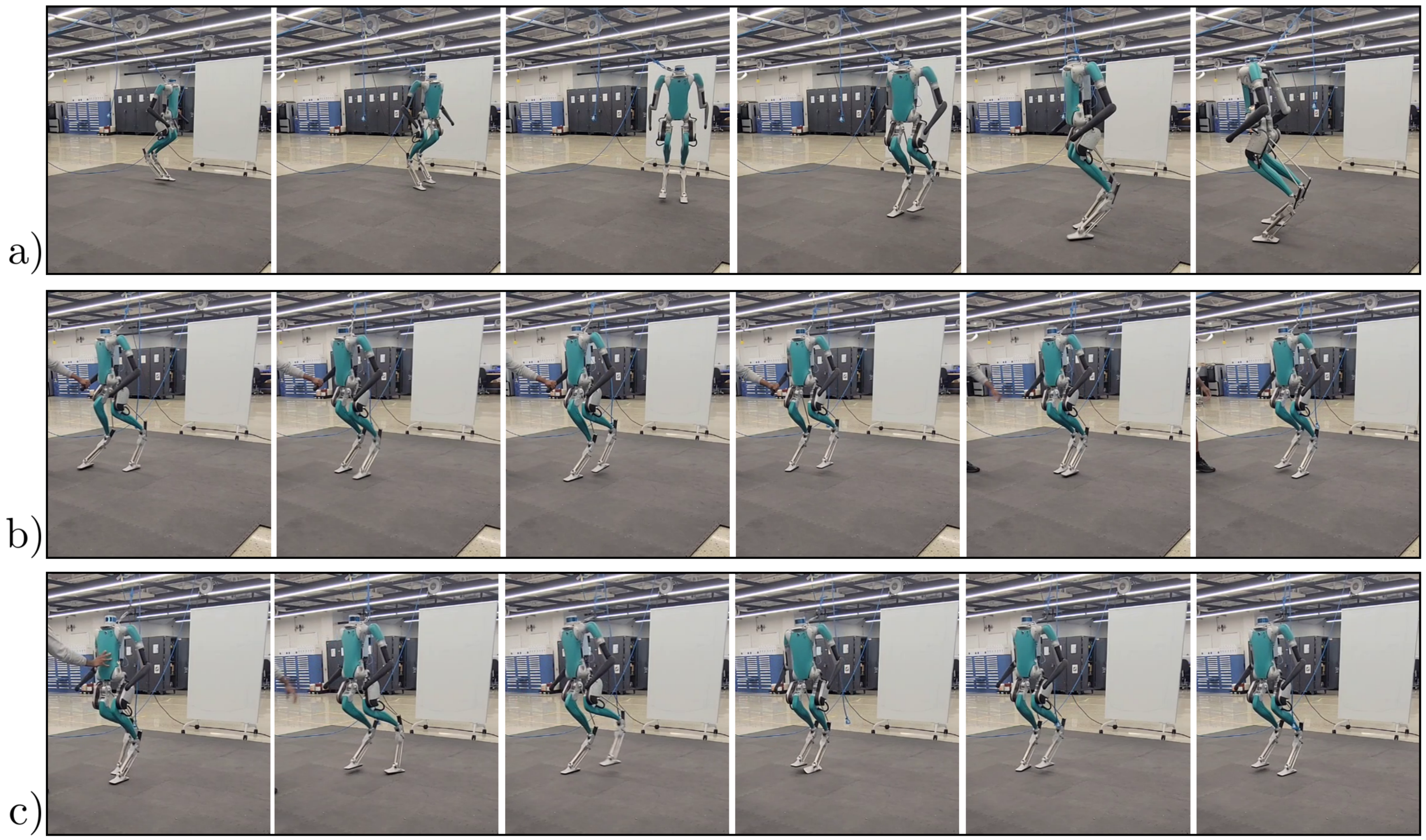


Figure 4.6: ALIP-MPC walking algorithm on flat ground. The experiment details are as follows: a) combined sagittal and turn walking in a circle, b) persistent time-varying disturbance rejection at the right-hand end-effector, and c) abrupt sagittal push recovery.

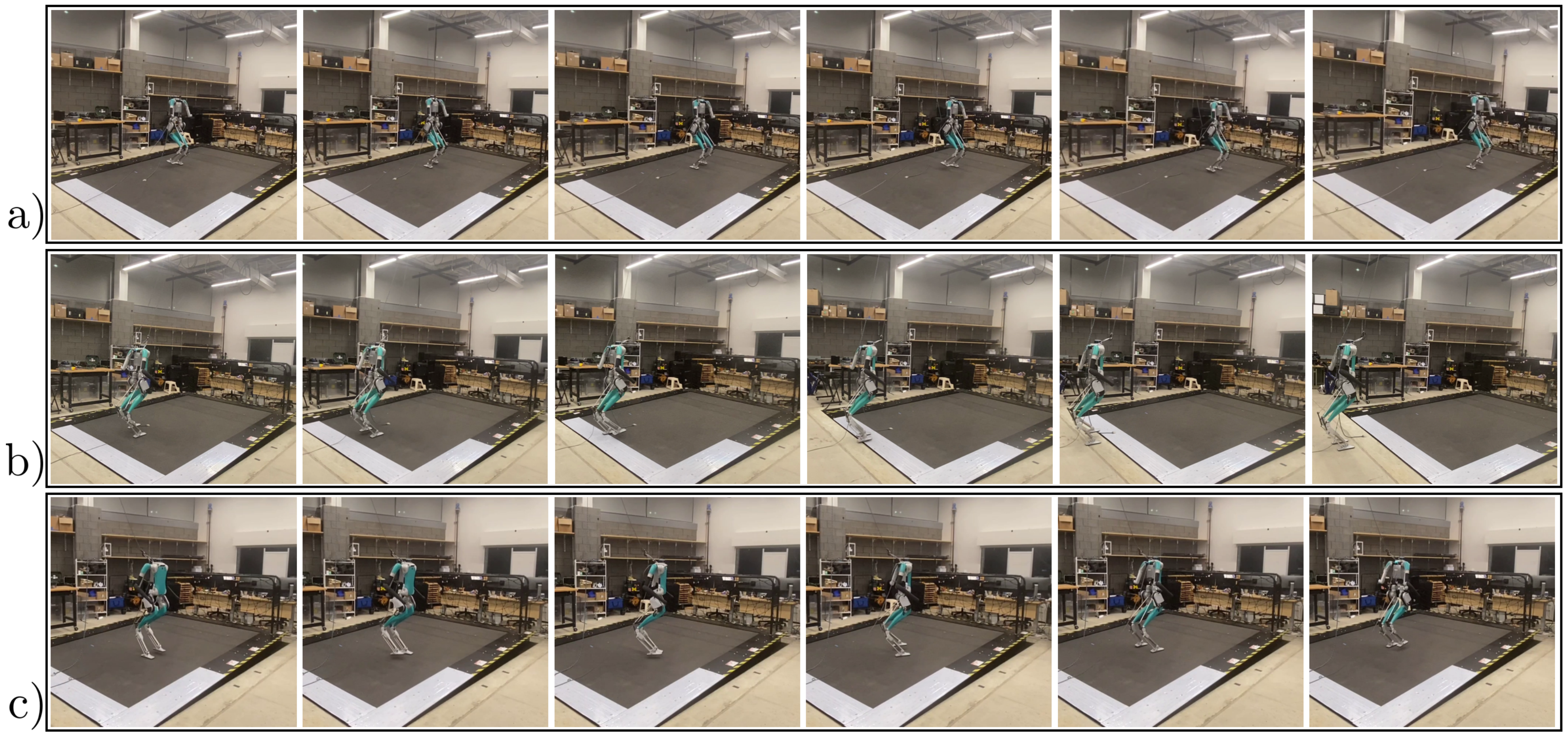


Figure 4.7: ALIP-MPC walking algorithm on 5° inclined treadmill. The experiment details are as follows: a) transverse sagittal backward walking across slope, b) sagittal backward walking transition from incline to flat ground, and c) turn in place on slope. The operator modified the reference velocity and terrain parameters in real-time.

Table 4.1: Nominal Walking Controller Constants and Parameters.

Walking Mode Parameters	**Values**
Mass (m)	46.2104 kg
Step Period (T_s)	0.38 s
CoM Height (z_H)	0.95 m
Step Width (W)	0.31 m
Nominal Friction Coefficient (μ)	0.6
Foot Clearance at Mid-step	0.1 m
Maximum Magnitude of Commanded Sagittal velocity	0.5 m/s
Maximum Magnitude of Commanded Lateral Velocity	0.5 m/s
Maximum Magnitude of Commanded Turn Rate	0.5 m/s

Table 4.2: Nominal ALIP-MPC Parameter Table.

ALIP-MPC Parameters	**Values**
Robot-step Horizon (N_s)	4
Maximum Center of Mass Sagittal Position ($x_{\text{CoM}}^{\text{max}}$)	0.15 m
Minimum Center of Mass Sagittal Position ($x_{\text{CoM}}^{\text{min}}$)	-0.15 m
Maximum Sagittal Foot Placement ($u_{\text{fp,x}}^{\text{max}}$)	0.25 m
Minimum Sagittal Foot Placement ($u_{\text{fp,x}}^{\text{min}}$)	-0.25 m
Magnitude of Maximum Lateral Foot Placement ($\lvert u_{\text{fp,y}}^{\text{max}}\rvert$)	0.5 m
Magnitude of Minimum Lateral Foot Placement ($\lvert u_{\text{fp,y}}^{\text{min}}\rvert$)	0.1 m
State Penalty Matrix (Q)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$
Foot Placement Penalty Matrix (R)	$\begin{bmatrix} 1 & 0 \\ 0 & 10^5 \end{bmatrix}$
Sagittal Center of Mass position Slack Variable Penalty ($S_{x_{\text{CoM}}}$)	10^8
Slip Constraint Slack Variable Penalty (S_{slip})	10^6
Sagittal Foot Placement Slack Variable Penalty ($S_{\text{fp,x}}$)	10^8
Lateral Foot Placement Slack Variable Penalty ($S_{\text{fp,y}}$)	10^6

Table 4.3: Nominal Walking Controller Gains. K_p denotes position error gains and K_d denotes damping error gains. Additional subscript notation corresponds to the definitions found in Section 4.2. The T subscript refers to the gain of both toe motors on a given leg. The gains for the arms are applied equally, regardless of the stance leg. The superscripts st and sw refer to stance and swing leg, respectively. All damping gains are equivalent for symmetric left and right joints.

Walking Mode Gains	**Values**	**Walking Mode Gains**	**Values**
-	-	Damping Multiplier (ρ)	0.9
$K^\text{sw}_\text{p,HR}$	800 N-m/rad	$K_\text{d,HR}$	$\rho \cdot 66.849$ N-m-s/rad
$K^\text{sw}_\text{p,HY}$	400 N-m/rad	$K_\text{d,HY}$	$\rho \cdot 26.1129$ N-m-s/rad
$K^\text{sw}_\text{p,HP}$	800 N-m/rad	$K_\text{d,HP}$	$\rho \cdot 38.05$ N-m-s/rad
$K^\text{sw}_\text{p,K}$	500 N-m/rad	$K_\text{d,K}$	$\rho \cdot 38.05$ N-m-s/rad
$K^\text{sw}_\text{p,T}$	500 N-m/rad	$K_\text{d,T}$	$\rho \cdot 28.5532$ N-m-s/rad
$K^\text{st}_\text{p,HR}$	800 N-m/rad	-	-
$K^\text{st}_\text{p,HY}$	400 N-m/rad	-	-
$K^\text{st}_\text{p,HP}$	400 N-m/rad	-	-
$K^\text{st}_\text{p,K}$	500 N-m/rad	-	-
$K^\text{st}_\text{p,T}$	0 N-m/rad	-	-
$K_\text{p,SR}$	100 N-m/rad	$K_\text{d,SR}$	$\rho \cdot 66.849$ N-m-s/rad
$K_\text{p,SP}$	100 N-m/rad	$K_\text{d,SP}$	$\rho \cdot 66.849$ N-m-s/rad
$K_\text{p,SY}$	100 N-m/rad	$K_\text{d,SP}$	$\rho \cdot 26.1129$ N-m-s/rad
$K_\text{p,E}$	100 N-m/rad	$K_\text{d,SP}$	$\rho \cdot 66.849$ N-m-s/rad

real-time estimations of both the friction coefficient and the surface normal of the immediate terrain surrounding the robot. A plane segmentation algorithm can estimate the surface normal, partitioning ground areas with relatively uniform slopes. On the other hand, the friction coefficient can be inferred through assessments of categorized ground textures and evaluations of perceived ground roughness. These computations mandate the constant acquisition and updating of a registered robot-centric point cloud, from which a filtered elevation map is generated [60]. This very need for real-time data processing underscores the rationale behind selecting a humanoid robot equipped with integrated perception sensors for this study.

4.5.1 Sensors

The Digit robot has a Velodyne™ VLP-16 LiDAR a Tis camera and five Intel® RealSense™ RGB-D sensors, offering a combined depth and color data stream. For the research edition of the Digit robot (digit-v3), direct access to the perception stream is not available. However, the robot does provide a TCP/IP communication protocol, enabling access to perception data while allowing customization of some settings. Regrettably, the point cloud registration process, involving the rigid-body transformation between the point cloud and the world frame, suffers considerable delays. Consequently, the onboard sensing is unsuitable for experimental use, as illustrated in Figure 4.8. Collaborative efforts with Agility Robotics aimed to facilitate performance on this platform. However, it is regrettable that as of the 2022.02.22a release (from which all presented results were derived), the synchronization between point clouds and the necessary transformations (between depth camera coordinate frame and world frame) remained unresolved.

To address this challenge, we implemented a solution involving the integration of an external Microsoft® Azure Kinect RGB-D sensor[2]. This sensor was attached to

[2]We extend our gratitude to Hao Chen and the ROAHM Lab at the University of Michigan for their contribution to the idea and design for the mount.

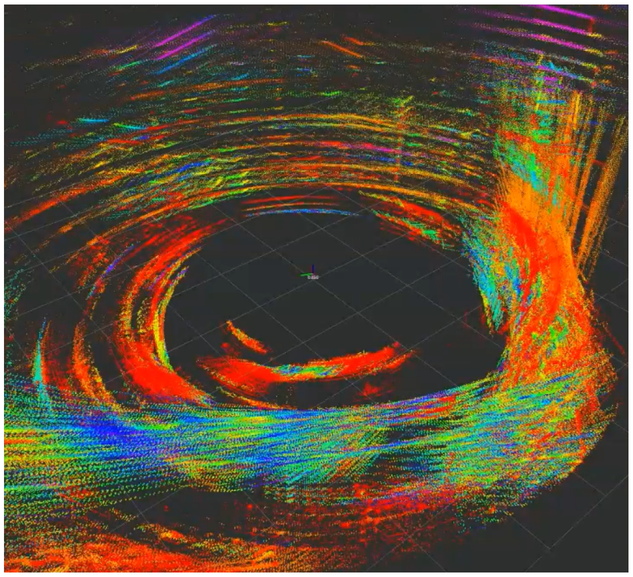


Figure 4.8: Exemplifying Subpar Point Cloud Registration with the Native LiDAR on the Digit-v3 Robot (release 2022.02.22a). In this demonstration, the robot operates in Stand mode while the desired torso orientation undergoes modification. As the robot rotates, an updated point cloud and corresponding rigid-body transformation (from LiDAR to the world frame) generate a time-lapse visualization of points within RViz [10]. In an optimal scenario, the geometric characteristics of the environment could be accurately inferred. However, a discernible blur in the visualization indicates a delay in the rigid-body transformation, impacting the fidelity of the registration process.

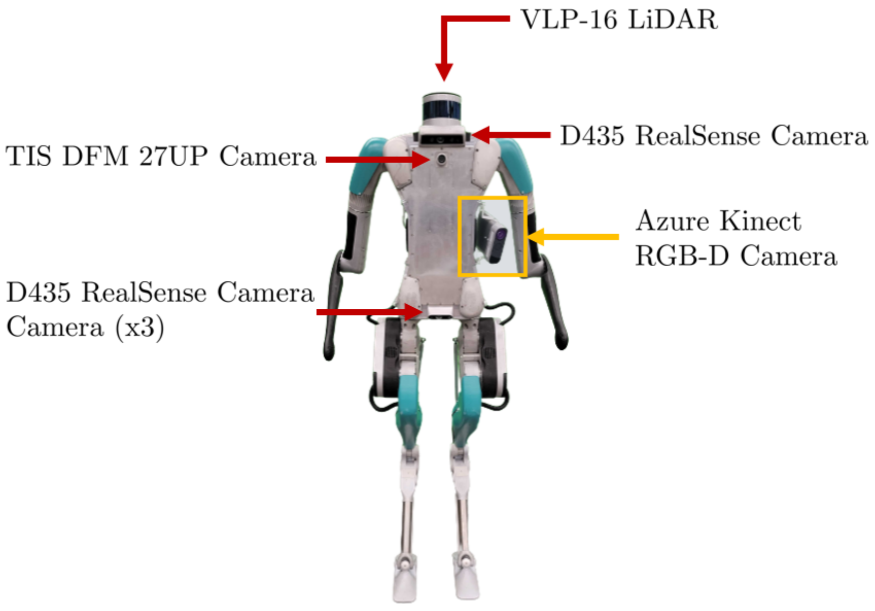


Figure 4.9: Schematic depicting the various onboard perception sensors of Digit. The native LiDAR and cameras exhibit subpar depth point registration, rendering them unsuitable for real-time use. Consequently, an external Azure Kinect camera is mounted and calibrated, with all mapping and estimation algorithms relying solely on data from this sensor.

the Digit robot and connected to the payload computer. Figure 4.9 illustrates the precise placement of all perception-related sensors.

With the ability to deliver a dense point cloud at a consistent 30 frames per second rate, the sensor's performance is satisfactory for experiments. We employed the Kalibr [87] calibration package for both the intrinsic calibration parameters of the camera and the extrinsic calibration alignment between the robot's base frame and the depth frame of the sensor. In Figure 4.10, we show what a *good* point cloud registration looks like using the Azure Kinect camera. A one-minute bag of depth point cloud data is visualized with respect to the inertial world frame of the robot as it moves around in the environment.

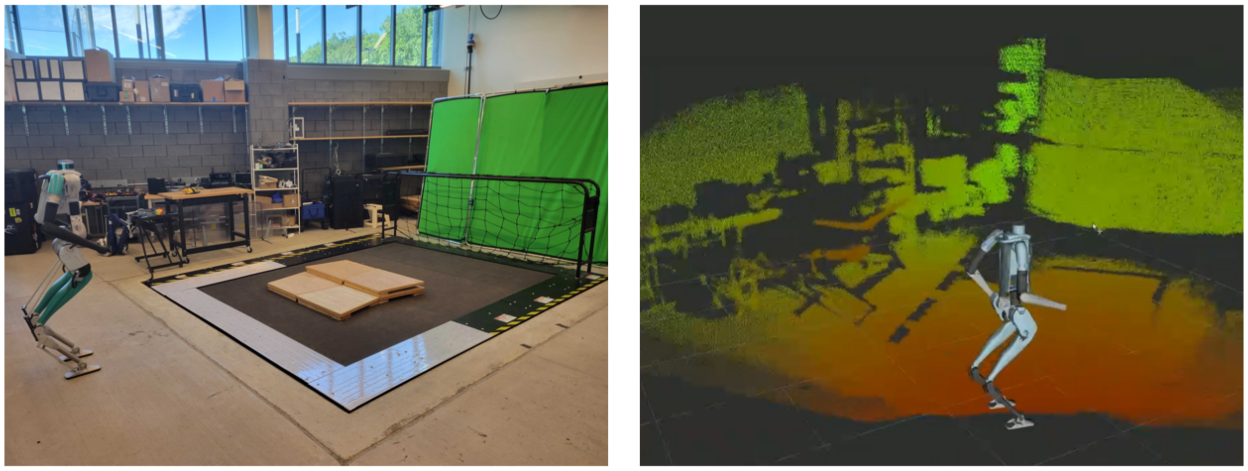


Figure 4.10: This figure shows a one-minute data capture of depth point cloud data from Digit as it moved within a lab environment. The left image illustrates the setup. On the right, Rviz visualizes the robot model of Digit alongside the depth points. The time-synchronized registration between the depth camera and the world inertial frame is accurate enough that geometric features, such as boxes, tables, and shelves, are easily discernible.

4.5.2 Elevation Mapping, Plane Segmentation, & Traversability

We compute elevation map, plane segmentation, and traversability estimates at 5 Hz utilizing the ETH `elevation_mapping_cupy` package [11]. This computational task is performed on an NVIDIA® Jetson Xavier platform. Figure 4.11 details some hardware results. As a last step, we create a final node to post-process the perception information and publish it to the payload computer for use in the proposed perception-locomotion algorithm detailed in Section 4.6. Each message contains an array of all segmented planes with surface normal and traversability data.

4.6 Proposed Method for Terrain-Aware Walking with Perception

Overcoming the challenge of real-time computation for humanoid foot placements, informed by environmental data, has remained a persistent hurdle in the realm of robot locomotion [88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98]. A prevailing trend a)

b)

c)

d)

e)

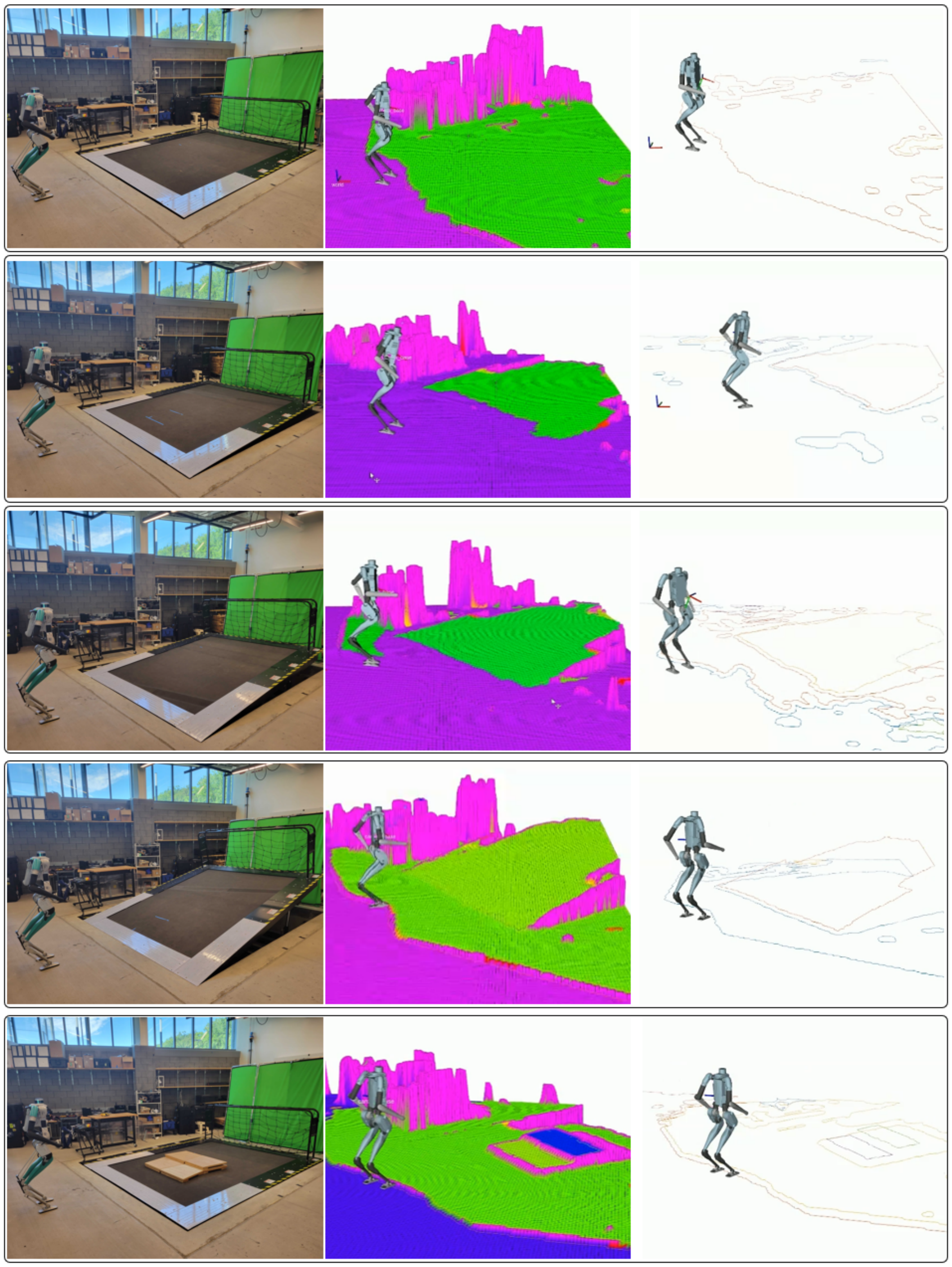


Figure 4.11: On the left of each test, we see the environment setup; the middle depicts plane segmentation, and the right showcases plane boundary results from the elevation mapping package referenced in [11]. A treadmill is employed to simulate varied inclines for the robot to estimate. Displayed scenarios include a) 0° inclination, b) 5° inclination, c) 10° inclination, d) 20° inclination, and e) 0° inclined steps.

across a significant portion of these methods is their protracted computation times, which can span from a few seconds to potentially longer [92], primarily attributed to the inherent non-convex nature of the optimization problems they address. To enable walking algorithms to seamlessly adapt to diverse terrains while remaining synchronized with the constraints of ongoing gait cycles, the key objective is to develop solutions characterized by rapid yet reliable computation of foot placement trajectories.

Our proposed approach, outlined in Figure 4.12, leverages terrain information (Section 4.5) to address the ALIP-MPC problem through a two-stage process. Initially, a solution is generated based on desired CoM velocity target values, projecting the estimated future footstep locations. Subsequently, this solution is overlaid onto the terrain map, facilitating updates to the terrain parameters associated with each footstep. Ultimately, the ALIP-MPC problem is resolved, leading to an updated desired foot placement. Unlike alternative methods that achieve rapid foot placement computation through stochastic processes [93], our approach offers an advantageous alternative by avoiding potential suboptimal outcomes.

4.7 Future Steps

Ongoing evaluations of this method are underway. We plan to conduct diverse hardware tests using the perception-locomotion approach outlined in this chapter as the next steps. I have listed the *tentative* author list and title of the corresponding article for future reference.

- **Grant Gibson**, Elizabeth A. Olson, Jessy W. Grizzle. Terrain-Aware Navigation for Bipedal Humanoid Robots.

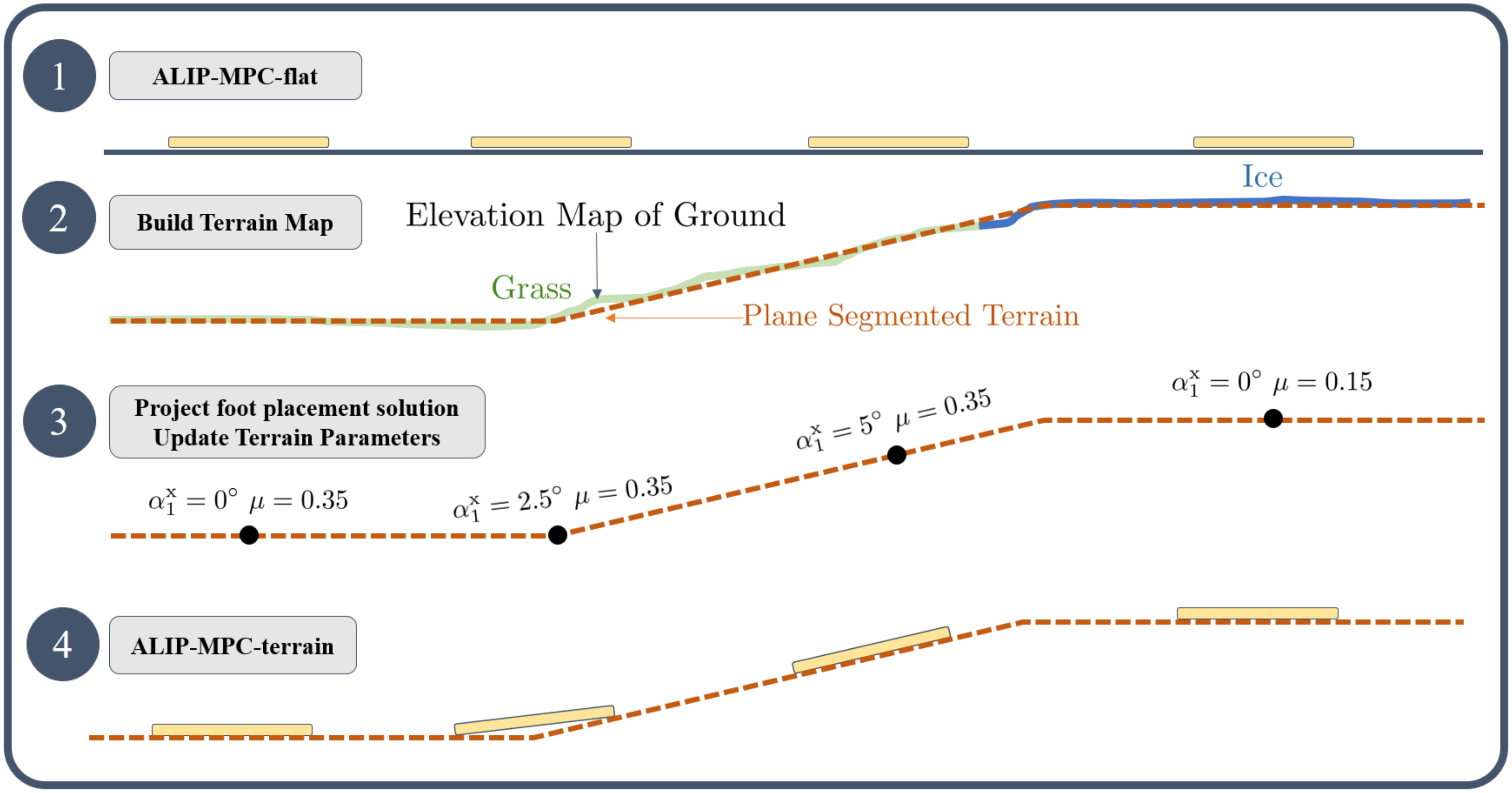

1
ALIP-MPC-flat
2
Build Terrain Map
Elevation Map of Ground
Grass
Ice
Plane Segmented Terrain
3
Project foot placement solution
Update Terrain Parameters
$\alpha_1^x = 0^\circ \ \mu = 0.35$
$\alpha_1^x = 2.5^\circ \ \mu = 0.35$
$\alpha_1^x = 5^\circ \ \mu = 0.35$
$\alpha_1^x = 0^\circ \ \mu = 0.15$
4
ALIP-MPC-terrain

Figure 4.12: Visualization of the Perception-Integrated Terrain-Aware Walking Strategy. Within this framework, the ALIP-MPC problem is approached twice. The ultimate solution, denoted as ALIP-MPC-terrain, mirrors the methodology elucidated in Section 4.4. However, instead of relying on user-defined terrain parameters, the ALIP-MPC-flat solution is harnessed to refine slope and friction information dynamically. The formulation of the ALIP-MPC-flat problem as an unconstrained quadratic program lends itself to swift computation via KKT methods and efficient linear solvers, as detailed by Borrelli et al. [12].

Chapter 5

Exploring Kinodynamic Fabrics for Reactive Whole-Body Control of Bipedal Humanoid Robots

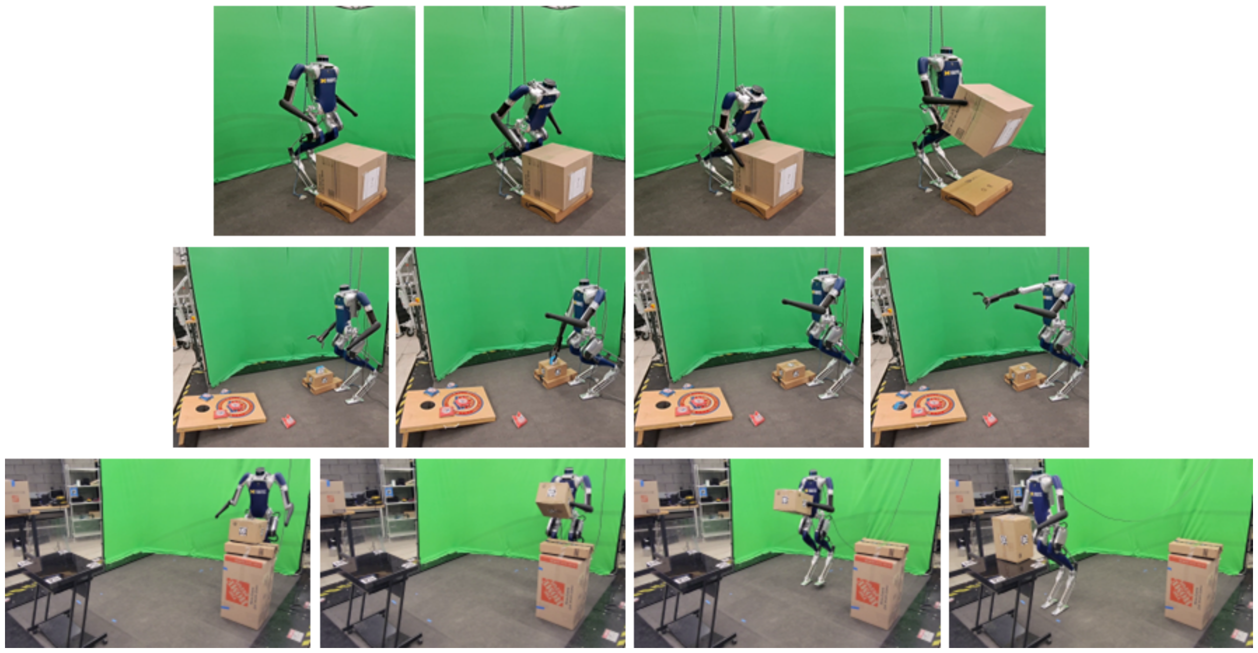


Figure 5.1: A Digit robot executes a variety of whole-body motions using the Kinodynamic Fabrics Framework. (Top) Digit lifts a bulky box of non-uniform, shifting mass distribution that the framework does not model. (Middle) Digit plays cornhole. (Bottom) Digit transports a package to a desired location. Digit is designed by Agility Robotics.

5.1 Introduction

The bipedal humanoid morphology is beneficial for robots for two main reasons. The first reason is that the humanoid morphology, though not specialized in traversing a particular environment, is highly agile, versatile, and generalizes across diverse environmental structures. This feature allows humans to navigate diverse environments through swimming when in water, running when on land, climbing trees and mountains, and crawling under tight spaces [99]. The second reason is that if the goal is to deploy robots in human-occupied spaces, it is only natural that we fashion them to mimic human morphology to require the least amount of environmental restructuring to cater to the successful operation of robots in such spaces. Given these benefits, a bipedal humanoid robot capable of fast and reactive whole-body control would be primed to take on diverse physical tasks, particularly those that may be tedious or risky for humans.

Whole-body control is used for highly redundant, high degree-of-freedom floating-base robots to simultaneously achieve multiple motion behaviors by exploiting the redundancy of the robot's morphology [100]. The problem of fast, reactive whole-body control of bipedal humanoid robots is challenging and raises two main unanswered questions.

The first question is, *how should motion behaviors be expressed in a way that is robust and expressive*? Virtual model control has been used to create motion behaviors that rely on the design of virtual forces and components [5], and feedback linearization has been used to track human-based motion primitives on humanoid robots for different modes of locomotion [67]. A limiting factor for using the methods is the need for expertly designed reference motions that fit into each framework.

A popular framework for solving whole-body control problems is to express the entire controller as a constrained optimization problem that is solved in each iteration of the control loop to output joint-space commands [101, 102, 103, 104, 105, 89, 88]. In this framework, motion behaviors are expressed as hard or soft constraints in the objective function depending on the importance of the motion behavior. The drawback of this framework is that it mandates motion behaviors to have a specific form. For instance, if the chosen constrained optimization problem form is a QP, motion behaviors have to strictly be linear constraints or quadratic (if appended to the objective function). Expressing motion behaviors, such as dynamic obstacle avoidance, can be challenging as linear constraints without making numerous simplifying assumptions that make the problem brittle. Also, expressing multiple obstacle avoidance behaviors as constraints in a single constrained optimization problem is likely to render the problem insoluble due to potential conflicts in the constraints.

The second question is *how should we formulate reactive whole-body control problems in a manner that is fast to solve and scales well with increasing number of motion behaviors*? Constrained optimization problems for whole-body control are often quite slow to solve and not amenable to real-time applications. As a result, most real-time whole-body control approaches try to make approximations to the constrained optimization problem to reduce solution time. One such approach is to formulate the constrained optimization problem as a QP by linearizing the constraints of the constrained optimization problem and making the objective function quadratic [88]. Besides making the problem less expressive, QPs scale badly with increasing motion behavior constraints, resulting in slow solution times and potentially infeasible optimization problems.

Given these challenges, we propose Kinodynamic Fabrics for fast, reactive whole-body control of bipedal humanoid robots. Kinodynamic Fabrics allow for the description of primitive motion behaviors as forced spectral semi-sprays (fabrics) in their respective task spaces and employ the pullback and summation operations from differential geometry to compose all motion behaviors into a single joint-space motion policy. This motion policy generates joint-space acceleration commands, which can be integrated into velocity and position commands and tracked by position- and velocity-controlled robots or fed as input to an inverse dynamics routine alongside desired contact forces to output joint torque commands for torque-controlled robots.

The ability to express primitive motion behaviors as fabrics in their respective task spaces provides the opportunity and flexibility to express the complex geometries of smooth motion behaviors as second-order differential equations. Solving for the composed joint-space motion policy through the pullback and summation operations is very fast and allows for the computation of motion policies at kilohertz rates. These qualities make Kinodynamic Fabrics a viable framework for fast, reactive whole-body control of bipedal humanoid robots. The unique contributions of this work are as follows:

- Firstly, we propose Kinodynamic Fabrics as a framework for prioritized, fast, and reactive whole-body control of bipedal humanoids. We describe how to express primitive motion behaviors as expressive second-order differential equations and how they integrate into the Kinodynamic Fabrics framework to generate smooth and dynamically consistent robot motions.
- Secondly, we describe how to efficiently represent complex motion behaviors like bipedal locomotion and bimanual manipulation as components of the Kinodynamic Fabrics framework, how to decompose these components into primitive motion behaviors, and how to execute extended sequences of motions smoothly.
- Thirdly, we demonstrate Kinodynamic Fabrics on various bimanual manipulation, bipedal locomotion, and mobile manipulation tasks on the physical Digit bipedal humanoid robot.
- Finally, we provide an open-source Julia implementation of the Kinodynamic Fabrics framework and code to reproduce all of the experiments and demonstrations in this work.[1]

We evaluate Kinodynamic Fabrics in simulations and experiments on a wide

[1]Open-source repository with code, videos, and results: https://github.com/UMich-BipedLab/KinodynamicFabrics.jl

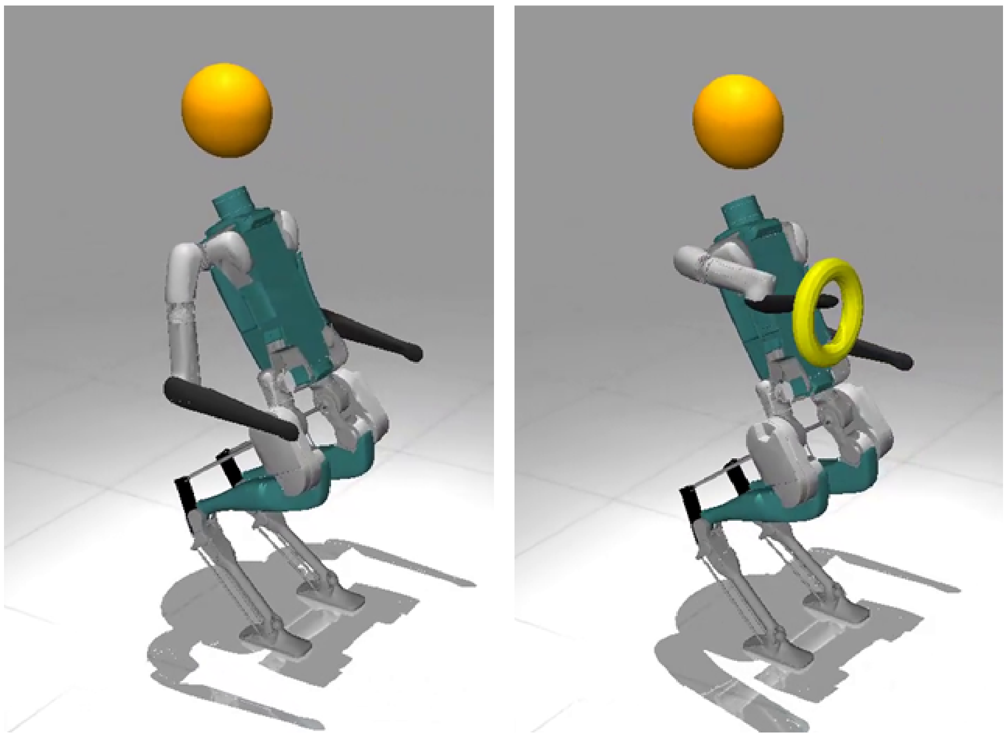


Figure 5.2: MuJoCo simulation of Digit executing motions generated by Kinodynamic Fabrics. (Left) Digit moves its whole body to dodge an incoming orange basketball. (Right) Digit dodges the incoming basketball while keeping the end of its right arm in the yellow hoop.

range of whole-body control tasks such as dynamic obstacle avoidance, bimanual whole-body manipulation, bipedal locomotion, and mobile manipulation, as shown in Figure 5.1. Kinodynamic Fabrics generate fast, reactive motions in all these tasks that allow the robot to accomplish its assigned goals. We also perform extensive benchmark comparisons of Kinodynamic Fabrics with a QP-based whole-body controller. Kinodynamic Fabrics outperforms the QP-based controller on run-time and reactivity metrics.

5.2 Related Works

5.2.1 Whole-body Control

Whole-body control is used for high DoF floating-base robots to simultaneously achieve multiple motion behaviors by exploiting the redundancy of the robot's structure [100]. Ever since whole-body control was first applied to humanoid robots in the form of Resolved Momentum Control, proposed by Kajita et al. [106], two main classes of approaches for solving whole-body control problems have been proposed; nullspace control approaches and constrained optimization-based approaches.

5.2.1.1 Nullspace Whole-Body Control

Nullspace control approaches [106, 107, 108, 109] are dominated by the use of prioritized dynamically-consistent Jacobians and their pseudo-inverses to enforce strict hierarchies between behaviors. Lower priority behaviors have Jacobians defined in the nullspace of higher priority behaviors. We adopt this prioritization in Kinodynamic Fabrics for enforcing behavior priorities. The drawback of the nullspace formulation is the inability to express inequality constraints.

5.2.1.2 Constrained Optimization-based Whole-Body Control

Constrained optimization-based whole-body control[101, 102, 103, 89, 105, 88, 110] formulates the whole-body control problem as a single constrained optimization problem or cascades of constrained optimization problems. High-priority motion behaviors are written as hard constraints, while low-priority behaviors are written as soft constraints in the objective function with weights to express their relative importance. Other constrained optimization-based approaches like Escande et al. [101] build a Hierarchical Quadratic Program to enforce strict priorities between behaviors. The main drawback of constrained optimization-based whole-body control approaches is they require non-convex formulations, which require longer computation times by solvers. This computation time grows with increasing motion behaviors, making the approach unsuitable for real-time applications. As a result, there is a need for linear approximations of behaviors (an example is the approximation as a Quadratic Program [88]) to speed up computations. These approximations invariably reduce the expressiveness of behaviors and increase the difficulty involved in designing motion behaviors. Another drawback of this class of approaches is that with an increasing number of motion behaviors as constraints due to, for example, an increasing number of dynamic obstacles to avoid, constrained optimization problems tend to become insoluble due to potential conflicts between the constraints.

5.2.2 Optimization Fabrics

Optimization Fabrics, or Fabrics for short, are a class of second-order differential equations called forced spectral semi-sprays [111]. The second-order differential equations define smooth primitive motion behaviors and are guaranteed to optimize to a minimum when forced by a potential function. The benefit of describing primitive motion behaviors as fabrics is that we can compose different smooth behaviors from different task spaces into a single acceleration-based motion policy using pullback and summation operations from differential geometry. Geometric Fabrics, which is a class of Optimization Fabrics, have been applied to several reactive motion control tasks with serial manipulators [112, 113]. As originally formulated, Fabrics have mainly focused on controlling fully-actuated robots. In this work, we make several extensions to the Optimization Fabrics framework to realize prioritized whole-body control of underactuated bipedal humanoid robots.

5.3 Background

The Kinodynamic Fabrics framework represents primitive motion behaviors as Optimization Fabrics. An Optimization Fabric, or Fabric for short, is a forced *spectral semi-spray* of the form

$$M\ddot{x} + f = -\delta_x \psi(x), \tag{5.1}$$

where M is a metric tensor, f is a virtual force, and $-\delta_x \psi(x)$ is the applied force that pushes the system to converge to a local minimum of the potential energy function $\psi(x)$. From (5.1), we can thus define a fabric as the second-order differential equation

$$\ddot{x} = -M^{-1} f \quad - \quad M^{-1} \delta_x \psi(x) \tag{5.2}$$

and define the acceleration-based motion policy as

$$\pi(x, \dot{x}) = \ddot{x}. \tag{5.3}$$

Each fabric component is associated with a task map, ϕ, that maps the generalized coordinates and velocities of the robot, q and $\dot{q}$, to the fabric's task space, x, where $x = \phi(q, \dot{q})$. Given the fabric components, the resultant joint-space acceleration is computed using pullback and summation differential geometry operations.

5.4 Problem Formulation

The general equations of motion of a bipedal robot can be represented as

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = S^\top \tau + J_c^\top F_c, \tag{5.4}$$

where $q, \dot{q}, \ddot{q}$ are the generalized positions, velocities, and accelerations respectively, $D(q)$ is the joint-space mass-inertia matrix, $C(q, \dot{q})$ is Coriolis matrix, $G(q)$ is the gravity vector, S is the torque distribution matrix, and τ, J_c, F_c are the joint torques, end-effector contact Jacobian, and end-effector contact wrench respectively.

The Kinodynamic Fabrics framework seeks to determine the desired joint acceleration $\ddot{q}$ that simultaneously achieves a collection of motion behaviors. From $\ddot{q}$, we can then determine the torque vector τ to apply to the robot's actuators to realize the desired motion.

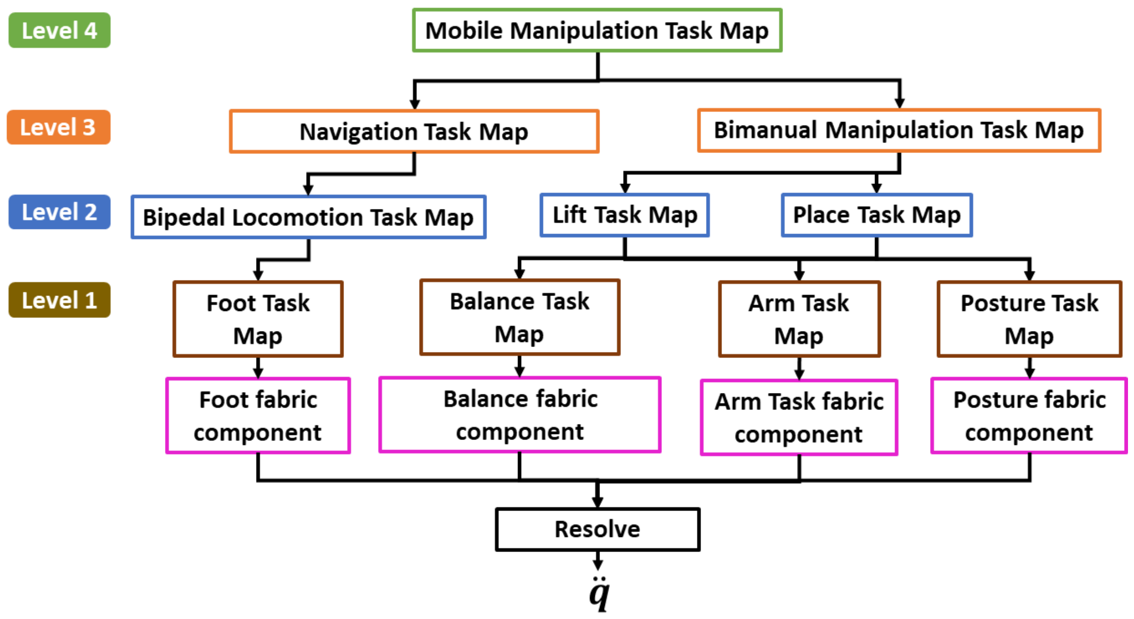


Figure 5.3: An illustration of the Kinodynamic Fabrics Tree for a mobile manipulation behavior. High-level behaviors compute desired target set-points for low-level behaviors. The resulting fabric components are resolved to output joint-space acceleration commands.

5.5 Methodology

Kinodynamic Fabrics take as input the robot's generalized position and velocity vectors $q, \dot{q}$ as well as β, the collection of K primitive motion behaviors whose target set-points are computed by high-level behaviors. The generalized coordinates and velocities $q, \dot{q}$ are first mapped to corresponding task space coordinates x_k for each primitive motion behavior using the behavior's task map ϕ_k. The prioritized Jacobian, J_k^* of each primitive motion behavior's task map is used to compute the derivative of the computed task space coordinates $\dot{x}_k$. Given the task space coordinates and their derivatives, $x, \dot{x}$, the task-space metric tensor $M_k(x, \dot{x})$ as well as the policy $\pi_k(x, \dot{x})$ are computed for each behavior. The pullback and summation operations from differential geometry are then applied to compose all $(M_k(x, \dot{x}), \pi_k(x, \dot{x}))$ of each motion behavior into a single joint-space acceleration vector $\ddot{q}$ (5.9).

The remaining parts of this section describe the various components of the framework in detail.

5.5.1 Primitive Motion Behaviors

We describe a primitive motion behavior (functionally equivalent to “motion primitives” in [67]) as a Kinodynamic fabric represented by the tuple (M, π, ϕ, ρ) where M and π are as defined in Section 5.3 above, ρ is an integer that indicates the priority level of the behavior and ϕ is the task map $\phi(q, \dot{q})$ that maps from the robot's configuration space position and velocity to the behavior's task space. This extension of task maps to transform configuration space positions and velocities to the task space is a unique feature of Kinodynamic Fabrics.

Given β, a collection of K behaviors,

$$\beta = \{(M_1, \pi_1, \phi_1, \rho_1), \quad \ldots, \quad (M_K, \pi_K, \phi_K, \rho_K)\},$$

as well as the robot's generalized position and velocity vectors, $q, \dot{q}$, we solve the Kinodynamic Fabrics problem to output joint-space accelerations $\ddot{q}$.

The theoretical convergence and stability properties of optimization fabrics have been proved in detail by Ratliff et al. for fully actuated systems [111]. The extension of optimization fabrics for its use in the control of underactuated hybrid systems requires a more detailed analysis, which we leave for future work. For example, the stability properties of the fabrics provide no guarantees for the dynamic balance and walking tasks of humanoid robots. As such, the present work will mainly focus on the practical aspects of designing and implementing reactive whole-body behaviors on underactuated bipedal humanoid robots.

5.5.2 Behaviors, Task Maps and Fabric Components

Here, we describe the various classes of primitive motion behaviors we consider in this work alongside their corresponding task maps and motion policies.

The task map ϕ_k for a primitive motion behavior k is a differentiable function that maps coordinates from the generalized coordinates and velocities $q, \dot{q}$ to the task space x_k. Each behavior has a unique task map that depends on the behavior's task space.

Fabric components are motion policies that express the behaviors described in the previous sections. Each motion policy comes along with a task-space acceleration policy $\pi(x, \dot{x})$ as well as a priority metric tensor $M(x, \dot{x})$. The priority metric tensor $M(x, \dot{x})$ is derived from a Finsler energy [111, 112]. It is an invertible matrix that encodes behavior by stretching the task space to indicate the relative priority of dimensions of the task space. The task-space acceleration policy is homogeneous of degree 2 and as such, has the form $\pi(x, \dot{x}) = -||\dot{x}||^2 \cdot \partial_x \psi(x) - B \cdot \dot{x}$ where $\psi(x)$ is a potential energy function whose local minimum satisfies task goals and $-\partial_x \psi(x)$, the negative gradient of the potential energy function, is the force that minimizes the function and B is a damping gain.

Descriptions of the classes of primitive motion behaviors for whole-body control of bipedal humanoid robots, as well as their corresponding task maps, potential energy functions, and priority metric tensors, are as follows:

5.5.2.1 Attractor Primitive Motion Behavior

This behavior generates motions to drive a kinematic or dynamic task-space vector toward a desired value. This behavior can express motions like the motion of the robot's arms, legs, body posture, or center-of-mass to desired poses.

The task map for this behavior is defined in (5.5a), where X_g is the desired value and σ_{att} is a differentiable function that maps the robot's generalized positions and velocities to the task-space vector. The potential energy function for this behavior is defined in (5.5b)

$$\phi(q, \dot{q}) = X_g - \sigma_{att}(q, \dot{q}) \tag{5.5a}$$

$$\psi(x) = \frac{1}{2}\lambda_e x^\top x \tag{5.5b}$$

$$M(x, \dot{x}) = W_{att} I_n \tag{5.5c}$$

where λ_e is a scalar gain parameter that indicates the strength of the attractive force.

The priority metric is defined in (5.5c) where W_{att} is a scalar weight representing the relative importance of the attractor behavior, I_n is an $n \times n$ identity matrix, and n is the size of vector x.

5.5.2.2 Repeller Primitive Motion Behavior

This behavior generates motions to drive a kinematic or dynamic task-space vector away from an undesired value (e.g., to express obstacle or self-collision avoidance motions). For computational tractability when expressing obstacle avoidance behaviors, we specify certain finite control points on the robot's body to which this behavior is applied.

The task map for this behavior is defined in (5.6a) where X_o is the undesired value and σ_{rep} is a differentiable function that maps the robot's generalized positions and velocities to the task-space vector. The potential energy function for this behavior is defined in (5.6b)

$$\phi(q, \dot{q}) = ||X_o - \sigma_{rep}(q, \dot{q})||^2 \tag{5.6a}$$

$$\psi(x) = \frac{\lambda_b}{2} \frac{d_{\max}\mathbf{1} - x}{(d_{\max}x)^\top (d_{\max}x)} \tag{5.6b}$$

$$M(x, \dot{x}) = W_{rep} I_n \Big(s(\dot{x}) \frac{\lambda_{om}}{x^\top x} \Big) \tag{5.6c}$$

where $\mathbf{1}$ is a vector of ones with the same length as x, λ_b is a scalar gain parameter that indicates the strength of the repulsive force, $d_{\max}$ is a scalar parameter that indicates the maximum distance to the obstacle beyond which no repulsive force should be felt.

The priority metric is defined in (5.6c) where λ_{om} is a scalar gain parameter, W_{rep} is a scalar weight representing the relative importance of the repeller behavior, and $s(\dot{x})$ is a velocity-based switching function. $s(\dot{x}) = 1$ if $\dot{x} < 0$ and $s(\dot{x}) = 0$, otherwise. This effectively eliminates the influence of the repulsive function when the control point moves away from the obstacle.

5.5.2.3 Limit Primitive Motion Behavior

This behavior generates motions to keep kinematic or dynamic task-space vectors within desired limits. This is how we express inequality constraints in the Kinodynamic Fabrics framework. For example, this behavior could generate motions to keep joint positions within joint limits, contact forces within friction cone limits, or the zero moment point within the support polygon for robot balance regulation when standing.

The task maps for the upper and lower joint limit behaviors are defined in (5.7a), where q_u is a vector of generalized joint upper limits, and q_l is a vector of generalized joint lower limits. The potential energy function for these behaviors is defined in (5.7b)

$$\begin{aligned} \phi_u(q, \dot{q}) &= q_u - q \\ \phi_l(q, \dot{q}) &= q - q_l \end{aligned} \tag{5.7a}$$

$$\psi(x) = \frac{\lambda_l}{x^\top x} \tag{5.7b}$$

$$M(x, \dot{x}) = W_{lim} I_n \Big(s(\dot{x}) \frac{\lambda_{lm}}{x^\top x} \Big) \tag{5.7c}$$

where λ_l is a scalar gain parameter that indicates the strength of the force.

The priority metric is defined in (5.7c) where λ_{lm} is a scalar gain parameter, W_{lim} is a scalar weight representing the relative importance of the limit behavior, and $s(\dot{x})$ is a velocity based switching function. $s(\dot{x}) = 1$ if $\dot{x} < 0$ and $s(\dot{x}) = 0$, otherwise. This effectively eliminates the influence of the limit barrier function when the joint position is away from the joint limit.

5.5.2.4 High-Level Behaviors

Kinodynamic Fabrics also allow for the expression of high-level behaviors, which determine the desired target set points for lower-level behaviors, with primitive motion behaviors occupying the lowest level in the behavior hierarchy. We organize the behaviors into levels where primitive motion behaviors occupy level 1, higher level behaviors like locomotion and manipulation behaviors occupy level 2, and so on. The highest level of behavior is the Mobile Manipulation behavior, which takes as input a long-horizon plan made up of a sequence of high-level actions (e.g., pick action, navigation action, etc.) and dictates which lower-level behaviors are activated or deactivated in each iteration of the Kinodynamic Fabrics control loop.

Only activated behaviors are evaluated in an iteration of the control loop. Once evaluated, their outputs serve as target set-points for lower-level behaviors that depend on them. The Kinodynamic Fabrics Tree represents the inter-dependence relationships between task maps of behaviors in the framework. Figure 5.3 illustrates the Kinodynamic Fabrics Tree for the framework we use in our mobile manipulation experiments. Even though the task maps for primitive motion behaviors must be differentiable, the task maps for higher-level behaviors do not have to be differentiable.

5.5.3 Prioritization

A unique extension to Kinodynamic Fabrics is the use of prioritized Jacobians [114, 115] to enforce a strict hierarchy of behaviors. Each behavior comes with a factor ρ that indicates the priority of that behavior, with $\rho = 1$ being the highest priority. Lower priority behaviors have the Jacobians of their task maps defined in the nullspace of higher priority Jacobians. We denote the Jacobian J_{ϕ_k} of task map ϕ_k as J_k for brevity. The prioritized Jacobian of behavior k with priority factor ρ is defined as

$$\begin{aligned}
J_k^* &= J_k \cdot S_k \cdot N_{pr(\rho)} \\
N_{pr(\rho)} &= \prod_{j=1}^{\rho-1} N_j \\
N_j &= I - \bar{J}_j J_j \\
\bar{J}_j &= D^{-1} J_j^T (J_j D^{-1} J_j^T)^{-1},
\end{aligned} \tag{5.8}$$

where $pr(\rho)$ indicates behaviors that have a higher priority than ρ, N_j is the nullspace of the behavior with priority j, $\bar{J}_j$ is the dynamically-consistent pseudo-inverse of J_j, D is the configuration space mass-inertia matrix of the robot and S_k is a selection matrix that selects the actuated joints for behavior k.

Behaviors at the same priority are given weights W on their priority metrics $M(x, \dot{x})$ to express their relative importance. In our experiments, to avoid discontinuities when the behavior priorities are changed, only stability behaviors are priority 1. This is because stability behaviors, like the balance behavior, are invariant across different tasks and are the most critical behaviors. All other behaviors are priority 2.

5.5.4 Resolution

Having computed the task-space acceleration policy and priority metric of each behavior, we apply the pullback and summation operations to compute the desired joint-space acceleration vector,

$$\ddot{q} = \left(\sum_{k=1}^{K} J_k^{*\top} M_k J_k^{*} \right)^{\dagger} \cdot \left(\sum_{k=1}^{K} J_k^{*\top} M_k (\pi_k(x_k, \dot{x}_k) - \dot{J}_k^{*} \dot{q}) \right) \tag{5.9}$$

where $(\cdot)^{\dagger}$ denotes the Moore-Penrose pseudo-inverse, $\ddot{q}$ is the desired actuated joint-space acceleration, and x_k, $\dot{x}_k$ is the task space vector and its derivative with respect to time.

5.6 Experiments

In this section, we describe various experiments we perform to evaluate the performance of Kinodynamic Fabrics on whole-body control tasks. Our experiments are performed in simulations and the real world on the Agility Robotics' Digit bipedal humanoid robot [7]. Figure 5.1 shows Digit as a full bipedal humanoid robot with 30-DoF and 20 actuated joints.

5.6.1 Comparison with Whole-body Quadratic Program

We compare the performance of Kinodynamic Fabrics with QPControl, a Quadratic Program whole-body control formulation proposed by Koolen et al. [88]. We compare

Algorithm	**PO+BL**	**PO+EA+BL**	**PO+BL+RE**	**PO+EA+BL+RE**
QPControl [88]	12.60 ± 2.49	13.01 ± 2.53	13.05 ± 2.48	12.88 ± 2.55
Kinodynamic Fabrics	0.81 ± 1.18	0.95 ± 1.12	0.92 ± 1.13	1.06 ± 1.14

Table 5.1: Comparison of the run-time (average duration in milliseconds) of each iteration of the control loop of QPControl and Kinodynamics for different combinations of motion behaviors. PO - Whole-body Posture Behavior, EA - End-effector Attractor behavior, BL - Balance Behavior, RE - Reactivity Behavior

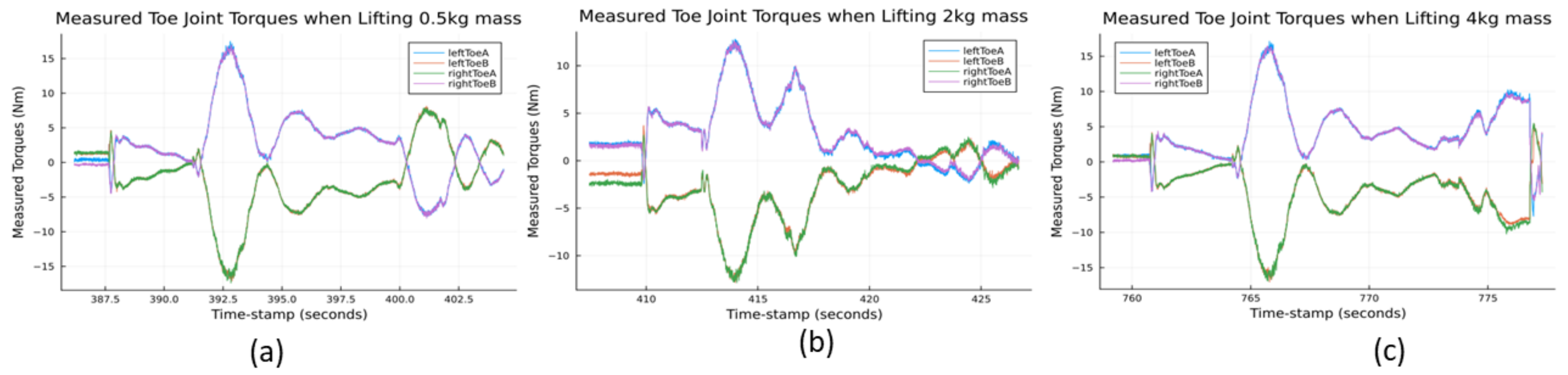


Figure 5.4: Plots of measured toe joint torques of the physical Digit robot while lifting large boxes. We show plots of boxes of total masses 0.5 kg, 2 kg, and 4 kg. On Digit, the bird-inspired toe acts similar to a foot on other humanoids.

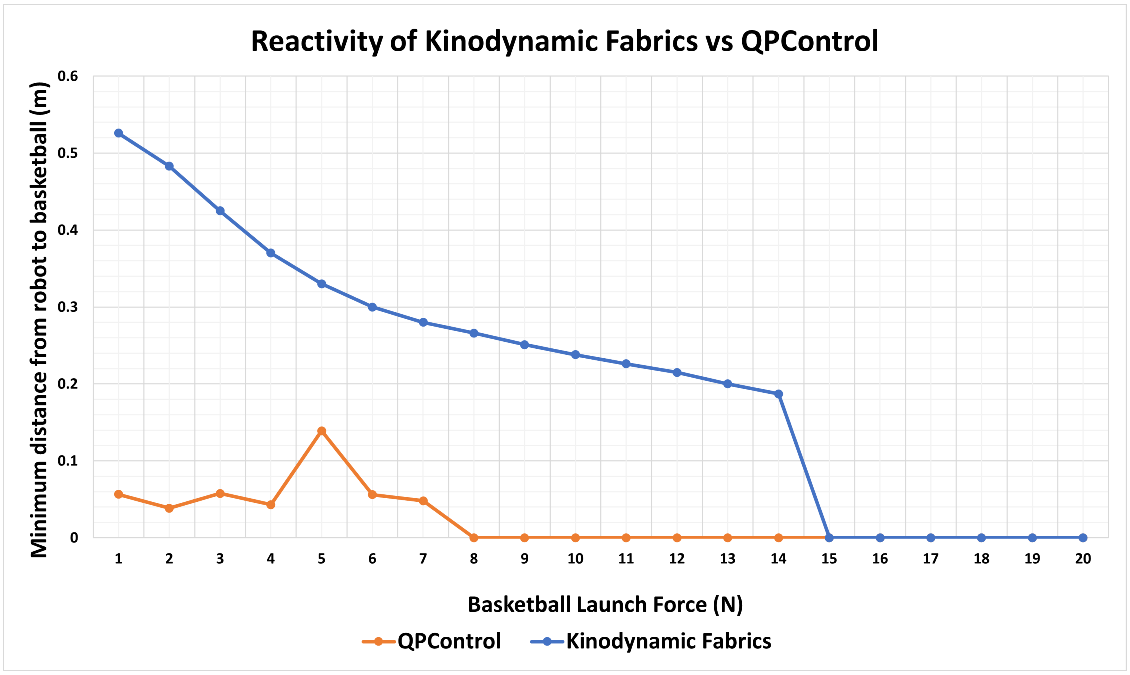


Figure 5.5: Plot of basketball launch forces (in Newtons) against the minimum distance of the basketball to the robot's head (in meters). A minimum distance of 0 meters indicates that the robot fails to dodge the basketball, which strikes the head of the robot and causes it to fall.

the performance of both approaches on whole-body control tasks using the Digit robot in a MuJoCo Physics simulation environment [46] as illustrated in Figure 5.2. The metrics we evaluate for both approaches are 1) *Run-time*, the average duration in milliseconds of each iteration of the control loop when generating motions for different combinations of motion behaviors, and 2) *Reactivity*, the closest distance a dynamic obstacle, in our case, a basketball, comes to the body of the robot when the basketball is shot at the robot at different launch forces. All tasks in the following experiments have a joint-limit motion behavior to keep joint configurations within their nominal limits. The experiments were run on a Razer Blade 15 laptop with an Intel Core i7 processor with 8 cores up to 5.1GHz.

5.6.2 Run-time Simulation Experiment

In this experiment, we compare the duration of each control loop iteration for QPControl [88] and Kinodynamic Fabrics for various motion behavior combinations. The various behaviors are:

- Whole-body posture behavior: This behavior is an attractor behavior (Section 5.5.2.1) that keeps the robot in a nominal, upright posture.
- End-effector attractor behavior: This behavior is an attractor behavior (Section 5.5.2.1) that keeps the left and right wrists of the robot at desired positions.
- Balance behavior: This behavior is a limit behavior (Section 5.5.2.3) that keeps the estimated zero moment point of the robot within the robot's support region.
- Reactivity behavior: This behavior is a repeller behavior (Section 5.5.2.2) that moves the robot's body away from an incoming basketball.

The experimental results in Table 5.1 indicate that Kinodynamic Fabrics is consistently faster than QPControl for all the combinations of motion behaviors. However, in QPControl, inequality constraints like joint limits or balance constraints are hard constraints, while in Kinodynamic Fabrics, they are regarded as weighted soft constraints. As such, these constraints in Kinodynamic Fabrics may be slightly violated.

5.6.3 Reactivity Simulation Experiment

This experiment compares the reactivity capabilities of QPControl and Kinodynamic Fabrics. Specifically, the experiment evaluates how close a basketball shot at the robot at different launch forces comes to the head of the robot as the robot tries to avoid it, as depicted in Figure 5.2. In each experiment, both approaches are tasked with a combination of the whole-body posture, end-effector attractor, balance, and reactivity motion behaviors. The minimum distance of the basketball from the head of the robot is recorded. In the experiments, we vary the launch force of the basketball of mass 0.62kg and radius 0.12m (standard NBA ball mass and size) from 1.0N to 20.0N. Figure 5.5 shows the experimental results for both approaches.

A general observation from the results in Figure 5.5 is that Kinodynamic Fabrics can effectively generate motions to keep the robot's body at a much larger distance from the basketball than QPControl. For launch forces greater than 7.0N, QPControl, due to its slower run-time, cannot generate motions fast enough to avoid collision with the basketball. The basketball collides with the robot's head in such situations, causing it to topple over. Kinodynamic Fabrics have a larger collision avoidance threshold of 14.0N launch force, beyond which it collides with the basketball.

5.6.4 Bimanual Manipulation and Mobile Manipulation on Physical Digit Robot

We demonstrate Kinodynamic Fabrics' capability to react to dynamic uncertainty through bimanual manipulation tasks on the physical Digit robot, as illustrated in Figure 5.1. In these tasks, Digit is made to lift bulky boxes with shifting, non-uniform mass distributions from the ground. The total masses of the boxes range from 0.5kg to 4.0kg. These masses are not modeled in the Kinodynamic Fabrics framework. From the perspective of Kinodynamic Fabrics, the shifting, non-uniform weights of the boxes are external disturbances that need to be attenuated to ensure the generation of smooth motions to keep the robot balanced while lifting the boxes.

To keep Digit balanced while lifting the boxes, we use the balance behavior described in the previous experiment to regulate the zero moment point to stay within the support polygon. This behavior is primarily actuated by the motors in the toe joints of Digit's legs. Figure 5.4 shows plots of the measured torques at the toe joints of Digit's legs as it squats and rises to lift the boxes from the ground.

As can be observed from the plots in Figure 5.4, the largest peaks in measured torques occur when Digit squats and begins to lift the box from the ground. We also demonstrate Kinodynamic Fabrics on cornhole and mobile manipulation tasks with Digit, as depicted in Figure 5.1.

For walking, we use the ALIP one-step ahead prediction method from [8] to specify components of the foot task map. Each virtual constraint is translated into an equivalent attractor primitive motion behavior. Similar set points for torso orientation and center of mass height are used, and the closed-form solution for foot placement is included as a parameter in the foot task map. Videos of all of these tasks can be found on the project webpage.

5.7 Conclusion

We proposed Kinodynamic Fabrics as an approach for the specification, solution, and simultaneous execution of multiple motion tasks in real-time while being reactive to dynamism in the environment. We evaluated the performance of Kinodynamic Fabrics on a variety of whole-body control tasks both in simulation and on a physical Digit robot made by Agility Robotics. Future work should integrate the Terrain-adaptive ALIP-MPC formalism of [42] from Section 3 into the Kinodynamic Fabrics formalism. We expect it to yield more dynamic and robust robot locomotion while preserving dynamic reactivity and bimanual manipulation capabilities.

Acknowledgements: We received partial funding for this project from the Toyota Research Institute and the Qualcomm Innovation Fellowship. Our team thanks Prof. Nima Fazeli for valuable discussions on Geometric Fabrics. We also thank Prof. Odest Chadwicke Jenkins for his encouragement and insightful contributions to mobile manipulation.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Chapter 3 introduces an MPC foot-placement controller that plans ahead considering terrain and friction for better foot placements, agility, and stability. The proposed ALIP-MPC controller is tested on Cassie Blue with successful experiments, including inclined and lateral walking and transitioning between terrains. These results demonstrate improvements over prior controllers and highlight the benefits of incorporating terrain insights into the control strategy.

Chapter 4 combines the ALIP-MPC foot placement strategy with integrated perception state estimation to achieve robust walking on diverse terrains. Challenges related to real-time computation are addressed by formulating a multi-stage optimization method that optimizes foot placements while considering terrain features. Ongoing hardware experiments are being conducted to validate the effectiveness of the perception-locomotion pipeline.

Chapter 5 introduces Kinodynamic Fabrics, a novel approach for whole-body control in bipedal humanoid robots, allowing them to interact with and avoid environmental objects. Kinodynamic Fabrics exhibited improved reactivity and speed compared to a high-performance Quadratic Program-based controller in both simulated and hardware tests on the Agility Robotics Digit robot.

6.2 Future Steps

Bipedal and humanoid locomotion span various robotics domains, encompassing control, perception, state estimation, planning, and more. This thesis primarily focused on enhancing locomotion control and stability by integrating different components within the humanoid system. Nevertheless, several challenges remain, poised for exploration as extensions of this work.

The core ALIP-MPC algorithm that grounds the presented walking strategies offers ample room for refinement. The ALIP model, while simplistic as a dynamical model for foot placement, could be advanced by incorporating ankle torque considerations. A noteworthy approach, highlighted in [116], involves leveraging the ALIP model's linear attributes to formulate an optimization problem that fine-tunes step frequency—crucial for mitigating push recovery and external wrench disturbances. While arm swing motions are tracked to predefined set points, integrating a yaw torque estimator with recent advancements in Angular Center of Mass [117] holds the potential for more meaningful arm trajectory movements.

The Kinodynamic Fabrics framework introduced in Chapter 5 represented an initial stride towards bridging the gap between rigorous model-based whole-body control methodologies and reactive task-oriented motion policies, with inspiration drawn from optimization fabrics. A more intricate analysis of fabric convergence properties within hybrid systems merits further investigation. Lastly, the results presented solely showcased the ALIP-1step foot placement technique proposed by [8]. However, it's worth noting that [42] has demonstrated the enhanced reliability of the ALIP-MPC approach, thereby warranting its integration instead of the current method.

Integrating high-level motion planners, such as CLF-CBF-RRT* [57], with lower-level foot placement and joint control techniques, remains a persistent challenge in the realm of humanoid locomotion. The divergence between these models can introduce discrepancies in reference commands, potentially leading to instability issues. Initial results have been collected on Cassie for ALIP-MPC walking using generated velocity commands generated by [56]. Encouraging advancements, exemplified by [118], offer valuable progress in driving this integration toward a more refined state.

Closing Remarks

As I bring this thesis to a close, I'm filled with an undeniable excitement about the path that humanoid locomotion is embarking upon. The journey of exploring bipedal and humanoid systems has been a rewarding endeavor, and I'm eagerly awaiting the moment when these advancements translate into meaningful changes in our society. The growing interest from private industries serves as a testament to the significance of this work, amplifying its visibility and potential influence. As we peer into the horizon, the potential for progress over the next few decades is both thrilling and promising. The future holds endless possibilities, and I can't help but feel a genuine excitement about what lies ahead!

Appendix A

Input-Output Nonlinear Model Predictive Control for Dynamic Stair Climbing

The following Appendix chapter includes preliminary simulation results of a novel trajectory tracking algorithm that I created for my research qualification exam. The initial results were promising and may provide future inspiration to others.

A.1 Introduction

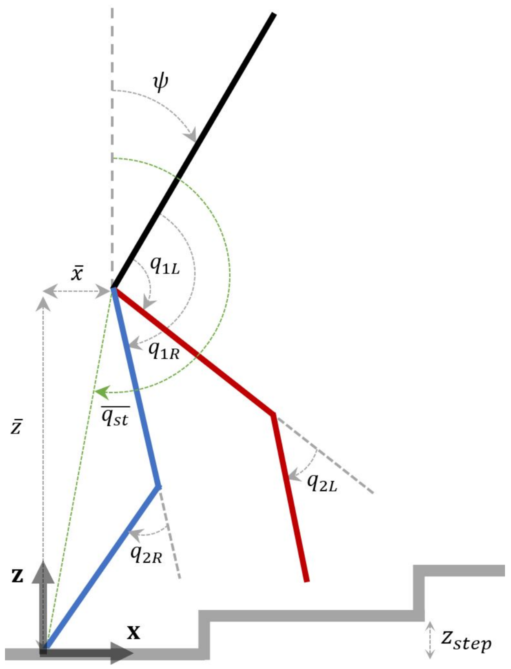


Figure A.1: Five-Link Walker (RABBIT) Schematic. The configuration is represented with floating base coordinates. The absolute stance leg angle $\bar{q}_{st}$ (shown in green) is used for phase-based control because its value monotonically changes.

Stair climbing (ascent or descent) is a task that most able-bodied humans complete with ease regularly. While various legged robots have demonstrated walking on flat terrain, stair climbing remains a challenge requiring a rigorous solution. Quadrupedal and wheeled configurations have been used to navigate stairs successfully. However, these configurations are not robust to steep steps or narrow landing platforms [119, 120]. Bipedal leg configurations are a natural choice for these conditions as they emulate the anatomy for which stairs have been designed.

Stair climbing solutions have already been developed for bipedal robots based on periodic motions (infinite stairs) and static climbing solutions (so-called ZMP walkers) [4, 121, 122]. In addition to the deficiencies already noted, most current solutions are not based on active perception but rather assume a known, perfect geometry and perform blind walking. Just as humans use their perception to alter swing leg trajectories based on stair height, width, depth, and asymmetry, a Model Predictive Control (MPC) based controller can improve real-world performance on bipedal robots.

This work formulates and analyzes a Nonlinear MPC (NMPC) controller for a planar five-link biped robot as a foundation for formulating *environmentally aware* control policies. Periodic stair trajectories (ascent and descent) are computed offline, and the controller performs trajectory tracking. State, output, and control constraints evaluate the controller's robustness to environmental and actuator disturbances within a specified prediction horizon.

Computational inefficiency and intractability have plagued these constraint-based methods from wide use in the past. However, improvements to nonlinear program optimization and hardware have sparked a renewed interest in this field [123]. With added advancements in mapping and vision, we anticipate that similar controller methods will be very useful for navigating dynamic and diverse terrain.

A.1.1 Background

ZMP-based controllers can generate provably safe trajectories. However, dynamic walking controllers can outperform these methods when analyzing certain metrics. ZMP controllers compute torques by constraining the robot's center of pressure to lie within its support polygon. For multi-legged configurations, the solutions are easily calculated and non-unique; however, as the number of leg contacts decreases, greater torque inputs and slower movements are needed to maintain stability. Dynamic walking controllers have no such support for polygon constraints. This class of controllers is designed to utilize impact energy to stabilize walking gaits, typically resulting in more agile and energy-efficient motions. For this reason, we have implemented a dynamic walking controller.

Both error-based and model-based trajectory tracking controllers have found success in bipedal locomotion [40, 4, 124]. Error-based controllers like Proportional-Derivative (PD) control are reactive and can handle model uncertainties. In contrast, model-based controllers, such as feedback linearization, can achieve significantly lower tracking error but are prone to failure as model uncertainty dominates. Model Predictive control can give the best of both worlds as the model dynamics are used to predict future trajectories. At the same time, the current tracking error is computed in a cost that is minimized to generate control commands. For these reasons, we use a predictive control method.

A.1.2 Contributions

We propose an Input-Output Nonlinear Model Predictive Control (IO-NMPC) method for achieving stable periodic stair-climbing motions. Previous works implement a form of input-output NMPC. However, their optimization problem is formulated using the linear dynamics of the transformed system [125, 126]. Conversely, our method uses the original nonlinear dynamics to predict future states. Another NMPC method was used to implement ZMP-based constraints for stair climbing in [127]. However, our reference trajectories are generated using the Hybrid Zero Dynamics (HZD) framework for dynamic walking [4]. We also implement IO-NMPC on a hybrid system to show periodic convergence instead of impacts.

A.2 Planar Five-Link Walker

A planar five-link biped robot (RABBIT) is used to analyze the NMPC controller performance for various trajectories (schematically shown in Figure A.1). The stair-climbing task imposes a hybrid system, so we must consider continuous and discrete dynamics. The continuous dynamics of the robot are derived from the Euler-Lagrange equations (A.1) and are parametrized by floating base coordinates,

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + J_c^T\lambda, \tag{A.1}$$

where $q = [\bar{x}, \bar{z}, \psi, q_{1R}, q_{2R}, q_{1L}, q_{2L}]^T$ are the generalized coordinates. Since this is an underactuated system, the control motor torques $u \in \mathbb{R}^4$ are only applied at the q_i joints. We assume that the robot has point feet; therefore, $\lambda \in \mathbb{R}^2$ is composed of horizontal and vertical ground reaction forces (no moments).

An additional contact constraint is required for the continuous dynamics of the floating base model. The constraint,

$$\begin{aligned} 0 &= \ddot{p}_{stancefoot} \\ &= J_c\ddot{q} + \dot{J}_c\dot{q}, \end{aligned} \tag{A.2}$$

sets the acceleration of the foot position to zero and relates generalized coordinates and velocities to this quantity. $p_{stancefoot}$ is the position of the stance foot and J_c is the foot contact jacobian.

The impact model,

$$\begin{bmatrix} \dot{q}^{+} \\ \lambda^{+} \end{bmatrix} = \begin{bmatrix} D(q^{-}) & -J^{T}_{swingfoot}(q^{-}) \\ J^{T}_{swingfoot}(q) & 0^{-1}_{2\times2} \end{bmatrix}^{-1} \begin{bmatrix} D(q^{-})\dot{q}^{-} \\ 0_{2\times1} \end{bmatrix}, \tag{A.3}$$

is applied when the swing leg makes contact with the ground. The minus (-) and plus (+) symbols represent instantaneous quantities before and after impact, respectively. The block matrix is always invertible for positive definite $D(q)$ and full rank $J_{swingfoot}$.

For this model to hold, we make the following assumptions: (i) pure elastic collision, (ii) ground impact forces on the swing leg are treated as impulses, (iii) there is no slipping or rebounding at impact $(q^{-} = q^{+})$, and (iv) the former stance leg foot releases from the ground. These assumptions should be modified when operating on slippery or uneven terrain.

Lastly, we update the states after impact with a reset map,

$$x^{+} = \begin{bmatrix} H_{reset}q^{-} \\ \dot{q}^{+} \end{bmatrix} \tag{A.4}$$

$$\text{where } H_{reset} = \begin{bmatrix} I_{3\times3} & 0_{3\times2} & 0_{3\times2} \\ 0_{2\times3} & 0_{2\times2} & I_{2\times2} \\ 0_{2\times3} & I_{2\times2} & 0_{2\times2} \end{bmatrix}.$$

Because we are using a symmetric model, the corresponding thigh and knee joint coordinates of opposite legs can be switched at impact.

A.3 Trajectory/Gait Generation

There are many different stair climbing tasks (i.e., stopping on a step, aperiodic stair climbing, etc.); however, for the scope of this research, we only analyze periodic climbing motions. These reference trajectories are generated using the Fast Robotics Optimization and Simulation Toolbox (FROST) [1]. This toolbox formulates a nonlinear program with direct collocation methods based on the Hybrid Zero Dynamics (HZD) framework [4, 124, 128].

The HZD principle guarantees low-dimensional periodic stability for prescribed virtual constraints that satisfy a hybrid invariance condition. For the case of the RABBIT model, the torso pitch angle ψ and the four relative joint coordinates q_i must be equivalent at the beginning of the step and after impact and reset. For simplicity, we have chosen the virtual constraints (A.5) to equal the thigh and knee joints (i.e., $y \in \mathbb{R}^4$). The desired trajectories h^d are bezier polynomials (A.6) whose shape is governed by the coefficients α. These polynomials exist for $s \in [0, 1]$. M refers to the degree of the polynomial and is user-defined. The virtual constraints are

$$y = h(x) - h^d(\alpha, s), \tag{A.5}$$

$$\text{where } h(x) = \begin{bmatrix} 0_{4\times3} & I_{4\times4} & 0_{4\times7} \end{bmatrix} x$$

$$\text{and } h^d(\alpha, s) = \sum_{k=0}^{M} \alpha_k^i \frac{M!}{k!(M-k)!} s^k (1-s)^{M-k}. \tag{A.6}$$

It is important to delineate between phase-based and time-based desired trajectories when using virtual constraints. We use a phase-based implementation; thus, h^d is parametrized by the phase variable s. As previously stated, $s \in [0, 1]$ must be strictly monotonically increasing. The absolute stance leg angle $\bar{q}_{st}$ has been used previously for parametrization of s [129] and so we define s as

$$s = \frac{\bar{q}_{st}(x) - \bar{q}_{st}^{begin}}{\bar{q}_{st}^{end} - \bar{q}_{st}^{begin}}, \tag{A.7}$$

$$\text{where } \bar{q}_{st} = \psi + q_{1R} + \frac{1}{2} q_{2R}.$$

$\bar{q}_{st}^{begin}$ and $\bar{q}_{st}^{end}$ are constants and correspond to $\bar{q}_{st}$ at the beginning and end of each step. These constants can remain unchanged throughout the trajectory or be updated after each impact. For simplicity, we keep them constant.

A.4 Input-Output Linearization

Input-Output Linearization (IO-L) feedback control is an established control policy to stabilize bipedal walking successfully gaits [4]. The basic principle is that a control law, u, is chosen such that the zero dynamics of the system are linear and exponentially stable. The continuous dynamical system described by (A.1) and (A.2) can be written in a modified control affine form,

$$\begin{aligned} \dot{x} &= f(x) + g(x)u + j(x)\lambda \\ y &= h(x) - h^d(\alpha, s), \end{aligned} \tag{A.8}$$

where $x = [q, \dot{q}]^T$.

By inspection, y has relative degree 2, and therefore, the virtual constraint acceleration ($\ddot{y}$) can be written as a function of the state, wrench, and phase (s). The control u is computed (A.10) by canceling all previous dynamics of $\ddot{y}$ and inserting user-defined linear dynamics, v_L. The equation can be further reduced by substituting in λ, which is uniquely found by combining (A.1) and (A.2). For simplicity, we use a PD control law (A.11) to achieve these stabilizing dynamics (LQR is another acceptable method and will give a similar output feedback law for the integrator system). The following equations can mathematically describe the above:

$$\ddot{y} = L_f^2 h(x) + L_j L_f h(x)\lambda + L_g L_f h(x) u - \ddot{h}^d(\alpha, s), \tag{A.9}$$

$$u = (L_g L_f h(x))^{-1}(v^L - L_f^2 h(x) - L_j L_f h(x)\lambda + \ddot{h}^d(\alpha, s)), \tag{A.10}$$

$$v^L = -K_d \dot{y} - K_p y. \tag{A.11}$$

The Lie derivative notation is used for brevity. An important observation is that the decoupling matrix $(L_g L_f h(x))$ is not well-conditioned for all choices of $h(x)$, and so the virtual constraints must be chosen carefully. The error dynamics will stabilize as long as $K_p, K_d > 0$. A more structured approach computes the gains as a function of the desired settling time and damping ratio.

A.5 Input-Output NMPC Feedback Control

The IO-NMPC optimization problem is described as

$$\begin{aligned}
\min_{\mathbf{Z}} \quad J = \sum_{k=0}^{N-1} [(x_k - \hat{x}_k)^T Q (x_k - \hat{x}_k) + (u_k - \hat{u}_k)^T R (u_k - \hat{u}_k)] + \\
\|\ddot{y}_k - v_k^L\|_2^2 + [y_N;\ \ \dot{y}_N]^T Q_{term} [y_N;\ \ \dot{y}_N] \\
\textbf{subject to} \\
x_{k+1} = x_k + \Delta T f(x_k, v_k) \\
v_k = [L_g L_f h(x_k)]^{-1}[u_k + {v^L}_k - L_f^2 h(x_k) - L_j L_f h(x)\lambda_k + \ddot{h}^d(\alpha_k, s_k)] \\
\lambda_k = [J_c D^{-1} J_c^T]^{-1}[J_c D^{-1}(C\dot{q}_k + G - B u_k) - \dot{J}_c \dot{q}_k] \\
v_k^L \rightarrow \text{see Equation (A.11)} \\
y_k, \dot{y}_k, \ddot{y}_k \rightarrow \text{see Equation (A.5,A.9)} \\
\ddot{h}^d(\alpha_k, s_k) \rightarrow \text{see Equation (A.6,A.7)} \\
x_0, x_k \in \mathbf{X}, \quad u_k \in \mathbf{U},
\end{aligned} \tag{A.12}$$

where $\mathbf{Z} = \{x_i, u_j, \lambda_j \ : \ i = 0, ..., N, j = 0, ..., N-1\}$ is the decision variable set, and x_i, u_j, and λ_j are the state, control input, and wrench, respectively, over the prediction horizon, N. $Q \succcurlyeq 0$ is the running stage penalty, $R \succ 0$ is the control input penalty, and $Q_{term} \succcurlyeq 0$ is the terminal cost. $\hat{\mathbf{X}} = \{\hat{x}_i, \hat{u}_j, \alpha_i, s_i : \ i = 0, ..., N, j = 0, ..., N-1\}$ is the reference trajectory set defined prior to optimization. The state and control constraint sets $\mathbf{X}$ and $\mathbf{U}$ form convex polytopes via the union of box constraints. The construction of $\mathbf{U}$ in the current problem formulation is not intuitive; therefore, a new constraint set $\mathbf{V}$ can be added, which imposes torque saturation box constraints on v_k, while $\mathbf{U}$ remains large. The nonlinear dynamics are updated with a simple forward Euler integration method.

A.5.1 Cost Function Intuition

The IO-L solution is equivalent to the unique point-wise solution of (A.13), subject to continuous dynamics shown in (A.1) and (A.2). The minimization in (A.12) can be viewed as an approximation of

$$\begin{aligned} u^*_{IO-L} = \arg\min_u ||\ddot{y} + K_d\dot{y} + K_p y||_2^2 \\ \textbf{subject to } \text{Equations } \ (A.1, A.2) \end{aligned} \tag{A.13}$$

over a desired prediction horizon subject to state and control constraints.

To guarantee stability and recursive feasibility of the optimization problem, a terminal cost is required to vanish as the reference error approaches zero when the current states and control inputs are admissible [130]. We give no admissibility verification, but it is a reasonable assumption that state and control values within a local region of the reference trajectory are admissible. The terminal penalty matrix Q_{term} is set as the solution to the continuous-time algebraic Riccati equation by solving the infinite-horizon Linear Quadratic Regulator (LQR) problem for the IO-L

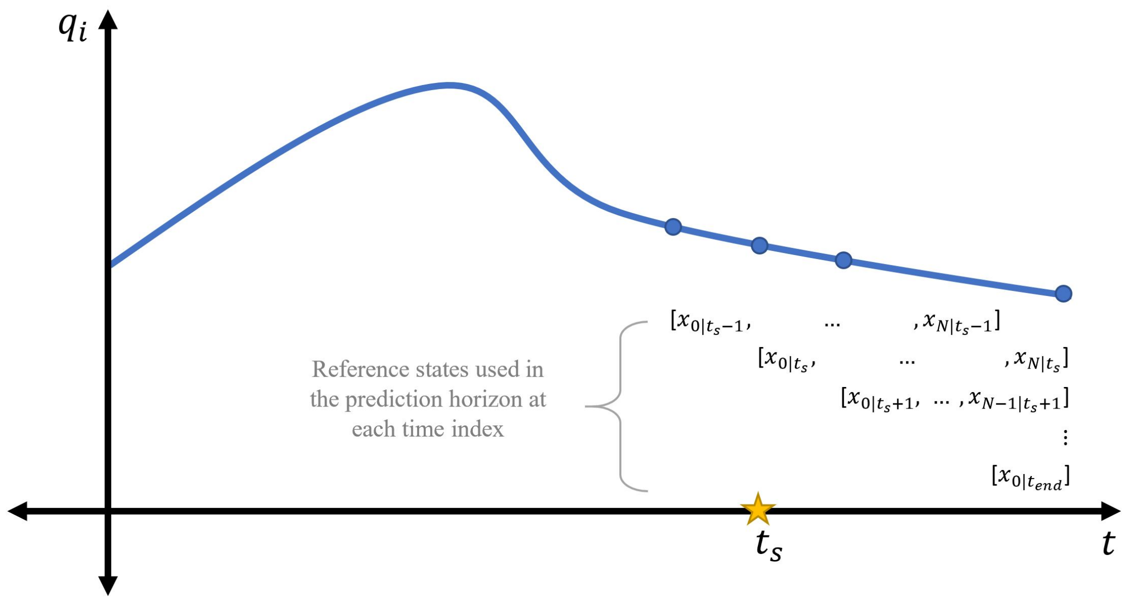

q_i
$[x_{0|t_s-1},\ \cdots\ ,x_{N|t_s-1}]$
$[x_{0|t_s},\ \cdots\ ,x_{N|t_s}]$
$[x_{0|t_s+1},\ \cdots\ ,x_{N-1|t_s+1}]$
$\vdots$
$[x_{0|t_{end}}]$
Reference states used in the prediction horizon at each time index
t_s
t

Figure A.2: Shrinking Horizon Schematic. t_s represents the shrinking horizon time. All subsequent prediction horizons will be shortened to ensure that the last state in the prediction horizon is the final reference state.

output dynamics (with pre-specified gains K_p, K_d).

A.5.2 Shrinking Prediction Horizon Modification

A shrinking prediction horizon modification can be made to simplify formulation modifications or when there is uncertainty in the impact model. Rather than propagate unknown dynamics, the prediction horizon can be decreased relative to your previous knowledge of when the impact should occur. For simplicity, we implement this shrinking horizon method (Figure A.2). A more accurate formulation would forego the shrinking horizon to instead predict and switch dynamics based on the estimated time to impact. A more complex formulation of the disturbance preview problem discussed in the next problem could be used.

A.5.3 Disturbance Preview Formulation

As previously mentioned, one motivation for this work is to incorporate perception information in formulating stable walking and climbing controllers. Therefore, we

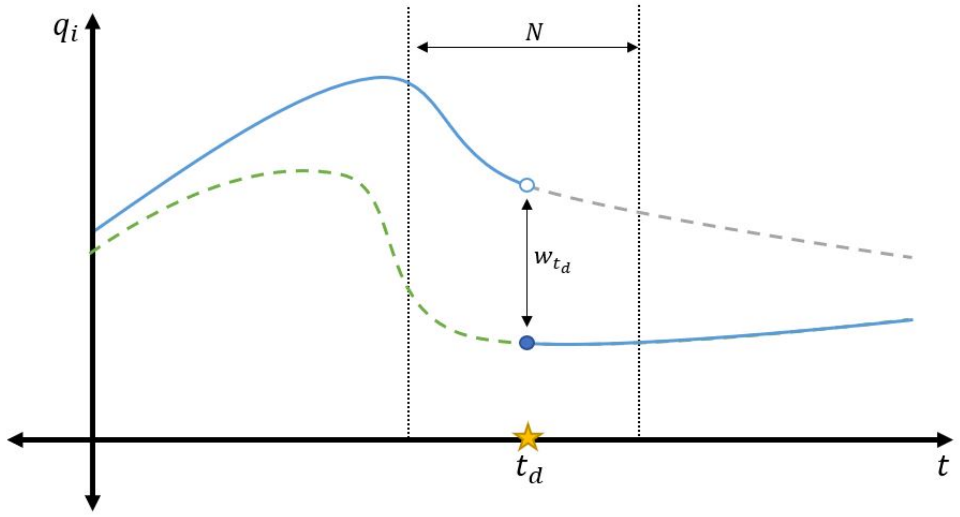


Figure A.3: New Reference Trajectory for Incorporating Disturbance Preview

reformulate (A.12) so that a new reference trajectory can be used mid-step without recompiling the solver. This new task is a disturbance preview problem, with the disturbance defined as

$$w_{t_d} = \tilde{x}_{t_d} - \hat{x}_{t_d}, \tag{A.14}$$

where $\hat{x}_{t_d}$ and $\tilde{x}_{t_d}$ are the states of previous and new reference trajectories. It is important to note that even though the formulation developed below is for a stair height change, with slight modifications, it can capture other disturbances, such as impact. In this case, an additional impact update and reset map equality constraint must be inserted at the appropriate time step.

To simplify the problem, we assume that we know, a priori, the time at which a new trajectory is set, allowing us to include new reference errors in the prediction horizon. Figure A.3 depicts a simple diagram of this preview setup. As a design choice, the new reference trajectory is continuous at the switching time t_d. The new desired state at the time of reference switch is

$$\tilde{x}_{t_d} = \hat{x}_{t_d-1} + \Delta T f + w_{t_d}. \tag{A.15}$$

The disturbance preview optimization problem can then be reformulated by modifying the forward Euler integration matrix inequality constraints, as shown below,

$$
\begin{aligned}
&\min_{Z} \quad J_N \\
&\textbf{subject to} \\
&\begin{bmatrix} x_0 - \hat{x_0} \\ x_1 - (x_0 + \Delta T f(x_0, u_0)) \\ \vdots \\ x_{k_{t_d}} - (x_{k_{t_d}-1} + \Delta T f(x_{k_{t_d}-1}, u_{k_{t_d}-1})) \\ \vdots \\ x_N - (x_{N-1} + \Delta T f(x_{N-1}, u_{N-1})) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ w_{t_d} \\ \vdots \\ 0 \end{bmatrix} \\
&\rightarrow \text{(Include previous constraints).}
\end{aligned} \tag{A.16}
$$

A.6 Preliminary Results

A direct multi-shooting approach is used to solve (A.12) by increasing the sparsity and, thus, the efficiency of the optimization solver. The problem is pre-processed using the Casadi algorithmic differentiation package and is solved with IPOPT [84, 131]. Various stair climbing trajectories are generated using FROST (0, 5, 7.5, and 10 cm, ascent and descent, with average step velocities of 0.5 and 0.75 m/s). An example animation is shown in Figure A.4 for a 5 cm stair ascent with an average velocity of 0.50 m/s. The unconstrained, 0-step prediction horizon version of the IO-NMPC controller generates comparable periodic trajectories to the IO-L solutions. Figure A.5 shows the state position and control trajectories over a ten-step sequence.

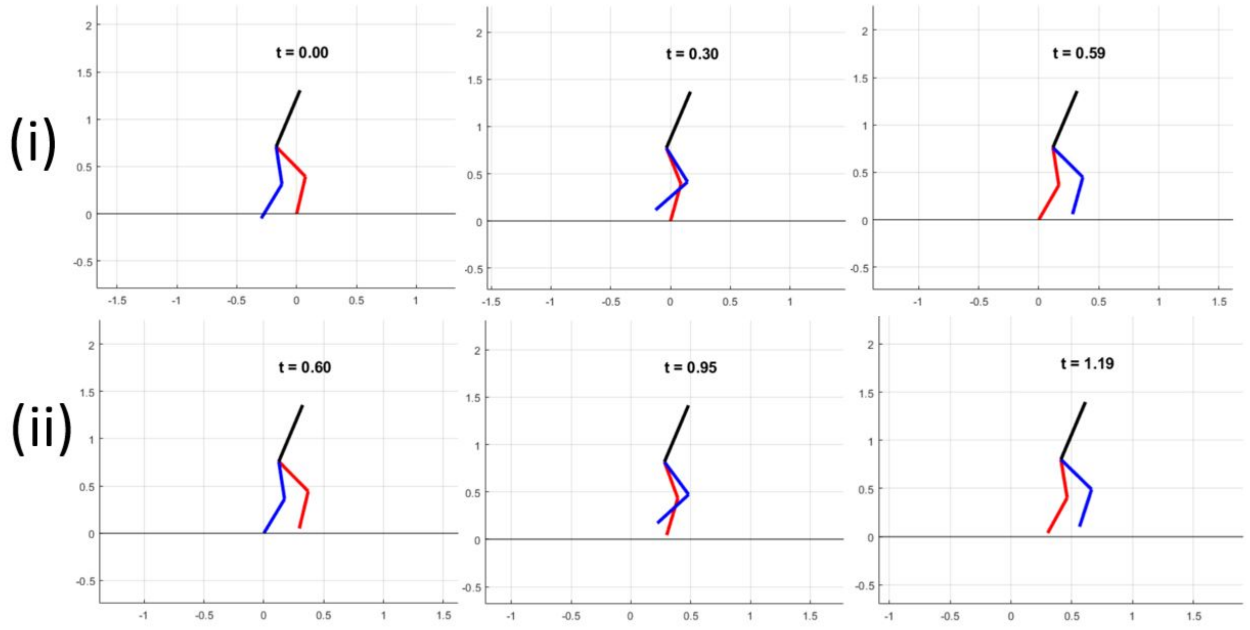


Figure A.4: 2-step animation result for stair ascent with IO-NMPC control. The first step is shown in the top row, and the second step is shown in the second row. The stance leg (red) swaps after impact and becomes the swing leg (blue).

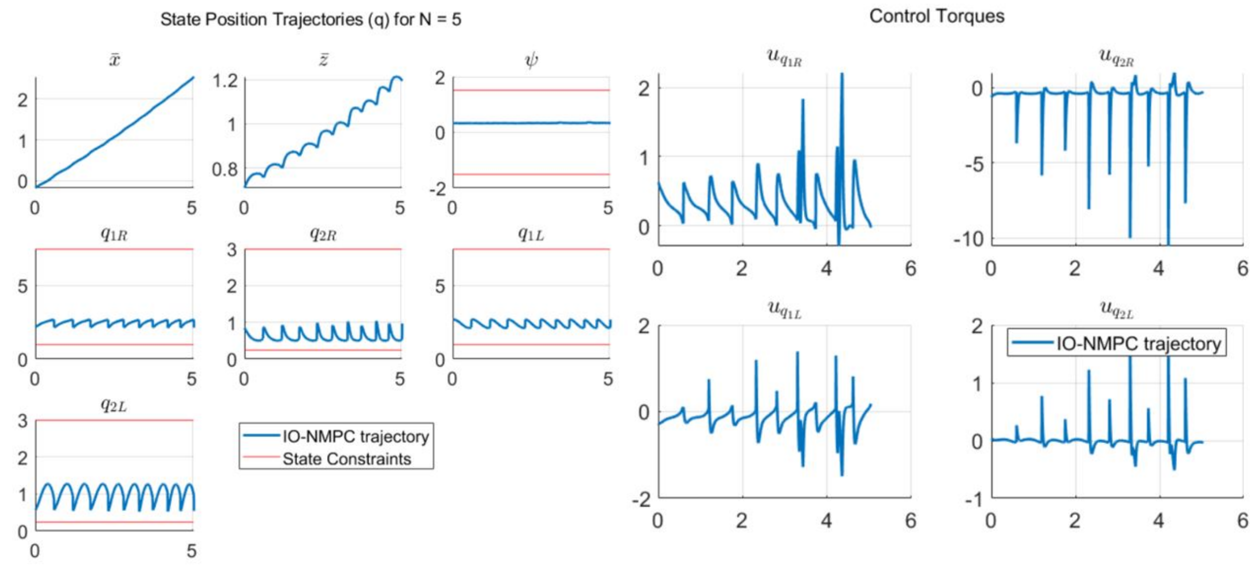


Figure A.5: State and control trajectories corresponding to the IO-NMPC solution of Figure A.4. A 5 cm stair ascent, 0.50 m/s velocity reference trajectory drives the phase-based control method.

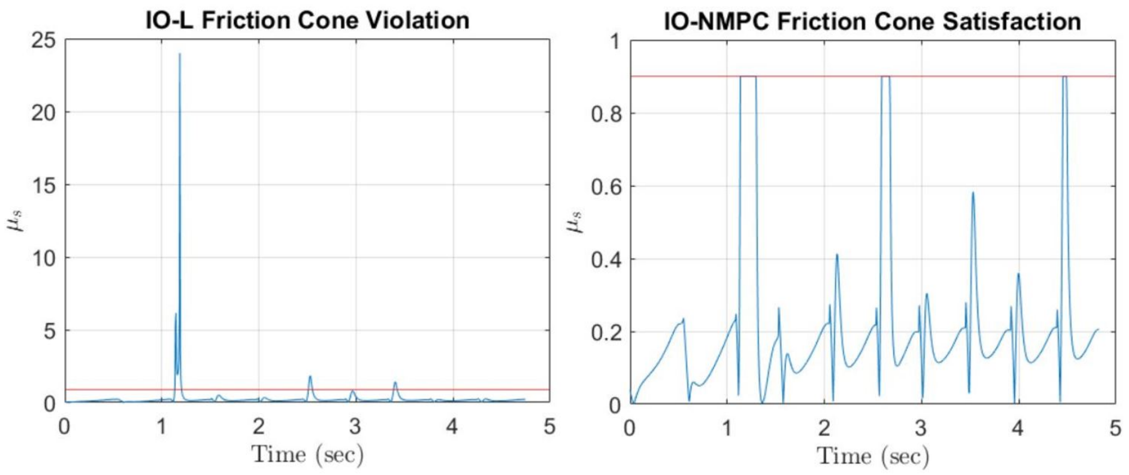


Figure A.6: Friction cone violation and satisfaction for the IO-L and IO-NMPC solution trajectories. A 10 cm ascent (with velocity = 0.75 m/s) reference trajectory was used, and an additional 10 N-m torque constraints were imposed. The static coefficient of friction is set to 0.9, comparable with rubber and concrete surfaces.

To highlight NMPC capabilities, we add output constraints

$$\textbf{(1)}\ \ \left|\frac{\lambda_k^x}{\lambda_k^z}\right| \leq \mu_s \qquad \text{and} \qquad \textbf{(2)}\ \ |v_k| \leq u_{max}. \tag{A.17}$$

These *natural* constraints represent real-world restrictions that would be enforced on the robot via surface conditions and hardware limitations. (A.17.1) represents the friction cone at the stance foot that must be below the coefficient of friction (μ_s) to avoid slipping. (A.17.2) represents motor torque saturation limits. A solution trajectory for both IO-L and IO-NMPC with 1-step prediction is shown in Figure A.6. As shown, the IO-NMPC controller satisfies ground reaction force constraints while the IO-L controller greatly violates them. Swing foot height constraints are also simulated to test obstacle avoidance. The normalized height of the swing foot z_{sw} is derived as a function of phase. Before each update, the bounds of z_{sw} are

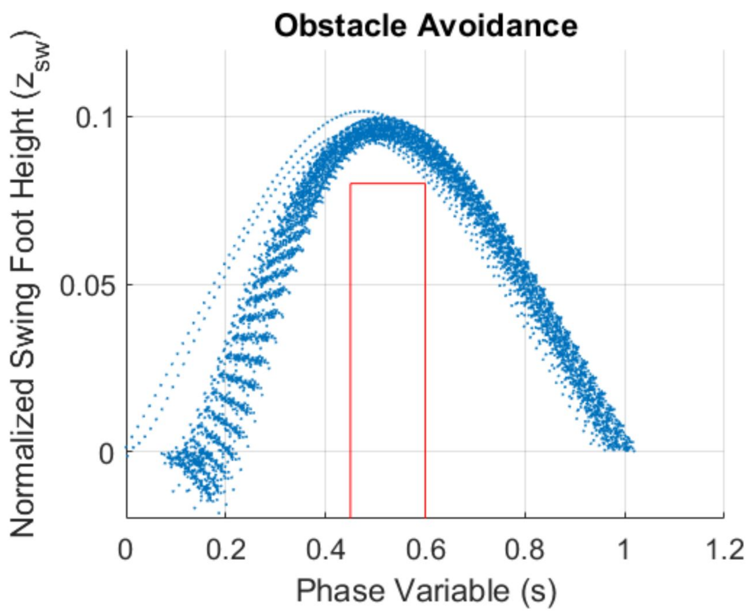


Figure A.7: Normalized swing foot height z_{sw} avoiding obstacle highlighted by the red box

updated based on current and predicted values of phase, as shown below,

$$\begin{aligned} \textbf{if} \quad 0.45 \leq s_k \leq 0.6 \quad (k = 0, ...N-1) \\ z_{swk} \geq 0.08 \quad \textbf{end}. \end{aligned} \tag{A.18}$$

Figure A.7 plots the results for flat-ground walking. Additional disturbance rejection tests were also performed on both controllers by applying external forces at the hip position of RABBIT. The number of failures (fall over) and successes were approximately equal under the same conditions (i.e., PD gains, speed, stair height, and varied initial conditions).

A.7 Discussion

When using unconstrained conditions, the two controllers performed equally well. However, the IO-NMPC outperformed IO-L after adding practical (friction and torque limits) and preferential (swing footpath) output constraints. Supplemental disturbance preview and perturbation rejection tests were performed; however, the two controllers had no significant differences in performance.

As currently simulated, the IO-NMPC controller is not built for online implementation. The IPOPT nonlinear program solver is bulky for a problem of this size, and large prediction horizons become intractable.

At first glance, the current IO-NMPC controller might seem redundant and overdefined. However, from an optimization standpoint, the dynamics equality constraints are highly nonlinear and make the problem non-convex. We argue that the input-output structure of the control input improves feasibility and helps avoid undesirable local minima since IO-L is a solution to the unconstrained minimization problem. Future work will extend this method to more complex models to evaluate its feasibility on real-world robots.

Appendix B

Additional Digit Locomotion Controller Details

B.1 Balance & Manipulation Mode

Dynamic balance is the ability of a bipedal robot to maintain stability during motion while contending with internal and external force disturbances. One standard method for achieving balance involves computing the Center of Pressure (CoP) of the ground reaction force, also known as the Zero Moment Point (ZMP). Ensuring that the ZMP remains within the convex hull of all ground contact points is crucial for maintaining stability [132].

Accurate ZMP calculation for a robot's 3-D dynamics requires precise knowledge of Ground Reaction Forces (GRFs) and joint accelerations. However, the complex mechanical configuration of the Digit robot, which includes closed-chain linkages and spring plates, can render GRF estimates unreliable and inaccurate. As a result, we adopt an alternative approach that relies on inverse kinematics to track desired trajectories. This method enables us to effectively maintain dynamic balance by utilizing joint position and orientation data to generate appropriate actuator commands, circumventing the abovementioned challenges.

B.1.1 Task Map

The task map tracks quantities in the joint space and workspace to represent desired posture and stability quantities. For balance control, the task map h is

$$h : \begin{bmatrix} \text{Torso orientation} \\ p_{\text{LT}\to\text{CoM}} \\ p_{\text{RT}\to\text{CoM}} \\ p_{\text{Base}\to\text{L,Arm}} \\ p_{\text{Base}\to\text{L,Arm}} \end{bmatrix}, \tag{B.1}$$

where the torso orientation is the intrinsic Euler representation ZYX for yaw, pitch, and roll, $p_{\text{LT}\to\text{CoM}}$ is the position vector from the left toe to the CoM, $p_{\text{RT}\to\text{CoM}}$ is the position vector from right toe to the CoM, $p_{\text{Base}\to\text{L,Arm}}$ is the vector from the base link to the left-hand end-effector, and $p_{\text{Base}\to\text{L,Arm}}$ is the vector from the base link to the right-hand end-effector position. This task map coordinates the desired posture of the robot. For sagittal stability, we compute toe motor torques as PD schema using the cart pole Zero Moment Point detailed in [121].

B.1.2 Parameters

Nominal parameters used in the balance and manipulation modes of the digit locomotion controller are presented in Table B.1 and Table B.2.

Table B.1: Balance & Manipulation Mode Parameters for Digit Locomotion Controller. Order of arm joint indices is $\text{LSR}, \text{LSP}, \text{LSY}, \text{LE}, \text{RSR}, \text{RSP}, \text{RSY}, \text{RE}$, following notation in Section 4.2

Balance Mode Parameters	**Values**
Nominal Torso Orientation (Yaw, Pitch, Roll)	$[0, 0, 0]$
$p_{\text{LT}\to\text{CoM}}$	
$p_{\text{RT}\to\text{CoM}}$	
Nominal $p_{\text{Base}\to\text{L,Arm}}$	
Nominal $p_{\text{Base}\to\text{R,Arm}}$	

Table B.2: Gains for Balance & Manipulation Mode on Locomotion Controller. K_p denotes position error gains and K_d denotes damping error gains. Additional subscript notation corresponds to the definitions found in Section 4.2. All gains are equivalent for symmetric left and right joints.

Walking Mode Gains	**Values**	**Walking Mode Gains**	**Values**
-	-	Damping Multiplier (ρ)	0.9
$K_\mathrm{p,HR}$	800 N-m/rad	$K_\mathrm{d,HR}$	$\rho \cdot 66.849$ N-m-s/rad
$K_\mathrm{p,HY}$	500 N-m/rad	$K_\mathrm{d,HY}$	$\rho \cdot 26.1129$ N-m-s/rad
$K_\mathrm{p,HP}$	500 N-m/rad	$K_\mathrm{d,HP}$	$\rho \cdot 38.05$ N-m-s/rad
$K_\mathrm{p,K}$	800 N-m/rad	$K_\mathrm{d,K}$	$\rho \cdot 38.05$ N-m-s/rad
$K_\mathrm{p,T}$	800 N-m/rad	$K_\mathrm{d,T}$	$\rho \cdot 28.5532$ N-m-s/rad
$K_\mathrm{p,SR}$	100 N-m/rad	$K_\mathrm{d,SR}$	$\rho \cdot 66.849$ N-m-s/rad
$K_\mathrm{p,SP}$	100 N-m/rad	$K_\mathrm{d,SP}$	$\rho \cdot 66.849$ N-m-s/rad
$K_\mathrm{p,SY}$	100 N-m/rad	$K_\mathrm{d,SP}$	$\rho \cdot 26.1129$ N-m-s/rad
$K_\mathrm{p,E}$	100 N-m/rad	$K_\mathrm{d,SP}$	$\rho \cdot 66.849$ N-m-s/rad

B.1.3 Results

An image of Figure B.1 displays some of the configurations that Digit can maintain while using the Balance & Manipulation mode of the locomotion controller. A longer video of dynamic balance results can be found at https://github.com/UMich-BipedLab/digit_locomotion_controller.

B.2 Constrained Iterative Inverse Kinematics for Digit

The inverse kinematics problem is one defined by the following optimization

$$\begin{gathered}\min_{\dot{q}} ||\dot{x}^{\text{task}} - J\dot{q}|| \\ \textbf{subject to} \\ \text{Closed Chain Kinematics Constraints (B.3)},\end{gathered} \tag{B.2}$$

where n_j is the number of joint coordinates, n_t is the number of tasks, $\dot{x}^{\text{task}} \in \mathbb{R}^{n_t}$ is the vector of desired task velocities, $\dot{q} \in \mathbb{R}^{n_j}$ is the joint velocities vector, and $J \in \mathbb{R}^{n_t \times n_j}$ is the jacobian of the associated tasks and the closed chain kinematics

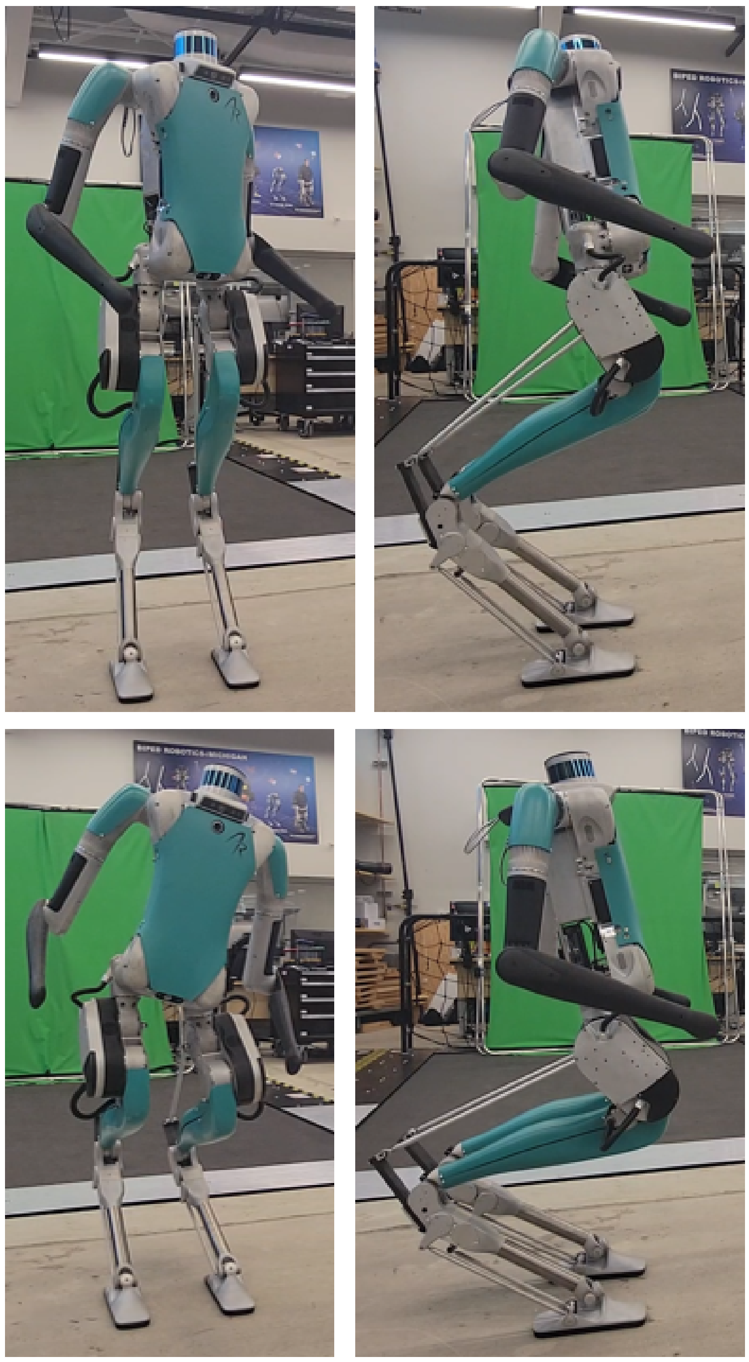


Figure B.1: Variety of orientations and center of mass height set points using the digit locomotion controller for dynamic balance.

constraint is

$$\begin{aligned} q_{\text{left,knee}} + q_{\text{left,shin}\rightarrow\text{tarsus}} &= 0 \\ q_{\text{right,knee}} + q_{\text{right,shin}\rightarrow\text{tarsus}} &= 0. \end{aligned} \tag{B.3}$$

This constraint is valid with the assumption that the springs on each leg remain rigid and unbent.

By inspection, many task map variables are already joint position variables. Therefore, inverse kinematics is only needed to relate six tasks to joint variables (CoM vector from the left and right toe, respectively). For computational efficiency, it is better to reduce the task space to these six variables and implement a different algorithm to solve for the six relevant joint coordinates. These coordinates are each leg's hip roll, hip pitch, and knee joints.

Briefly ignoring the constraint of (B.2), the solution to the least-squares optimization problem is

$$\dot{q}^* = J^\dagger \dot{x}^{\text{task}}, \tag{B.4}$$

where $J^\dagger = (J^T J)^{-1} J^T$. For the inverse kinematics problem, we also seek to find the desired joint position, not just the velocity. The solution in (B.4) can be used to formulate an iterative algorithm. Applying finite difference and removing equivalent terms, we arrive at

$$q_{k+1} = q_k + J^\dagger (x_{k+1} - x_k) \tag{B.5}$$

where the indices $k, k+1$ are values at consecutive update intervals.

Next, notice that the constraint in (B.3) can be removed by modification of the newly altered square Jacobian matrix. Notice that the time derivative of a function $\frac{d}{dt} f(q_{\text{knee}}, q_{\text{shin}\rightarrow\text{tarsus}}) = J_{\text{knee}} \dot{q}_{\text{knee}} + J_{\text{shin}\rightarrow\text{tarsus}} \dot{q}_{\text{shin}\rightarrow\text{tarsus}}$ can be further simplified by applying (B.3). The result is

$$\frac{d}{dt}f(q_{\text{knee}}, q_{\text{shin}\to\text{tarsus}}) = (J_{\text{knee}} - J_{\text{shin}\to\text{tarsus}})\dot{q}_{\text{knee}}. \tag{B.6}$$

We define Algorithm B.1 using this result. A reference to the notation used in this algorithm can be found in Chapter 4. In practice, mathematical calculations are computed using efficient algorithms in the Eigen Library [133].

Algorithm B.1: Constrained Inverse Kinematics for Digit

```
1  Compute desired task map function: $f_{fk}(\cdot)$;
2  Initialize desired task values: $x^{\text{des}}, \dot{x}^{\text{des}}$;
3  Initialize joint solution guess with current state: $q_{\text{IK}}, \dot{q}_{\text{IK}}$;
4  Set tolerance threshold: $\varepsilon > 0$;
5  Compute task error from joint guess: $x_{\text{error}} = f_{fk}(q_{\text{IK}}) - x^{\text{des}}$;
6  while $x^{\text{error}}_{\text{fk}} \leq \varepsilon$ do
7      Enforce Joint Constraints [Eq. (B.3)]: Update $q_{IK}$ ;
8      Compute Task Jacobian: $J = \frac{\partial f_{fk}}{\partial q}(q_{\text{IK}})$;
9      Enforce Jacobian Constraint [Eq. (B.6)]: Update $J$;
10     Compute Jacobian Psuedoinverse [Eq. (B.4)]: $\Delta q = J^{\dagger} x_{\text{error}}$;
11     Update joint guess: $q_{\text{IK}} -= \Delta q$;
12     Update task error: $x_{\text{error}} = f_{fk}(q_{\text{IK}}) - x^{\text{des}}$;
13 end
14 Compute desired joint velocities: $\dot{q}_{\text{IK}} = J^{\dagger}\dot{x}^{\text{des}}$;
15 return $q_{\text{IK}}, \dot{q}_{\text{IK}}$
```

A downside of this method is that it is developed to track set points. The inverse kinematics approach calculates joint positions and velocities based on fixed task map reference positions. This can cause non-smooth movements when desired reference trajectories follow prescribed paths at a desired speed (e.g., crouching up and down). Alternative approaches include constrained QP methods or Kinodynamic Fabrics detailed in Chapter 5. This inverse kinematics method converts task space virtual constraints to joint encoder values for all locomotion modes employed by Digit.

BIBLIOGRAPHY

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

[2] Theresa Klein. *A Neurorobotic Model of Humanoid Walking*. PhD thesis, University of Arizona, Dec 2011.

[3] Ryan Batke, Fangzhou Yu, Jeremy Dao, Jonathan Hurst, Ross L. Hatton, Alan Fern, and Kevin Green. Optimizing bipedal maneuvers of single rigid-body models for reinforcement learning. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, page 714–721, Nov 2022.

[4] Eric R. Westervelt, Jessy W. Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, 1 edition, 2007.

[5] J. Pratt, P. Dilworth, and G. Pratt. Virtual model control of a bipedal walking robot. In *Proceedings of International Conference on Robotics and Automation*, volume 1, page 193–198 vol.1, Apr 1997.

[6] Agility Robotics. Digit robot. https://github.com/agilityrobotics/cassie-doc/wiki.

[7] Agility Robotics. Digit robot. https://agilityrobotics.com/robots.

[8] Yukai Gong and Jessy Grizzle. One-step ahead prediction of angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-inspired controller. In *IEEE International Conference on Robotics and Automation*, 2021.

[9] Grant Gibson, Oluwami Dosunmu-Ogunbi, Yukai Gong, and Jessy Grizzle. Terrain-Aware Foot Placement for Bipedal Locomotion Combining MPC, Virtual Constraints, and the ALIP. https://youtu.be/AmPvQMpIHSw, 2021.

[10] Hyeong Ryeol Kam, Sung-Ho Lee, Taejung Park, and Chang-Hun Kim. Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60(2):337–345, Oct 2015.

[11] Takahiro Miki, Lorenz Wellhausen, Ruben Grandia, Fabian Jenelten, Timon Homberger, and Marco Hutter. Elevation mapping for locomotion and navigation using gpu. *arXiv*, Apr 2022. arXiv:2204.12876 [cs].

[12] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[13] Qihao Zhang, Wei Zhao, Shengnan Chu, Lei Wang, Jun Fu, Jiangrong Yang, and Bo Gao. Research progress of nuclear emergency response robot. *IOP Conference Series: Materials Science and Engineering*, 452(4):042102, Dec 2018.

[14] Ítalo Renan da Costa Barros and Tiago Pereira Nascimento. Robotic mobile fulfillment systems: A survey on recent developments and research opportunities. *Robotics and Autonomous Systems*, 137:103729, Mar 2021.

[15] David Lattanzi and Gregory Miller. Review of robotic infrastructure inspection systems. *Journal of Infrastructure Systems*, 23(3):04017004, Sep 2017.

[16] Maria Kyrarini, Fotios Lygerakis, Akilesh Rajavenkatanarayanan, Christos Sevastopoulos, Harish Ram Nambiappan, Kodur Krishna Chaitanya, Ashwin Ramesh Babu, Joanne Mathew, and Fillia Makedon. A survey of robots in healthcare. *Technologies*, 9(11):8, Mar 2021.

[17] Erico Guizzo. By leaps and bounds: An exclusive look at how boston dynamics is redefining robot agility. *IEEE Spectrum*, 56(12):34–39, Dec 2019.

[18] Simon Zimmermann, Roi Poranne, and Stelian Coros. Go fetch! - dynamic grasps using boston dynamics spot with external robotic arm. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, page 4488–4494, May 2021.

[19] Amanda Bouman, Muhammad Fadhil Ginting, Nikhilesh Alatur, Matteo Palieri, David D. Fan, Thomas Touma, Torkom Pailevanian, Sung-Kyun Kim, Kyohei Otsu, Joel Burdick, and Ali-akbar Agha-Mohammadi. Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 2518–2525, Oct 2020.

[20] C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter. Anymal in the field: Solving industrial inspection of an offshore hvdc platform with a quadrupedal robot. In Genya Ishigami and Kazuya Yoshida, editors, *Field and Service Robotics*, Springer Proceedings in Advanced Robotics, page 247–260, Singapore, 2021. Springer.

[21] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C. Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, Remo Diethelm, Samuel Bachmann, Amir Melzer, and Mark Hoepflinger. Anymal - a highly mobile and dynamic quadrupedal robot. In

2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), page 38–44, Oct 2016.

[22] Abhijit Mahapatra, Shibendu Shekhar Roy, and Dilip Kumar Pratihar. *Multi-body Dynamic Modeling of Multi-legged Robots*. Cognitive Intelligence and Robotics. Springer, Singapore, 2020.

[23] Jing Liu, Min Tan, and Xiaoguang Zhao. Legged robots — an overview. *Transactions of the Institute of Measurement and Control*, 29(2):185–202, Jun 2007.

[24] Matthew Spenko, Stephen Buerger, and Karl Iagnemma. *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*. Springer, 2018.

[25] Ross Hartley, Maani Ghaffari, Ryan M Eustice, and Jessy W Grizzle. Contact-aided invariant extended kalman filtering for robot state estimation. *The International Journal of Robotics Research*, 39(4):402–430, Mar 2020.

[26] Victor Dhédin, Haolong Li, Shahram Khorshidi, Lukas Mack, Adithya Kumar Chinnakkonda Ravi, Avadesh Meduri, Paarth Shah, Felix Grimminger, Ludovic Righetti, Majid Khadiv, and Joerg Stueckler. Visual-inertial and leg odometry fusion for dynamic locomotion. *arXiv*, Oct 2022. arXiv:2210.02127 [cs].

[27] Pengcheng Shi, Zhikai Zhu, Shiying Sun, Xiaoguang Zhao, and Min Tan. Invariant extended kalman filtering for tightly coupled lidar-inertial odometry and mapping. *IEEE/ASME Transactions on Mechatronics*, page 1–12, 2023.

[28] John Lygeros, Shankar Sastry, and Claire Tomlin. *Hybrid Systems: Foundations, advanced topics and applications*. Springer Verlag, 2012.

[29] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 1, page 239–246 vol.1, Oct 2001.

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

[31] Mark Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2 edition, 2020.

[32] Sotaro Katayama, Masaki Murooka, and Yuichi Tazaki. Model predictive control of legged and humanoid robots: models and algorithms. *Advanced Robotics*, 2023.

[33] Hamid Sadeghian, Christian Ott, Gianluca Garofalo, and Gordon Cheng. Passivity-based control of underactuated biped robots within hybrid zero dynamics approach. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, page 4096–4101, May 2017.

[34] Zhenyu Gan. *Exploring Passive Dynamics in Legged Locomotion.* Thesis, University of Michigan, 2018. Accepted: 2019-02-07T17:54:42Z.

[35] Philip Holmes, Robert J. Full, Dan Koditschek, and John Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM Review*, 48(2):207–304, 2006.

[36] Bowen Weng, Guillermo A. Castillo, Wei Zhang, and Ayonga Hereid. On safety testing, validation, and characterization with scenario-sampling: A case study of legged robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 5179–5186, Oct 2022.

[37] Athanasios S. Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, May 2017.

[38] Mark W. Spong, Jonathan K. Holm, and Dongjun Lee. Passivity-based control of bipedal locomotion. *IEEE Robotics & Automation Magazine*, 14(2):30–40, Jun 2007.

[39] M.W. Spong and F. Bullo. Controlled symmetries and passive walking. *IEEE Transactions on Automatic Control*, 50(7):1025–1031, Jul 2005.

[40] Yukai Gong, Ross Hartley, Xingye Da, Ayonga Hereid, Omar Harib, Jiunn-Kai Huang, and Jessy Grizzle. Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway. In *2019 American Control Conference (ACC)*, page 4559–4566, Jul 2019.

[41] Yukai Gong and Jessy Grizzle. Zero dynamics, pendulum models, and angular momentum in feedback control of bipedal locomotion. *arXiv preprint arXiv:2105.08170*, 2021.

[42] Grant Gibson, Oluwami Dosunmu-Ogunbi, Yukai Gong, and Jessy Grizzle. Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 6724–6731, Oct 2022.

[43] Jonah Siekmann, Kevin Green, John Warila, Alan Fern, and Jonathan Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. *arXiv*, May 2021. arXiv:2105.08328 [cs].

[44] Xiaobin Xiong, Jenna Reher, and Aaron D. Ames. Global position control on underactuated bipedal robots: Step-to-step dynamics approximation for step

planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, page 2825–2831, May 2021.

[45] Patrick M. Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. Optimization-based control for dynamic legged robots. *arXiv*, Nov 2022. arXiv:2211.11644 [cs].

[46] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 5026–5033, Oct 2012.

[47] Bullet Physics. Bullet real-time physics simulation. https://pybullet.org/wordpress/.

[48] Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv*, Aug 2022. arXiv:2208.07860 [cs].

[49] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Learning humanoid locomotion with transformers. *arXiv*, Mar 2023. arXiv:2303.03381 [cs].

[50] Antonio Loquercio, Ashish Kumar, and Jitendra Malik. Learning visual locomotion with cross-modal supervision. *arXiv*, Nov 2022. arXiv:2211.03785 [cs].

[51] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel van de Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. *Conference on Robot Learning (CoRL)*, page 13, 2019.

[52] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, page 2811–2817, May 2021.

[53] Guillermo A. Castillo, Bowen Weng, Wei Zhang, and Ayonga Hereid. Hybrid zero dynamics inspired feedback control policy design for 3d bipedal locomotion using reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, page 8746–8752, May 2020.

[54] Tadeusz Mikolajczyk, Emilia Mikołajewska, Hayder F. N. Al-Shuka, Tomasz Malinowski, Adam Kłodowski, Danil Yurievich Pimenov, Tomasz Paczkowski, Fuwen Hu, Khaled Giasin, Dariusz Mikołajewski, and Marek Macko. Recent advances in bipedal walking robots: Review of gait, drive, sensors and control systems. *Sensors*, 22(1212):4440, Jan 2022.

[55] Hun-ok Lim and Atsuo Takanishi. Biped walking robots created at waseda university: Wl and wabian family. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1850):49–64, Nov 2006.

[56] Jiunn-Kai Huang and Jessy W. Grizzle. Efficient anytime clf reactive planning system for a bipedal robot on undulating terrain. *IEEE Transactions on Robotics*, page 1–18, 2023.

[57] Jinze Liu, Minzhe Li, Jiunn-Kai Huang, and Jessy W. Grizzle. Realtime safety control for bipedal robots to avoid multiple obstacles via clf-cbf constraints. *arXiv*, Jan 2023. arXiv:2301.01906 [cs].

[58] Lu Gan. *Semantic-Aware Robotic Mapping in Unknown, Loosely Structured Environments*. Thesis, University of Michigan, 2022. Accepted: 2022-05-25T15:20:18Z.

[59] Michał R. Nowicki, Dominik Belter, Aleksander Kostusiak, Petr Cížek, Jan Faigl, and Piotr Skrzypczyński. An experimental study on feature-based slam for multi-legged robots with rgb-d sensors. *Industrial Robot: An International Journal*, 44(4):428–441, Jan 2017.

[60] Péter Fankhauser. *Perceptive Locomotion for Legged Robots in Rough Terrain*. PhD thesis, ETH Zurich, 2018.

[61] Jens-Steffen Gutmann, Masaki Fukuchi, and Masahiro Fujita. 3d perception and environment map generation for humanoid robot navigation. *The International Journal of Robotics Research*, 27(10):1117–1134, Oct 2008.

[62] Hendrik Kolvenbach, Philip Arm, Elias Hampp, Alexander Dietsche, Valentin Bickel, Benjamin Sun, Christoph Meyer, and Marco Hutter. Traversing steep and granular martian analog slopes with a dynamic quadrupedal robot. *Field Robotics*, 2(1):910–939, Mar 2022.

[63] Antonio Loquercio, Ashish Kumar, and Jitendra Malik. Learning visual locomotion with cross-modal supervision. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, page 7295–7302, May 2023.

[64] Omur Arslan and Daniel E Koditschek. Sensor-based reactive navigation in unknown convex sphere worlds. *The International Journal of Robotics Research*, 38(2-3):196–223, 2019.

[65] Santiago Paternain, Daniel E Koditschek, and Alejandro Ribeiro. Navigation functions for convex potentials in a space with convex obstacles. *IEEE Transactions on Automatic Control*, 63(9):2944–2959, 2017.

[66] Jia-chi Wu and Zoran Popović. Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics (TOG)*, 29(4):1–10, 2010.

[67] Matthew J. Powell, Huihua Zhao, and Aaron D. Ames. Motion primitives for human-inspired bipedal robotic locomotion: walking and stair climbing. In *2012 IEEE International Conference on Robotics and Automation*, page 543–549, May 2012.

[68] Katie Byl and Russ Tedrake. Metastable walking machines. *The International Journal of Robotics Research*, 28(8):1040–1064, Aug 2009.

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

[70] Brent Griffin and Jessy Grizzle. Nonholonomic virtual constraints and gait optimization for robust walking control. *The International Journal of Robotics Research*, 36(8):895–922, Jul 2017.

[71] Kaveh Akbari Hamed, Jeeseop Kim, and Abhishek Pandala. Quadrupedal locomotion via event-based predictive control and qp-based virtual constraints. *IEEE Robotics and Automation Letters*, 5(3):4463–4470, Jul 2020.

[72] Hirofumi Miura and Isao Shimoyama. Dynamic walk of a biped. *The International Journal of Robotics Research*, 3(2):60–74, Jun 1984.

[73] Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, page 200–207, Dec 2006.

[74] Johannes Englsberger, Christian Ott, Maximo A. Roa, Alin Albu-Schäffer, and Gerhard Hirzinger. Bipedal walking control based on capture point dynamics. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 4420–4427, Sep 2011.

[75] Ting Wang and Christine Chevallereau. Stability analysis and time-varying walking control for an under-actuated planar biped robot. *Robotics and Autonomous Systems*, 59(6):444–456, Jun 2011.

[76] Xiaobin Xiong and Aaron D. Ames. Orbit characterization, stabilization and composition on 3d underactuated bipedal walking via hybrid passive linear inverted pendulum model. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 4644–4651, Nov 2019.

[77] Jacob Reher, Eric A. Cousineau, Ayonga Hereid, Christian M. Hubicki, and Aaron D. Ames. Realizing dynamic and efficient bipedal locomotion on the humanoid robot durus. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, page 1794–1801, May 2016.

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

[79] Jiunn-Kai Huang, Yukai Gong, Dianhao Chen, Jinze Liu, Minzhe Li, Jianyang Tang, Lu Gan, Ray Zhan, Wami Ogunbi, and Jessy Grizzle. Fully Autonomous on the Wave Field 2021. https://youtu.be/gE3Y-2Q3gco, 2021.

[80] A. Sano and J. Furusho. Realization of natural dynamic walking using the angular momentum information. In *, IEEE International Conference on Robotics and Automation Proceedings*, page 1476–1481 vol.3, May 1990.

[81] Matthew J. Powell, Wen-Loong Ma, Eric R. Ambrose, and Aaron D. Ames. Mechanics-based design of underactuated robotic walking gaits: Initial experimental realynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in 2016 ieee international conference on robotics and automation (icra). ieee, 2016, pp. 1447–ization. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, page 981–986, Nov 2016.

[82] Marc H. Raibert. Legged robots. *Communications of the ACM*, 29(6):499–514, Jun 1986.

[83] J.K. Hodgins and M.N. Raibert. Adjusting step length for rough terrain locomotion. *IEEE Transactions on Robotics and Automation*, 7(3):289–298, Jun 1991.

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

[85] Joel A.E. Andersson and James B. Rawlings. Sensitivity analysis for nonlinear programming in casadi*. *IFAC-PapersOnLine*, 51(20):331–336, 2018. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[86] E. K.-W. Chu, H.-Y. Fan, W.-W. Lin, and C.-S. Wang. Structure-preserving algorithms for periodic discrete-time algebraic riccati equations. *International Journal of Control*, 77(8):767–788, May 2004.

[87] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, page 4304–4311, May 2016.

[88] Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas de Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and Jerry Pratt. Design of a momentum-based control framework and application to the humanoid robot atlas. *International Journal of Humanoid Robotics*, 13(01):1650007, Mar 2016.

[89] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G. Atkeson. Optimization based full body control for the atlas robot. In *2014 IEEE-RAS International Conference on Humanoid Robots*, page 120–127, Nov 2014.

[90] Alexander Stumpf, Stefan Kohlbrecher, David C. Conner, and Oskar von Stryk. Supervised footstep planning for humanoid robots in rough terrain tasks using a black box walking controller. In *2014 IEEE-RAS International Conference on Humanoid Robots*, page 287–294, Nov 2014.

[91] Matthew Johnson, Brandon Shrewsbury, Sylvain Bertrand, Tingfan Wu, Daniel Duran, Marshall Floyd, Peter Abeles, Douglas Stephen, Nathan Mertins, Alex Lesman, John Carff, William Rifenburgh, Pushyami Kaveti, Wessel Straatman, Jesper Smith, Maarten Griffioen, Brooke Layton, Tomas de Boer, Twan Koolen, Peter Neuhaus, and Jerry Pratt. Team ihmc's lessons learned from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):192–208, 2015.

[92] Duncan Calvert, Bhavyansh Mishra, Stephen McCrory, Sylvain Bertrand, Robert Griffin, and Jerry Pratt. A fast, autonomous, bipedal walking behavior over rapid regions. *arXiv*, Jul 2022. arXiv:2207.08312 [cs].

[93] Moonyoung Lee, Youngsun Kwon, Sebin Lee, JongHun Choe, Junyong Park, Hyobin Jeong, Yujin Heo, Min-Su Kim, Jo Sungho, Sung-Eui Yoon, and Jun-Ho Oh. Dynamic humanoid locomotion over rough terrain with streamlined perception-control pipeline. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 4111–4117, Sep 2021.

[94] Robert J. Griffin, Georg Wiedebach, Stephen McCrory, Sylvain Bertrand, Inho Lee, and Jerry Pratt. Footstep planning for autonomous walking over rough terrain. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, page 9–16, Oct 2019.

[95] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. *2014 IEEE-RAS International Conference on Humanoid Robots*, page 279–286, Nov 2014.

[96] Philipp Karkowski, Stefan Oßwald, and Maren Bennewitz. Real-time footstep planning in 3d environments. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, page 69–74, Nov 2016.

[97] Steve Tonneau, Daeun Song, Pierre Fernbach, Nicolas Mansard, Michel Taix, and Andrea Del Prete. Sl1m: Sparse l1-norm minimization for contact planning on uneven terrain. *arXiv*, Sep 2019. arXiv:1909.09044 [cs].

[98] Sung-Joon Yoon and Baek-Kyu Cho. Strategies for generating footsteps of biped robots in narrow sight. *Sensors*, 2022.

[99] Jerry Pratt. Toward humanoid avatar robots for co-exploration of hazardous environments. https://youtu.be/HefjKANiZx0, 2018.

[100] Federico L. Moro and Luis Sentis. Whole-body control of humanoid robots. In Ambarish Goswami and Prahlad Vadakkepat, editors, *Humanoid Robotics: A Reference*, page 1161–1183. Springer Netherlands, Dordrecht, 2019.

[101] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *International Journal of Robotics Research*, 33(7):1006–1028, Jun 2014.

[102] Joseph Salini, Sébastien Barthélemy, and Philippe Bidaud. Lqp-based controller design for humanoid whole-body motion. In Jadran Lenarcic and Michael M. Stanisic, editors, *Advances in Robot Kinematics: Motion in Man and Machine*, page 177–184, Dordrecht, 2010. Springer Netherlands.

[103] Francesco Romano, Andrea Del Prete, Nicolas Mansard, and Francesco Nori. Prioritized optimal control: A hierarchical differential dynamic programming approach. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, page 3590–3595, May 2015.

[104] Alexander Herzog, Ludovic Righetti, Felix Grimminger, Peter Pastor, and Stefan Schaal. Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 981–988, Sep 2014.

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

[106] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Resolved momentum control: humanoid motion planning based on the linear and angular momentum. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 2, page 1644–1650 vol.2, Oct 2003.

[107] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, page 2641–2648, May 2006.

[108] Alexander Dietrich, Thomas Wimbock, Alin Albu-Schaffer, and Gerd Hirzinger. Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom. *IEEE Robotics & Automation Magazine*, 19(2):20–33, Jun 2012.

[109] Michael Mistry, Forbes Ave, and Ludovic Righetti. Operational space control of constrained and underactuated systems. *Robotics: Science and systems*, 7:225–232, 2012.

[110] Michael A. Hopkins, Alexander Leonessa, Brian Y. Lattimer, and Dennis W. Hong. Optimization-based whole-body control of a series elastic humanoid robot. *International Journal of Humanoid Robotics*, 13(01):1550034, Mar 2016.

[111] Nathan D. Ratliff, Karl Van Wyk, Mandy Xie, Anqi Li, and Muhammad Asif Rana. Optimization fabrics. *arXiv*, Aug 2020. arXiv:2008.02399 [cs, math].

[112] Mandy Xie, Karl Van Wyk, Anqi Li, Muhammad Asif Rana, Qian Wan, Dieter Fox, Byron Boots, and Nathan Ratliff. Geometric fabrics for the acceleration-based design of robotic motion. *arXiv*, Jun 2021. arXiv:2010.14750 [cs].

[113] Karl Van Wyk, Mandy Xie, Anqi Li, Muhammad Asif Rana, Buck Babich, Bryan Peele, Qian Wan, Iretiayo Akinola, Balakumar Sundaralingam, Dieter Fox, Byron Boots, and Nathan D. Ratliff. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics and Automation Letters*, 7(2):3202–3209, Apr 2022.

[114] Oussama Khatib. Motion/force redundancy of manipulators. *USA Symposium on Flexible Automation*, 1990.

[115] Bojan Nemec and Leon Zlajpah. Null space velocity control with dynamically consistent pseudo-inverse. *Robotica*, 18(5):513–518, Sep 2000.

[116] Yukai Gong. *Feedback Control of Highly Dynamic 3D Bipedal Locomotion.* PhD thesis, University of Michigan, 2022.

[117] Yu-Ming Chen, Gabriel Nelson, Robert Griffin, Michael Posa, and Jerry Pratt. Angular center of mass for humanoid robots. *arXiv*, Oct 2022. arXiv:2210.08111 [cs].

[118] Kunal S. Narkhede, Dhruv A. Thanki, Abhijeet M. Kulkarni, and Ioannis Poulakakis. Overtaking moving obstacles with digit: Path following for bipedal robots via model predictive contouring control. *arXiv*, Jul 2023. arXiv:2308.00119 [cs].

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

[120] Markus Eich, Felix Grimminger, and Frank Kirchner. A versatile stair-climbing robot for search and rescue applications. In *2008 IEEE International Workshop on Safety, Security and Rescue Robotics*, page 35–40, Oct 2008.

[121] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, page 1620–1626 vol.2, Sep 2003.

[122] Jung-Yup Kim, Ill-Woo Park, and Jun-Ho Oh. Experimental realization of dynamic stair climbing and descending of biped humanoid robot, hubo. *International Journal of Humanoid Robotics*, 6(02):205–240, 2009.

[123] David Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, Dec 2014.

[124] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and J 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, Aug 2011.

[125] Xiao-Bing Kong, Ying-jie Chen, and Xiang-Jie Liu. Nonlinear model predictive control with input-output linearization. In *2012 24th Chinese Control and Decision Conference (CCDC)*, page 688–693, May 2012.

[126] Masoud Soroush and Hossein M. Soroush. Input-output linearizing nonlinear model predictive control. *International Journal of Control*, 68(6):1449–1474, Jan 1997.

[127] Reza Heydari and Mohammad Farrokhi. Model predictive control for biped robots in climbing stairs. In *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*, page 1209–1214, May 2014.

[128] Aaron D. Ames, Eric A. Cousineau, and Matthew J. Powell. Dynamically stable bipedal robotic walking with nao via human-inspired hybrid zero dynamics. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, HSCC '12, page 135–144, New York, NY, USA, Apr 2012. Association for Computing Machinery.

[129] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E.R. Westervelt, C. Canudas-De-Wit, and J.W. Grizzle. Rabbit: a testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, Oct 2003.

[130] Lars Grüne and Jürgen Pannek. *Nonlinear Model Predictive Control*. Communications and Control Engineering. Springer International Publishing, Cham, 2017.

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

[132] Miomir Vukobratović and Branislav Borovac. Zero-moment point — thirty five years of its life. *International Journal of Humanoid Robotics*, 01(01):157–173, Mar 2004.

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