Geometric Control and Learning for Dynamic Legged Robots

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Mechanical Engineering

Avinash Siravuru

B.S., Mechanical Engineering, VIT University, India
M.S., Computer Science , IIIT-Hyderabad, India

Carnegie Mellon University
Pittsburgh, PA

December 2019

To my grandfather.

Acknowledgements

I was told great mentors are those who show you where to look but not what to see. By that yardstick, I am extremely fortunate to have Prof.Koushil Sreenath as my advisor. His support and mentorship on issues both academic and personal is unparalleled. For that I am forever grateful. I would like to thank my committee members, Prof. Aaron Johnson, Prof. Burak Kara, Prof. Hartmut Geyer, and Prof. Katerina Fragkiadaki for their valuable feedback and support. I also thank all the Hybrid Robotics Lab members with whom I learned and collaborated all these years.

Thanks also to CMU MechE department, particularly Chris Hertz and Ed Wojciechowski, for their prompt responses whenever needed and always having my back while delivering on my departmental commitments. Finally, I would like to thank my family whose innumerable sacrifices and sustained support enabled me to realize my dream.

This work was supported by grants from Pennsylvania Infrastructure and Technology Alliance (PITA), Google, Berkeley Deep Drive (BDD), and NSF NRI (in particular, IIS-1834557 and IIS-1526515).

Abstract

Geometric Control for Dynamic Legged Robots -

Most successful dynamic walkers today use either decoupled forward and sideways models or use locally-defined Euler formulations to model the combined $3D$ dynamics. While decoupling may lead to stability conflicts, local parametrizations of the orientation dynamics trades-off dynamism and shortens the stabilizable range of motion. Through a rigorous empirical study of the $3D$ pendulum stabilization problem, we show that Euler-parametrization based orientation control in $3D$ requires greater input to stabilize on average, not just for large-error situations.

To resolve these issues, we present novel geometric bipedal robot models and design suitable geometric controllers by extending commonly used non-linear control design techniques to non-Euclidean Lie-group manifolds. The dynamics and control actions thus obtained are very compact, singularity-free, and more importantly, they naturally capture the inherent coupling between the rotational degrees-of-freedom.

The presented models are the fully-actuated Reaction Mass Biped (RMB) and the geometric Cassie robot model both evolving on $\mathbb{SO}(3)$ product manifolds. The RMB model uniquely allows for modeling of the torso with variable inertia. Additionally, in the Cassie robot model the dynamics are augmented to capture the transverse plane dynamics generated while riding a pair of Hovershoes. Further, the RMB dynamics are also discretized using variational principles. Finally, suitable geometric variational integrators for numerical integration of the RMB dynamics while preserving its manifold structure and conservation properties for long time-scales.

On the control design front, for fully-actuated models like RMB, we define geometric motion plans for walking on straight and curved paths along with suitable trajectory tracking controllers that afford almost-global stability guarantees. Motivated by these promising theoretical results, we leveraged the geometric model of Cassie to plan and control highly dynamic behaviors like turning in place on a 20 degree-of-freedom bipedal robot, Cassie standing on a pair of Hovershoes.

Learning for Dynamic Legged Robots -

Deep learning has been widely used to develop smooth control policies for robotic systems. In the field of bipedal locomotion, *gait libraries* are a powerful tool to update gait parameters step-by-step via interpolation to render the bipedal system approximately neutrally stable. A discrete set of optimal gaits in the gait library is generated offline through nonlinear trajectory optimization

using the full-order hybrid robot model and satisfies all the associated unilateral ground contact and friction constraints, joint limits and motor limits. However, the gaits are locally stable around the specific gait parameter choice. Moreover, the interpolation time-complexity grows *linearly* and gait library space-complexity grows *exponentially* with the number of gait parameters.

Considering these factors, we combine model-based gait library design and deep learning to yield a near constant-time and constant-memory policy for fast, stable and robust bipedal robot locomotion. To achieve this, we design a custom network, called *Gait-Net*, using an autoencoder-based architecture to jointly learn both a gait-parameter to gait mapping and a gait-parameter reconstruction mapping. The reconstruction mapping is used to assess the quality of the learned gait. It can also be provided to the high-level planner to search for alternate plans that can result in better quality gait predictions to ensure stable and sustained locomotion. We validated our Gait-Net performance on a high-fidelity physics simulator that is custom-built for the bipedal robot Cassie.

A popular application of gait libraries is to walk on discrete terrain where the robot has to constantly modulate its step length to accurately step on discrete footholds. In such scenarios, it is also very important to sense and estimate the distance to the next valid foothold apriori to leverage the gait library for step length modulation. A perception module can be developed for this task but it must be very fast at detection and accurate at localization. The latest deep-learning fueled advances in computer vision make this possible. However, these neural network models need a lot of data.

Generating large datasets for every possible locomotion task is impractical. Alternatively, a graphics simulator capable of generating photo-realistic images can be used to rapidly generate synthetic datasets with desired diversity in visual features to mimic real-world situations. We take the latter approach.

A convolutional neural network, called *SL-CNN* is designed for predicting step-length from a synthetic dataset of monocular images rendered from the robot's point-of-view. Further, the network architecture is customized to minimize the worst-case prediction error keeping in mind the safety-critical nature of the task. Finally, the visual simulator and estimator thus developed are integrated into the physical model of the robot and the gait-library-based controller to realize autonomous planar walking in simulation.

Contents

List of Tables

List of Figures

Part I

Preliminaries

Chapter 1

Introduction

1.1 Geometric Control for Dynamic Legged Robots

To build versatile robots suitable for operating in almost all environments designed by and for human motion, it is natural to lean towards bipedal robots or humanoids. A lot of prior research has made walking with two legs stable [2, 3, 4, 5], fast [6, 7] and very energy efficient [8, 9, 10]. Now, active attempts are focused towards extending these gains to unstructured terrain [11, 12, 13] and to other related locomotion tasks like running [14], climbing, turning, etc.

Humans have a remarkable ability of generating highly dynamic motions that are characterized by arbitrarily large swing and/or stance leg rotations, as shown in Fig 1.1. This is a very critical requirement if robots are to replace humans in disaster missions. However, in the DARPA Robotics Challenge (DRC), the biggest competition held to demonstrate disaster response, the performance of humanoid robots was underwhelming [15, 16]. It is interesting to note that, 10 of the 17 robot falls in [16] were sideways, highlighting the poor lateral stability properties of the current state-of-the-art. So far, true dynamic 3D coupling, that is inherent in human maneuvering, has not been fully extended to humanoids. Most humanoid controllers are designed for straight (planar) walking and allow for slow turns, to avoid significantly deviating from the planarity assumptions. This is due to their local and relatively small domains of convergence, and their sensitivity to kinematic singularities. Bipedal robots can potentially emulate human motion if we can develop controllers that are able to harness the inherent coupling between forward, leaning and turning motions, and possess the ability to recover from large orientation errors.

Figure 1.1: This sharp turning maneuver is an important part of tactical play in football. Its a very commonly seen skill in humans with varying degrees of proficiency.

1.2 Learning for Dynamic Legged Robots

1.2.1 Learning a Unified Walking Policy from Gait Libraries

Bipedal robot locomotion is an active area of research with several control design challenges like high degrees-of-freedom, nonlinear dynamics, underactuation, and persistent impacts. Despite the control complexity, they have many potential applications ranging from disaster response, warehouse automation, last-mile delivery, elderly care, etc. Unlike wheeled robots, for these tasks, legged robots can be more versatile as they are better equipped to accomodate terrain elements in urban settings that are customized for human usage like stairs, escalators, narrow corridors, sharp corners, etc. Therefore, it can be argued that by achieving robust legged locomotion, and integrating it with advances in manipulation and perception, we can offer a better value proposition for robot deployments in our homes, hospitals, office spaces, factories, etc. Control design to realize dynamic and versatile motions still remains a challenge for bipedal robots. Specifically, explicit modeling of ground contact is not possible. Even if the robot can be stabilized within stride (while stepping forward), the impact at the end of the step could potentially destabilize the robot. Further, any within stride perturbation or a drastic change in the desired velocity or steering rate post impact only aggravates this problem. However, on a positive note, we have made considerable progress in developing stable *limit-cycle walkers* ([7]). i.e., robots that can maintain the commanded velocity or step length, while also stabilizing and absorbing intermittent impacts. This periodic motion is called a *gait* and the velocity or step length parameters used to develop it are called *gait parameters*. The objective of this work is to leverage this prior work and enhance the ability of these limit-cycle walkers to smoothly transition between velocity-specific limit cycles to realize a continuum of velocities and step lengths. This is achieved via learning a smooth policy by regressing over the finite set of pre-optimized and stable limit-cycle policies. The resulting learned policy can be used by a high-level planner to realize full range navigation.

1.2.2 Deep Perception for Autonomous Walking

Human-sized dynamic bipedal robots, like ATRIAS [17], are expensive and cannot be used for data generation for long duration without risking some form of hardware damage, especially for the problem of walking on discrete terrain. Even tele-operation of the robot to correctly step on discrete footholds - for the purpose of data generation - is very hard. Therefore, a realistic visual simulator is desired to rapidly generate synthetic data and train the robot to estimate key gait parameters on that. Additionally, similar to how we managed to narrow the gap between robot motion as indicated by a physical simulator and when tested on the real robot, we also wish to narrow the gap between predictive performance of the deep perception model in the visual simulator and the real world. Consider the simulated stepping stones terrain shown in Fig. 1.2. The objective is to train the deep visual perception model on such synthetic images but guarantee operable performance when tested on Fig. 1.3 assuming camera intrinsic and extrinsic parameters are consistent across the two settings.

Figure 1.2: A popular discrete terrain example is stepping stones [1] most commonly seen while crossing rivers, swamps, creeks, etc.

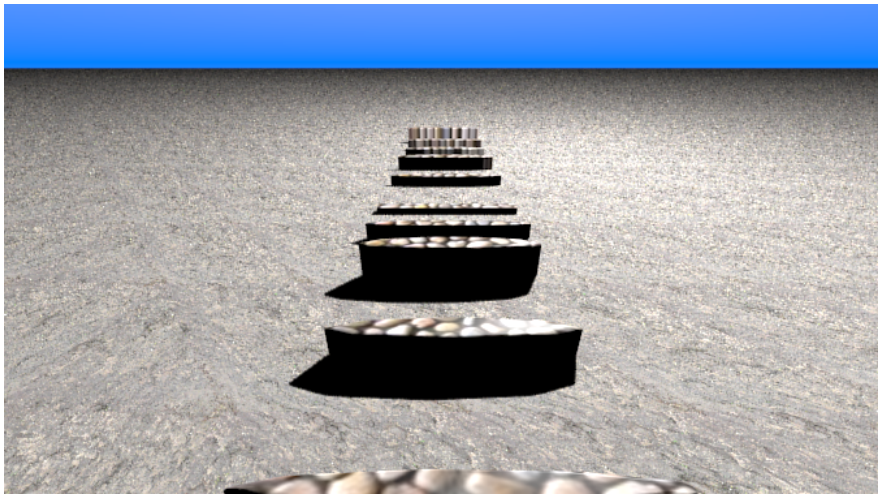


Figure 1.3: An example simulated stepping stones terrain.

1.3 Contributions

Considering the above-mentioned research objectives, this dissertation has made the following contributions:

1.3.1 Geometric Control for Dynamic Legged Robots

1. Empirical Evaluation of Euler-based and Geometric Controllers for 3D Pendulum Stabilization:

Pendulum dynamics are widely utilized in robotics control literature to evaluate novel control design techniques. While exhibiting many features commonly seen in real-world nonlinear systems, they are still simple enough to allow deeper analysis, quick prototyping and benchmarking. In this work, we study the impact of pendulum attitude parametrization on the nonlinear control performance. Mainly, for orientation control problems, we show that global coordinate-free formulations of dynamics and control are not only singularity-free but they are also more input-efficient. They are able to leverage the inherent manifold-space curvature and flow along geodesics for efficient stabilization. We validate this empirically by running over 700 stabilization simulations across the full configuration space of a $3D$ pendulum and compare the performances of the geometric and Euler-parametrized controllers. This work was presented in [18] and yet to be published.

2. Geometric Modeling and Control:

   - A novel reduced order model, called *Reaction Mass Biped*, is proposed. Unlike other popular models like LIPM, SLIP, etc., the RMB explicitly considers a variable inertia torso and models leg inertia. A Hybrid Geometric Model is developed using variational principles directly on the configuration manifold of the robot. The dynamics are said to be coordinate-free and have no singularity issues. Assuming full actuation, a trajectory tracking controller is developed for walking and turning with almost-global stability properties. Its worth noting that the controller can track circular walking trajectories with torso lean angles going up to 30 degrees. Note that, preliminary results of this work were reported in [19]. This dissertation adds to the original contribution in the following ways: First, the controller's working performance is more critically analyzed by enforcing unilateral ground contact and friction cone constraints at the stance foot.
   - A new Geometric Model is developed for the Cassie-Hovershoes multimodal robot [20] to plan and control dynamic motions like turning in place. Note that, this is a challenging maneuver to execute using a high center-of-gravity bipedal robot standing on Hovershoes. Essentially, the system is a like a pair of $3D$ cart-poles connected by the pelvis. The robot is only *dynamically stable*. Motion plans are devised in the transverse plane for the turning in place and geometric controllers are designed to stabilize the erect robot pose while executing dynamic transverse plane motions. These controllers are numerically and experimentally realized.

3. Geometric Variational Integrator for RMB:
   A coordinate-free discrete mechanical model of the RMB is developed using variational principles. Additionally, the discrete model is used to build a Geometric Variational Integrator (GVI). GVIs preserve important mechanical properties like energy conservation (for conservative systems), momentum conservation (when symmetries exist), while ensuring that the dynamics evolve on the configuration manifold of the system. This structure preserving property is also useful while building controllers using energy-like quantities like Lyapunov

functions, as shown in this work. Only through the use of GVIs preservation of the above-mentioned structures or properties can be guaranteed for long simulation times. This work has been published in [21].

1.3.2 Learning for Dynamic Legged Robotics

1. Learning a Unified Walking Policy from Gait Libraries:
   A data-driven smooth policy, called *Gait-Net*, is developed to generate gait parameter based walking plans for a bipedal robot called Cassie. The policy is faster and memory-efficient while maintaining performance better than or comparable to the currently used interpolation-based techniques. The network was trained to jointly learn the pre-optimized control parameters and to reconstruct the input (gait parameters) using an autoencoder-based architecture. The input reconstruction error can be used to evaluate and improve the quality of gait predictions at test-time. This work was presented in [22] and yet to be published.

2. Custom Deep Perception Module for Step Length Prediction:
   In this dissertation, we developed a custom deep neural network to estimate the step length (distance to the next stepping location) using a single camera image that is obtained at the beginning of each step. Detecting footholds and estimating distance is a classic object localization problem similar to object grasping in robotic manipulation, however in the case of locomotion there are additional challenges due to the time-critical and safety-critical nature of the problem. Failures can lead to very adverse consequences and there is very slim scope for recovery as the robot is operating in a dynamic regime around an unstable equilibrium. The deep neural network was customized keeping these challenges in mind and with an objective to minimize its worst-case performance. This work has been published in [23].

Chapter 2

Literature Review

Wheeled robot locomotion remains the most popular choice for most autonomous urban navigation tasks, with famous instances like self-driving cars [24] in outdoor, and personal robots like HERB [25], Pepper [26], etc. in indoor settings. However, usage of these robots is limited to flat-surfaced environments. For traversing on non-flat surfaces - that include common indoor settings like stairs, stages, etc. and outdoor settings like hills, rubble, stepping stones, etc. - legs are a superior choice as a locomotion mode. This has resulted in the emergence of an active subdomain in robotics for legged robot locomotion research. Within this sub-domain hexapedal (or) six-legged, quadrupedal (or) four-legged and bipedal (or) two-legged robots are more dominant.

Quadrupeds and Hexapeds strike a sweet balance between stability and terrain-adaptable mechanics. To further elaborate, stability comes from a much larger contact area and leg mechanics allow stepping over obstacles and even climbing in some instances. These features induce natural robustness to disturbances during unstructured terrain navigation [27], making them ideal for load-carrying or exploratory applications. Popular examples are RHex, BigDog, Spot, etc. Unfortunately, all the limbs of a quadruped are invested in locomotion, and therefore, they cannot be used for doing additional tasks (manipulation, brachiation, etc.). Attempts are being made to equip them with manipulators. However, even a simple task of opening a door can be arduous for such a robot. This was one of the prime reasons for apes to graduate from quadrupedal to bipedal locomotion in nature.

Note that, freeing some limbs for doing additional tasks comes at a cost i.e., a) fewer limbs implies smaller area of contact, therefore less robustness to disturbances, and b) the body center of mass goes up and increases sensitivity to terrain disturbances. This makes bipedal locomotion a much more challenging problem. It is worth pointing out though that there are some locomotion advantages to bipedalism. Stepping on discrete footholds with very small surface areas is easier with two legs. These are most commonly seen when crossing rivers, creeks, etc., or while climbing hills and mountains.

2.1 A Brief Introduction to Bipedal Robot Locomotion

Bipedal locomotion mechanics and control has been an active area of research across the robotics and bio-mechanics communities over the last half century and many valuable scientific and prac-

tical insights gained thereof have inspired the ongoing work presented in this thesis. An important consequence of understanding bipedal mechanics and control is the opportunity to apply them in the development of exoskeletons ([28, 29]) for persons suffering from locomotor impairments.

In order to design mechanisms or control techniques, we need to first understand the underlying physics in walking. Suitable physical models need to be devised that capture all the necessary degrees-of-freedom and kinematics that characterize walking motion. This is discussed in detail in the next sub-section.

2.1.1 Types of Bipedal Robot Models

Physical modeling of bipedal robot can be studied through two lenses namely a) Lens of Complexity and b) Lens of Parametrization. Next, we discuss each type in more detail.

Lens of Complexity

Bipedal robots are very high degree-of-freedom systems, have non-trivial mass distribution, involve intermittent to prolonged phases of under-actuation, and are more sensitive to falls (and physical damage) than most other robot classes. Therefore, even if accurate modeling of such systems is possible, understanding its highly complex dynamics itself is very difficult, let alone designing controllers for it. When control design is possible, testing its performance across a range of terrain interactions could offer another set of challenges. As an alternative, several simpler reduced-order models are proposed in literature. Through these reduced-order models, we can keep the focus of control design on particular elements of the model that are directly responsible for walking and nullifying (either by explicit omission or through separate control action) unrelated model elements. This way, a spectrum of bipedal robot models exist that can be broadly divided into two types: (a) Reduced-order Robot Model, (b) True (or) Full Robot Model.

Reduced-Order Models: Since legs are primary elements responsible for generating walking motions, most reduced order models are just physical descriptions of robot legs that either have no torso or assume the torso as a point-mass fixed at the hip. The simplest reduced-order model is the Linear Inverted Pendulum Model (LIPM) [3] where the leg itself is assumed to be massless with all the mass concentrated at the hip. Additionally, the dynamics is assumed to be linear. Here, we assume the stance leg to be pinned to the ground and behaving like an inverted pendulum. The objective is to model walking as the transfer of hip mass from one location to another in order to make the robot take a step. This motion is called a stride. Assuming nonlinear dynamics for the stance leg instead gives the Inverted Pendulum Model (IPM). Now, to model foot placement, one can stick a pendulum at the hip to describe the swing leg evolution. Since the legs are assumed to be mass-less, at the end of the step, one can simply switch the legs, and with proper bookkeeping of the impact effect, the same LIPM or IPM model can be used to describe the motion of the other leg. Walking is then simply the exchange of leg behaviours from inverted pendulum to pendulum on repeat. Now, more complexity is added by assuming legs to have mass concentrated at their respective COMs and torso mass concentrated at the hip. This model is called a Compass-Gait Biped (CGB) [30]. Another popular model is the Spring Loaded Inverted Pendulum Model (SLIP) [31] which adds spring compliance to legs. This is a very useful abstraction to describe running motion and also to model fast walking using knees (spring compression is mapped to knee-joint compression). Other reduced-order models include three link walker, five link walker,

etc., where other links like torso, shin, thigh are also explicitly modeled as pendulums. Finally, for planar walking studies, these links are assumed to be in $2D$ and for spatial studies, they are assumed to be in $3D$.

Full-Order Models: As we mentioned earlier, full robot models feature all the elements contained in the actual robot - legs with non-trivial mass distribution, torso, hands, etc. Examples include Rabbit [32], ATRIAS [33, 34], Cassie , ATLAS [5, 4], VALKYRIE , etc.

***Remark* 2.1.1.** *There are many instances where reduced-order models, which were primarily conceived for modeling and analysis, have also been built physically. We call any such instance as a reduced-order model realization. Popular reduced-order model realizations are McGeer's Walker [35], MIT Toddler [8], Cornell Ranger [10], among others. It is worth noting that, in this case, there is no distinction anymore between the reduced-order model and true robot model. These robots are primarily designed to experimentally validate reduced-order models and characterize their behavior more rigorously and thereby inspire future humanoid design.*

Lens of Parametrization

Conventionally, the orientations of robot links used in bipedal robot models are defined using Euler angles. It is a convenient and very useful design choice to describe and analyze the diverse behaviors exhibited by these models. This choice allows one to fully utilize the wealth of mathematical techniques from linear algebra, functional analysis and non-linear control developed to be applicable in Euclidean spaces (also called flat spaces), which are denoted by $\mathbb{R}^n$, where n is the robot's degrees-of-freedom. We call the models using Euler angles to define their orientations as Euler Models. All the robot models described in this literature survey (except where described as otherwise) are Euler Models. Despite their immense advantage, there are some limitations to this choice of orientation parametrization. Unlike linear motions, angular motions are not flat. They can be flattened in a closed range of operation. This restriction is justified mostly because of mechanical constraints (for example, knee joint only folds inwards, torso cannot rotate through the legs, etc.) that come from the design itself. However, imagine the case of a whirling ballerina (while performing a pirouette for example) who is turning on one foot. Such motions are characterized by the property of returning to their initial positions after completing a cycle. Unlike linear motions, rotations live in space that is compact and cyclic. This space (also called a manifold) is non-euclidean and attempts to flatten them always comes at a price, called singularity. This makes the design and control of such motions very difficult, if not impossible, on euclidean manifolds. Additionally, from a more practical standpoint, the equations of motion for Euler Models, while easy to derive, become very complex and cumbersome (filled with numerous trigonometric expressions) to visualize and intuit for non-trivial robot models.

These problems inspire us to introduce Geometric Models for bipedal robots. We define Geometric Models as those using not Euler angles but angle-free definitions of link orientations. Instead of using Euler angle based parametrizations, the links orientations are defined in their native manifolds i.e., either on S2 or SO(3). These manifolds are called Lie Groups . This choice makes the dynamics singularity-free and the equations of motion become very compact and the inherent dynamical couplings are more easily discernible. To obtain these models, we leverage recent advances in differential geometry, lie group theory and geometric mechanics. More details

on this will follow in Section 2.2 where we review the general motivation and application of geometric models and control in robotics. Finally, based on parametrization, bipedal robot models can be classified as: (a) Euler Model, (b) Geometric Model. Having outlined the various types of bipedal robot models, in the next subsection, we similarly describe the various prevalent control paradigms used in bipedal robot locomotion research.

2.1.2 Control Design Types for Walkers

Broadly, three prevailing schools of thought exist in bipedal locomotion control design as listed below:

- Biomechanics-based
- Learning-based
- Physics-based

Biomechanics-based control design assumes either the actuator to be modeled after the human muscle-model [36, 37], the control logic to be modeled after biologically motivated neuronal structures like Central Pattern Generators [38, 39], or both. Next, in learning-based control, the physical robot model is considered either inadequate or too complex for policy design using traditional approaches. Therefore, the control design in this case assumes no prior knowledge of the model (called model-free). Popular learning-based controllers include [40, 41, 42, 43]. Finally, we have physics-based control design where the robot's dynamics knowledge is exploited for control design purposes. The focus of this thesis is on Physics-based controllers. Next, we further subclassify physics-based controllers and briefly comment on their relative merits and demerits. For a higher level summary-view of the space of bipedal robot modeling and control, see Fig. 2.1. We first begin by classifying physics-based Bipedal Walkers [1] as either being a) Static or b) Dynamic. Later, we classify Dynamic Walkers as either being a) Passive or b) Active. Going further, Active Dynamic Walkers can be further resolved into being either a) Fully-actuated or b) Under-actuated. Finally, Under-actuated Active Dynamic Walkers can use either a) Euler-Parametrized physical robot model or b) Geometric physical robot models. For a pictorial overview of this classification, see Fig. 2.1.

Note that, the focus of the first part of this thesis will be on control design, and for that purpose, we use a) Geometric physical models. However, for the second part of this thesis, where the primary focus is on designing a perception module for Under-actuated Dynamic Walkers, we will build on existing prior work on b) Euler-parametrized Under-actuated Dynamic Walkers. We will again remind the readers of this distinction in the respective chapters for convenience. For the sake of brevity, we will only summarize at a higher-level the walking principles used in Static Walkers, Passive Dynamic Walkers and Fully-actuated Dynamic Walkers. We point interested readers to the attached references to obtain a more detailed understanding. However, for Under-actuated Dynamic Walkers, we shall also review the related control design principles to sufficient detail to aid in fully appreciating the novel modeling and control design choices presented in this thesis.

[1] Throughout the rest of this thesis we will use the term Walker as a short form for a legged robot designed to walk. In the same spirit, a legged robot designed for running will be termed a Runner. Here, design implies mechanism and/or control design.

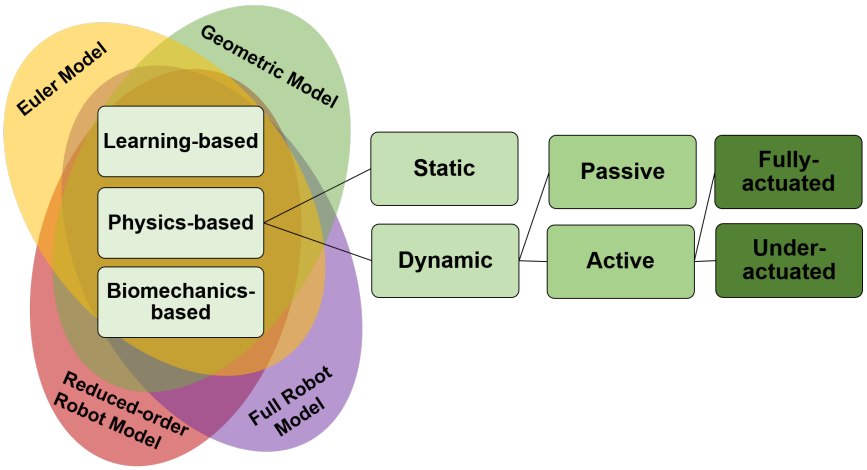


Figure 2.1: The classification of prior work in bipedal locomotion modeling and control research is pictorially summarized here.

2.1.3 Static Walkers

Earliest attempts to achieve walking on bipedal robots took a more conservative approach. The objective in this class of control design focuses on generating walking motion with the robot kept upright at all possible times, even if it were stopped mid-way. This implies the walking motion is slow and it generates minimal accelerations and body moments in the process, and parallely, all the external forces acting on the robot are balanced. This results in static stability (it is quasi-static in practice) and therefore the term *Static Walker*.

There is a key requirement to ensure this form of quasi-static motion. The projection of the center-of-mass (COM) of the robot to the ground must always remains within the support polygon. The support polygon is defined as the convex hull formed by the vertices of the robot's foot that is always connected flatly to the ground, also called stance foot. Consequently, with a bigger foot the COM of the robot has a larger range of motion possible without violating the static stability constraints. This point of projection of the COM to the ground is called the Zero Moment Point (ZMP) [2]. The implementation details are comprehensively reviewed in [143], where they conclude that as long as the ZMP stays within the support polygon, the robot is stable. This technique has been applied successfully to many popular humanoid robots like Honda's ASIMO [44], Sony's QRIO [45], HRP [46], and Toyota's humanoid [47] among others [48].

The advantage of this approach is that its very intuitive and allows the use of full-actuation (including the ankle joint as the foot is locked to the ground) at all times. However, the disadvantage is the need for big and heavy feet (to offer more robustness), slow walking (to remain quasi-static), and as a consequence, have very low energy efficiency [49]. Finally, later studies showed that the need of enforcing ZMP criterion is neither sufficient [50] nor necessary [51, 7] for ensuring stable walking.

2.1.4 Dynamic Walkers

As described in the previous section, static walkers walk slowly, keep inertial effects to the minimum and ensure that the CoM is carefully transferred from one leg to another while ensuring it always remain within the support polygon. Bent knees, small step-lengths characterize this form of walking (also popularly called *robotic walking*).

To obtain better energy efficiency and faster motion, some of the restrictions like the need for static stability need to be relaxed. The center-of-mass of the robot is allowed to go beyond the support polygon (allows for longer step lengths) and even generate inertial forces/moments (therefore called *dynamic*) and take advantage for them for quicker motion. Walking in the dynamic sense is redefined as *stepping quickly enough to catch the robot from falling* and being able to repeat this feat indefinitely. Remarkably, this dynamic nature of walking was first demonstrated using robots that had no actuation! They were called Passive Dynamic Walkers and they are summarized next.

Passive Dynamic Walkers

In his seminal work [35], Tad McGeer showed a bipedal robot walking down a shallow slope with no actuation at all. This demonstrated the most energy efficient walking ever possible (using only potential energy). This was subsequently extended to a three-dimensional walker with arms [49]. Despite the great energy efficiency and faster walking, passive walkers fail on one count. They are very sensitive to perturbations and they can't be deployed on flat ground where there is not potential energy change. While the former still remains an issue, the latter was actively researched within the past two decades. Several smart control techniques emerged that attempted to inject energy into the robot through minimal actuation.

Active Dynamic Walkers

Active dynamics walkers borrow from efficient walking principles from passive walkers but don't suffer from the limitation of needing a slope for walking. They inject required amount energy using actuators. Popular examples of Active Dynamic Walkers include Ranger [10], MIT Toddler [8, 40]. The concept of injecting exactly the amount of energy (potential energy difference between flat-ground and slope) required to execute sloped walking on flat ground is called Controlled Symmetries [52].

Fully-actuated Dynamic Walkers: A totally different way to achieve active dynamics walking is through the use of reduced order [4, 5] or full-robot models [53, 54] and optimizing over the nonlinear dynamics of the robot to discover stable walking trajectories. These robots are faster than Static Walkers and more energy-expending than active walkers inspired from passive dynamical techniques. However, they are comparatively more robust to perturbations. Unfortunately, this control design methodology assumes full actuation. However, several practical walking situations exist (like toe-off) where the robot is under-actuated at the foot and has not control over the toe's rolling motion. Any perturbation of the above-mentioned Walkers in such situations will invariable lead to a fall.

Underactuated Dynamic Walkers: In order to address this issue, alternately, robot models alongwith some under-actuation (specifically at the foot) are considered for control design. Unlike the previous Walkers, these robot models are no longer fully controllable but only stabilizable [51]. Additionally, using reduced-order models might not be a good idea in control design in this case as

the strong stability arguments that can translate from reduced-order to full robot model cannot be made in the face of under-actuation. In order to address this issue, instead of simplifying the robot model, virtual constraints [55, 12] may be enforced on all the actuated degrees of freedom. These virtual constraints control the motion of the joints to evolve along specific trajectories that lead to a non-trivial walking motion. Finally, as the last piece in this puzzle, the under-actuation can be used to our advantage to serve as a timing variable for this virtually constrained evolution of actuated joints. This way, reflexes are naturally encoded into the system's dynamics. Finally, to also achieve energy efficiency, the virtual constraints design itself is posed as an optimization problem whose objective is to minimize energy. Popular underactuated dynamic walkers include MABEL [7, 56], ATRIAS [12, 57], DURUS [58, 59], etc. The robot models and controllers studied in this work are Under-actuated Dynamic Walkers. In the next chapter, we will go into the details of control design for this class of robots.

2.2 Geometric Modeling and Control in Robotics

The ATLAS robots in DRC used popular reduced-order models like ZMP [2] and LIPM [3] to develop optimal walking policies [5] for slow walking. On the other hand, there exist dynamic walkers like RABBIT [32], MABEL [6, 7], AMBER [60], demonstrating more versatile and faster walking gaits. However, this has been demonstrated mostly for planar robots.

In [61], frontal plane stabilization during walking is presented, decoupled from the sagittal plane dynamics for modest roll angles and angular velocities. Other related works include stabilization of 3D walking using techniques based on controlled symmetries and Routhian reduction [62, 63], and on hybrid zero dynamics [51, 64]. These methods have been extended to yaw steering of 3D robots [65]. None of these techniques have been provably extended for large angular deviations in leg position.

Parallely, great progress has been made in the application of geometric control techniques to the UAV load carrying problem [66, 67], spacecraft attitude control [68, 69, 70]. Exponentially stable tracking controllers have been built to cancel arbitrarily large attitude errors in the load and UAV positions. This was achieved by the application of variational mechanics principles to their dynamic modeling and control design [71, 72]. Human-like maneuvering, push recovery and periodic gaits with large domains of attraction in 3D, with under-actuation, still remain as open problems [51].

2.3 Deep Perception in Robotics

One of the key challenges for building robust autonomous navigation and manipulation systems is the development of a strong intelligence pipeline that is able to efficiently gather incoming sensor data and take suitable control actions with good repeatability and fault-tolerance. In the past, this was addressed in a modular fashion, where specialized algorithms were developed for each subsystem and integrated with fine tuning. More recent trends show a revival of end-to-end approaches that learn complex mappings directly from the input to output by leveraging large volume of task-specific data and the remarkable abstraction abilities afforded by deep neural networks. Large amounts of data available for network training and significant advances in parallel computation hardware helped accelerate this effort. Before delving into deep perception for robotics, we briefly

describe the emergence of deep learning as a very powerful computational abstraction tool and review its impact in the related domains of Computer Vision and Graphics.

The fields of Computer Vision and Graphics where the greatest beneficiaries of deep learning advances for their pattern recognition tasks. Instead of painstakingly hand-engineering features of interest to reduce computational overhead for traditional machine learning techniques, now with deep learning, the pipelines could be automated and data could be used to learn features of interest automatically and more appropriately to the task at hand. Object Classification [73], Scene Understanding , Human Pose Estimation [74], learning dynamics visually [75, 76], etc., are all possible today, and in some cases, even exceed human performance levels. Computer Graphics [41], Games [77] and Computer-aided Design (CAD) [78, 79][41, 151] have all seen improvements in the learning rich, high-dimensional models which were never before possible.

The dramatic gains in research outcomes through the use of deep learning techniques in related domains led to a natural surge of interest in robotics as well. From learning end-to-end visuomotor policies for object manipulation [80], to learning to fly UAVs in cluttered environments [81], or in self-driving cars [82, 83], deep learning is impacting all major robotics sub-domains. In humanoid robotics also, CNNs were used for innovative applications like surface friction estimation from images, to help in slip prediction [84]. However, it is very challenging to build end-to-end fully data-driven policies for stable and safe limit cycle walking, let alone on discrete terrain. Keeping this in mind we limit the use of data-driven visual perception modules for suitable observer design only in this work.

2.3.1 Simulation-to-Reality in Robotic Vision

It is a standard practice in robotics to build simpler but characteristic models of real systems and/or settings, and extensively test potential solutions (control policies) before deploying in the real world. This helps in quickly iterating through multiple potential solutions without wearing out the robot or risk getting into unforeseeable accidents (when the robot or environment models used are not fully accurate). The DARPA Virtual Robotic Challenge [85] is a classical example of this practice. Fueled by this need, researchers have developed several high-fidelity numerical simulators [86] that can pretty accurately model robot mechanics, robot-environment interaction (touch, grasp, collision, etc.), etc. With the advent of OpenAI-Gym [87], it is now also possible to rigorously test and improve data-driven physics models and control policies within the secure confines of a simulation. The current bottleneck in robotics is dearth of simulators that can simulate real-world scenes to photo-realistic detail in order to leverage the use of visual feedback to close the loop. Great strides have been made in Computer Vision in the last decade, and yet, very little has trickled down to enhance visual perception abilities of modern day robots. Recently, researchers began addressing this issue by building visually realistic simulators like Microsoft AirSim [88] for drones, Udacitys Self-Driving Car Simulator [89] for cars, etc. While these models are available to rapid development of visual perception modules the jury is still out on how well they can perform when tested in the real-world. Put differently, it still remains to be seen how much the reality gap can be narrowed using these visual simulators. A related question is, what can the designer do from his end to further this cause.

Typically in computer vision, in order to narrow this performance gap, a little amount of real-world data is used to modify the learned representation in a way that it can be reliably used in the

real-world. This process is called domain adaptation and it is an active area of research [90, 91, 92]. There were some attempts to extend this idea to robotics as well [93], where an inverse dynamics model was learned to adapt the policy, designed in simulation, to the real-world model in face of modeling error. Note that, real-world data is not used to learn the representation of interest but just to account for discrepancy in the data used for training and testing. For example, in a classical domain adaptation problem, one attempts to learn face detection in simulation and uses additional real-world data purely to adapt the learned face detector to also work with the changes in lighting, color and texture seen in the real-world. The gist here is that understanding *what is a face* is a harder problem, needing more data, and therefore relegated to simulation. However, learning the color distribution changes, lighting changes, etc., is easier and needs lesser data.

Alternately, there is also an active exploration of mixing simulated and real-world data [94] to eliminate the additional domain adaptation step. The deep network's heavy data demand is fractionally addressed using real-world data and mostly through rapidly generated synthetic data. However, both these approaches still need a decent amount of real-world data, in the order of tens of thousands instead of hundreds. This is still hard to generate for some robotic applications. For the case of a bipedal robot stepping on stones, it is not even clear how to generate this data. First of all, meticulously setting up thousands of stones at specific measured locations is very laborious and impractical. Even if that was possible, it is hard to tele-operate the robot and guarantee it steps on the stones.

Recently, there is an active interest to explore the possibly of learning purely in simulation and check if it can be applied to real-world tasks directly without any domain adaption or mixing of real and simulated data. This is aggressively being pursued in the context of autonomous driving to offer smaller players a quick entry point even if they don't have a huge corpus of driving data available. In both [95], researchers showed that this is indeed possible given a photo-realistic visual simulator and large enough dataset. In contrast, more recently, researchers working on robot manipulation tasks [96] and a visually-servoing UAV [81] have shown that it is possible to go from simulation-to-reality without needing photo-realism as well! In order to do this, in the former work, they introduced a new technique called domain randomization whose working principle is to randomize all visual properties while creating the synthetic dataset. Here, the objective is to make any real world instance of the scene as just a sampling from this vastly randomized distribution. This technique offers a practical solution to the problem of building a deep perception module for a bipedal robot to walk on discrete terrain. However, this is a safety-critical task, and unlike object manipulation, failure is not an option even for once. Therefore, additional customization of the presented deep networks needs to be carried out to adapt to our safety-critical requirements.

2.4 Perception in Legged Locomotion

Perception for bipedal locomotion primarily focused on foot-step planning for statically stable or linear dynamical model based walkers. Usually, LIDAR-camera combination is preferred in this case. Accurate high resolution depth data obtained from LIDAR is used for safe footstep detection and planning [97, 98]. Unlike these walkers, dynamics robots have point feet, move much faster and therefore need faster execution and the ability to pick footholds of any size. This makes the search problem harder on the full 3D map.

Visual Perception for Walking

Visual Perception, as opposed to other forms of sensing, is cheap, readily available and very information dense. However, the space is very high-dimensional and computationally expensive to operate in. Moreover, information is noisy due to changes in lighting conditions, weather, time of the day, etc. Yet, the advantages of visual perception are far more significant.Conventionally, robot perception involves parsing the entire scene, labeling objects of interest, and feeding this information for planning and control. This technique, called *Mediated Perception*, is used extensively in self-driving cars [99] with some extensions to legged robots [100]. Usually, this technique is built over LIDAR and Stereo Camera datasets, requiring large processing overhead. For some tasks, on the other hand, information necessary for taking a suitable control action can be compacted to a few critical state and environment descriptors, called affordances, and doesn't require knowledge of the entire scene. In Robot Perception, the philosophy of converting a visual input directly into actionable affordances is called *Direct Perception* [83] and this is inspired from popular human psychological studies [101]. As the number of tasks increase and the decision-making time drops, searching the entire scene for all tasks is expensive and trade-offs are inevitable. Reactive vision-in-the-loop control for sub-tasks, wherever possible, reduces overhead on higher-level planners and injects more dynamism into the system. The objective of this work is to serve this need for the sub-task of walking safely on discrete terrain. Searching over very high-dimensional spaces simultaneously for all the tasks becomes intractable and is also arguably unnecessary. Consider the situation of full-fledged autonomous dynamic robot walking on a discrete terrain with speeds in excess of 0.6 m/s. The robot needs to decide exactly where to step, monitor it's eventual target position, constantly remain mindful of obstacles and avoid them, etc. Significant computational overhead needs to be invested in target tracking and collision avoidance, leaving lesser time and resource for accurate foothold detection.

In fact, experiments with humans and cats have shown that during over $50 - 60\%$ time of the gait cycle, gaze is invested in target detection and the higher-level spatial navigation problem [102]. When required to walk on complex terrain requiring accurate foot placement, humans operate with intermittent visual samples of the foothold location and use the information in a feed-forward manner to adjust step length just $1 - 2$ steps a priori [102, 103]. For walking on discrete terrain, the key finding here is that, instead of active modulation of gait through visual feedback during the entire stance phase, humans prefer to adjust their gait one-two step ahead using an intermittent visual preview and execute an energy-efficient ballistic motion [104]. If the foothold remains constant, continuous visual feedback may even be unnecessary. Our work is inspired by these biological findings and we wish to elicit similar behavior from the dynamic bipedal robot ATRIAS.

Vision-in-the-loop walking with gait adjustment (comparable to our approach) was implemented on a Quadruped in [100] to operate on steep slopes and dense vegetation. However, the problem of discrete terrain is not addressed. We believe our solution is complimentary to their effort and a combined solution could pave the way for true rough terrain navigation.

2.5 Summary

An attempt has been made here to cover a wide range of topics relevant to this work spanning major domains like robotics, control theory, machine learning and computer vision. We believe

understanding and connecting suitable topics from across the board is not only useful but very important to fully appreciate the results shown in this work.

In the first section, we begin by introducing bipedal robots and highlight their importance in unstructured terrain navigation tasks often seen during search and rescue situations or space exploration. Going further, we summarize the various well-studied bipedal robot modeling choices and we broadly classify them through the lenses of complexity and parametrization choice. Next, we embark on a similar exercise but from the controller design standpoint. This classification is done in finer detail by introducing several sub-classifications that are characterized either by the amount of actuation available, extent to which the robot model is simplified, etc. We end this section with the brief review of under-actuated bipedal robot control using HZD-technique. We save the detailed exposition of this technique for the next chapter.

In the second section, we shift our attention towards geometric modeling and control and its applicability to robots. We begin by first highlighting limitations in existing Euler-based robot configuration parametrization. This was briefly explained in the context of bipedal robots in the previous section as well. Later, we review prior work which used geometric control to correct large attitude error, especially in UAVs. Finally, we connect it to bipedal robots again by referencing our recent work (RMB [19]) where these techniques were used.

In the last section, we review various perception techniques used in bipedal locomotion. In doing so, we focus our attention eventually to deep perception related work, and later, zoom out to more generally review the use of deep learning in robotics and computer vision. A good overview of these recent advances across other domains helps further motivate this particular perception module choice in our work. Finally, we address the need for large data and the issue with data-generation for the stepping on stones problem and thereby highlighting the need for a visual simulator. Now, in relation to this proposition, we review recent advances in the realm of simulation-to-reality transfer learning in robotics. We finally conclude by indicating the most suitable sim-to-reality transfer learning technique for this work.

Chapter 3

Background on ATRIAS and Cassie Robots

In Part III of this thesis we will introduce two learning techniques: (a) in Chapter 8, we learn a unified walking policy from a dataset of gaits generated for the Cassie Robot, and (b) in Chapter 9, we automate discrete terrain walking using a deep perception module that detects upcoming footholds to enable accurate step length regulation. Here, we used walking gaits developed for the ATRIAS robot. The learning techniques in both these chapters use prior work on modeling and control design of these two robots. In this chapter, we briefly summarize them to provide the full picture. However note that ATRIAS robot model and gait library design is summarized before that of Cassie. This is in the increasing order of model complexity as a planar model used for ATRIAS whereas the $3D$ model is used for Cassie.

3.1 ATRIAS Robot

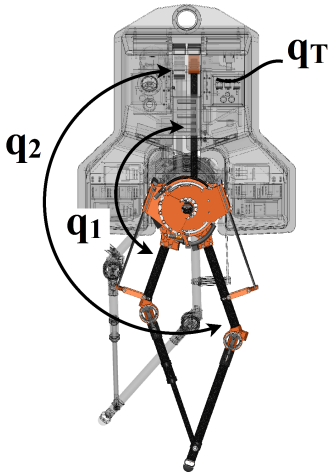


Figure 3.1: Biped coordinates. The world frame pitch angle is denoted by q_T, while (q_1, q_2) are body coordinates. The model is assumed left-right symmetric.

The bipedal robot shown in Fig. 3.1 is a planar representation of ATRIAS. Its total mass is

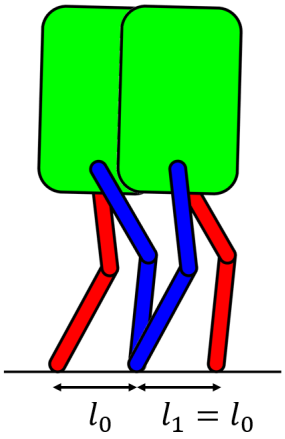


Figure 3.2: Periodic walking gait has the resulting step length (l_1) similar to the initial step length (l_0), or in other words $l_1 = l_0$.

63 kg, with approximately 50% of the mass in the hips and 40% in the torso, and with light legs formed by a four-bar linkage. The robot is approximately left-right symmetric.

The configuration variables for the system are defined as $q := (q_T, q_{1R}, q_{2R}, q_{1L}, q_{2L}) \in \mathbb{R}^5$. The variable q_T corresponds to the world frame pitch angle, and $(q_{1R}, q_{2R}, q_{1L}, q_{2L})$ refer to the local coordinates for linkages. Each of the four linkages are actuated by a DC motor behind a 50:1 gear ratio harmonic drive, with the robot having one degree of under-actuation. The four-bar linkage mechanism comprising of the leg coordinates (q_1, q_2) map to the leg angle and knee angle (q_{LA}, q_{KA}), as $q_{LA} := \frac{1}{2}(q_1 + q_2)$ and $q_{KA} := q_2 - q_1$. The state x denotes the generalized positions and velocities of the robot and u denotes the joint torques. A hybrid model of walking can be expressed as

$$\begin{cases} \dot{x} &= f(x) + g(x)u & x \notin \mathcal{S} \\ x^+ &= \Delta(x^-) & x \in \mathcal{S}, \end{cases} \tag{3.1}$$

where $\mathcal{S}$ is the impact surface and Δ is the reset or impact map. A more complete description of the robot and a derivation of its model are given in [105].

3.1.1 Gait Library Overview

Having described the dynamical model of ATRIAS in Section 3.1, we will now present the background on optimization for periodic gait design using virtual constraints and input-output linearization, a nonlinear feedback controller to exponentially stabilize the closed-loop system.

Periodic Gait Design Using Virtual Constraints

The nominal feedback controller is based on the virtual constraints framework presented in [106]. Virtual constraints are kinematic relations that synchronize the evolution of the robot's coordinates via continuous-time feedback control. One virtual constraint in the form of a parametrized spline can be imposed for each (independent) actuator. Parameter optimization is used to find the spline coefficients so as to create a periodic orbit satisfying a desired step length, while respecting physical constraints on torque, motor velocity, and friction cone. Since the gait is periodic, the initial

Table 3.1: Optimization constraints

Motor Toque	$\|u\| \leq 7$ Nm
Impact Impulse	$F_e \leq 15$ Ns
Friction Cone	$\mu \leq 0.6$
Vertical Ground Reaction Force	$F_{st}^v \geq 200$ N
Mid-step Swing Foot Clearance	$h_f\|_{s=0.5} \geq 0.1$ m

step length and the resulting step length must be the same (see Fig. 3.2). The optimizer used here is based on the direct collocation framework from [107], although other optimization methods, such as [108] or `fmincon` can be used as well.

The virtual constraints are expressed as an output vector

$$y = h_0(q) - h_d(s(q), \alpha), \tag{3.2}$$

to be asymptotically zeroed by a feedback controller. Here, $h_0(q)$ specifies the quantities to be controlled

$$h_0(q) = \begin{bmatrix} q_{LA}^{st} \\ q_{KA}^{st} \\ q_{LA}^{sw} \\ q_{KA}^{sw} \end{bmatrix}, \tag{3.3}$$

where st and sw designate the stance and swing legs, respectively, and $h_d(s, \alpha)$ is a 4-vector of Beziér polynomials in the parameters α specifying the desired evolution of the $h_0(q)$, where s is a gait phasing variable defined as

$$s := \frac{\theta - \theta_{init}}{\theta_{final} - \theta_{init}}, \tag{3.4}$$

with $\theta = q_T + q_{LA}^{st}$ being the absolute stance leg angle.

The cost function and constraints for the optimization are formulated as in [106] [Chap. 6.6.2], with the constraints given in Table 3.1 and the cost taken as the integral of squared torques over step length:

$$J = \int_0^T ||u(t)||_2^2 \, dt. \tag{3.5}$$

In addition to the above constraints, we also need to guarantee the periodicity in the gait:

- The initial state at start of the first step is given by $x = x_0^+$ with corresponding (initial) step length of l_0.

- The state at end of the first step (before impact) is given by $x = x_1^-$ with corresponding (resulting) step length of l_1.
- Impact constraints at the end of the step are enforced as $x_1^+ = \Delta(x_1^-)$.
- Periodic constraints are then enforced as $x_1^+ = x_0^+$, resulting in $l_1 = l_0$.

Here, the superscript '$+$' and '$-$' represent the state right before and right after the impact, and Δ is the reset or impact map from (3.1).

Input-output linearization

The optimization results in a desired walking gait encoded through $h_d(s(q), \alpha)$ in (3.2) and therefore our control goal is to drive $y(q) \to 0$. In our method, we use input-output linearization, a nonlinear feedback controller to enforce exponential stability for the system [109]. If $y(q)$ has vector relative degree 2, then the second derivative takes the form

$$\ddot{y} = L_f^2 y(q, \dot{q}) + L_g L_f y(q, \dot{q})\, u. \tag{3.6}$$

We can then apply the following pre-control law

$$u(q, \dot{q}) = u^*(q, \dot{q}) + (L_g L_f y(q, \dot{q}))^{-1}\, \mu, \tag{3.7}$$

where

$$u^*(q, \dot{q}) := -(L_g L_f y(q, \dot{q}))^{-1} L_f^2 y(q, \dot{q}), \tag{3.8}$$

and μ is a stabilizing control to be chosen. Defining transverse variables $\eta = [y,\ \dot{y}]^T$, and using the IO linearization controller above with the pre-control law (3.7), we have,

$$\ddot{y} = \mu. \tag{3.9}$$

The exponential convergence of the control output y then can be easily derived using PD controller:

$$\mu = -K_p y - K_d \dot{y}. \tag{3.10}$$

Having presented the background on periodic gait optimization using Hybrid Zero Dynamics and input-output linearization, we now introduce our proposed approach using 2-step periodic gait optimization to handle randomly-varying discrete terrain resulting in consecutive changes in step length and step height at each walking step.

2-Step Periodic Gait Design Using Virtual Constraints

Inspired by the main issue of step transition on stepping stones [110], we develop an optimization framework to design 2-step periodic walking gaits, taking into account not only the desired footstep location of the next step but also the current configuration of the robot. The method combines virtual constraints, parameter optimization, and an interpolation strategy for creating a continuum of gaits from a finite library of gaits. The notion of 2-step periodic gait means that the robot states are designed to be converge back to the initial condition after 2 walking steps. To be more specific, we will start off with the problem of changing step length only or walking on flat ground with varied step length.

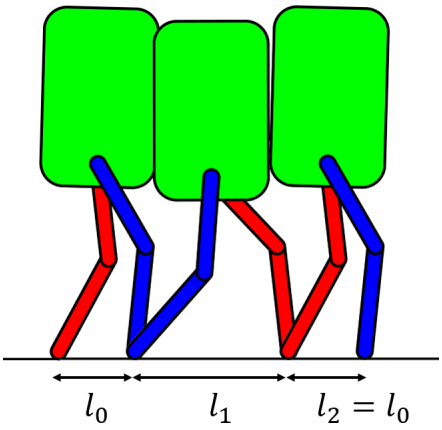


Figure 3.3: 2-Step periodic walking with changing step lengths only. The walking gait is 2-step periodic therefore the step length of the second step and that of the initial condition are the same ($l_2 = l_0$).

Changing Only Step Lengths

In the nominal problem of periodic optimization presented in Section 3.1.1, we need to optimize for only one walking step with the constraint on the resulting step length (l_1) to be equal to the initial step length (l_0) (see Fig. 3.2). For this problem, we use the same optimization framework discussed in 3.1.1, but we will optimize for 2 walking steps while following additional constraints that allows us to have different step lengths during transition (see Fig.3.3):

- The initial state at start of the first step is given by $x = x_0^+$with corresponding (initial) step length l_0.
- The state at the end of the first step (before impact) is $x = x_1^-$ with (resulting) step length l_1.
- Impact constraints at the end of the first step are enforced as $x_1^+ = \Delta(x_1^-)$.
- The initial state at start of the second step is given by $x = x_1^+$ with corresponding (initial) step length of l_1.
- The state at the end of the second step (before impact) is $x = x_2^-$ with (resulting) step length of l_2.
- Impact constraints at the end of the second step are enforced as $x_2^+ = \Delta(x_2^-)$.
- Periodic constraints are then enforced as $x_2^+ = x_0^+$, resulting in $l_2 = l_0$.

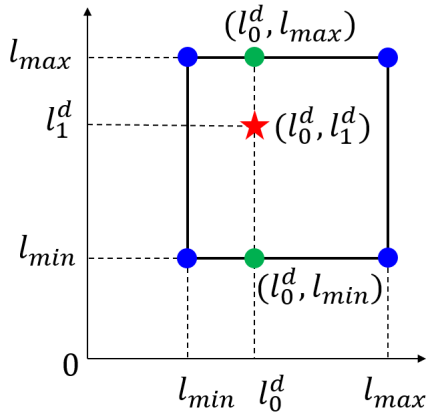


Figure 3.4: Gait interpolation for the problem of changing step length only.

The optimization problem is then used to generate a gait library with different values of l_0 and l_1. In this work, we optimize four different gaits corresponding to:

$$\begin{cases} (l_0, l_1) = (0.3, 0.3)\ m \\ (l_0, l_1) = (0.3, 0.7)\ m \\ (l_0, l_1) = (0.7, 0.3)\ m \\ (l_0, l_1) = (0.7, 0.7)\ m. \end{cases} \tag{3.11}$$

It is similar to precomputing four gait primitives corresponding to walking with small steps ($(l_0, l_1) = (0.3, 0.3)\ m$), switching from a small step to a large step ($(l_0, l_1) = (0.3, 0.7)\ m$), switching from a large step to a small step ($(l_1, l_0) = (0.7, 0.3)\ m$) and walking with large steps ($(l_0, l_1) = (0.7, 0.7)\ m$). Having a gait library with different gaits representing for some general motion primitives, we then do gait interpolation to get the desired walking gait with an arbitrary set of (l_0^d, l_1^d).

Let $\alpha(l_0^d, l_1^d)$ be the Beziér coefficients (defined in (3.2)) of the desired walking gait that has the initial step length l_0^d and the resulting step length l_1^d. If $l_0^d \in [0.3 : 0.7]\ m$ and $l_1 \in [0.3 : 0.7]\ m$, we will compute $\alpha(l_0^d, l_1^d)$ using bilinear interpolation of the coefficients from the four nominal gait parameters precomputed using optimization. A detailed explanation for bilinear interpolation can be found in [111] or Fig. 3.4. In this work, we use the MATLAB function "interp2" to implement the algorithm.

***Remark* 3.1.1.** *If $l_0 \notin [0.3 : 0.7]\ m$ or $l_1 \notin [0.3, 0.7]\ m$, we can use extrapolation to compute the gait parameters for the desired gait.*

The gait library and gait interpolation are used to update the walking gait for every walking step based on the desired footstep placement of the next step (l_1) and the current configuration of the robot (l_0). They are then incorporated using input-output linearization to control the robot to follow the updated walking gait. The closed-loop control diagram is shown in Fig. 3.5.

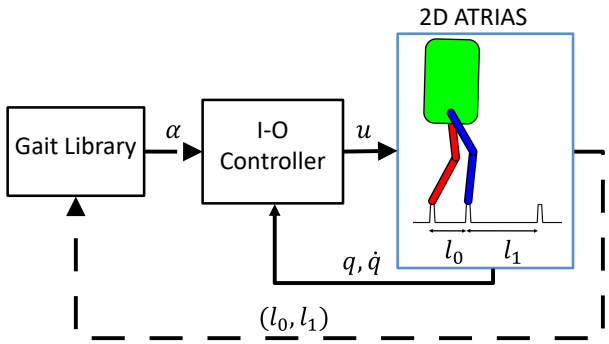


Figure 3.5: Diagram of the controller structure for the problem of changing step length only, integrating the gait library and I-O linearization controller. Solid lines represent signals in continuous time; dashed lines represent signals in discrete time.

***Remark* 3.1.2.** *Note that the proposed method has a resemblance to MPC. While we use a 2-step periodic gait, we switch the gait at the end of each step, i.e., half-way into the 2-step periodic gait. For instance, with current step length being l_0, and subsequent step lengths being l_1, l_2, we use a gait with (l_0, l_1) and switch at the end of the first step to (l_1, l_2) so that there is an overlap of one step between the gaits. This easily address gait transitions that typically cause large violations in unilateral force constraints, friction constraints, and torque constraints.*

***Remark* 3.1.3.** *Also note that the authors of [110] use control barrier functions to handle gait transitions. While this appears to work well, the feasibility of the quadratic program that enforces the control barrier constraint is not guaranteed. In this present work, as we will see, we achieve better results without using control barrier functions. We can easily add control barrier functions on top of the current method to further enforce these safety-critical constraints. Since the underlying method achieves the foot placement without requiring the barriers, the barriers will remain inactive most of the time, leading to better feasibility of the quadratic program.*

3.2 Cassie Robot

Cassie is a highly dynamic, under-actuated bipedal robot. Cassie has twenty DOFs as listed in (3.12):

$$\begin{aligned} q = [&q_x,\ q_y,\ q_z,\ q_{yaw},\ q_{pitch},\ q_{roll},\\ &q_{1L},\ q_{2L},\ q_{3L},\ q_{4L},\ q_{5L},\ q_{6L},\ q_{7L},\\ &q_{1R},\ q_{2R},\ q_{3R},\ q_{4R},\ q_{5R},\ q_{6R},\ q_{7R}]^T. \end{aligned} \tag{3.12}$$

where, $(q_x\ q_y\ q_z)$ and $(q_{yaw}\ q_{pitch}\ q_{roll})$ are the Cartesian coordinates of the pelvis and the Euler Angles in the Z-Y-X order, and $(q_{1L}, ..., q_{7L})$, $(q_{1R}, ..., q_{7R})$ are the generalized coordinates of the left and right legs, respectively. These correspond to the DOFs for each leg and are defined in

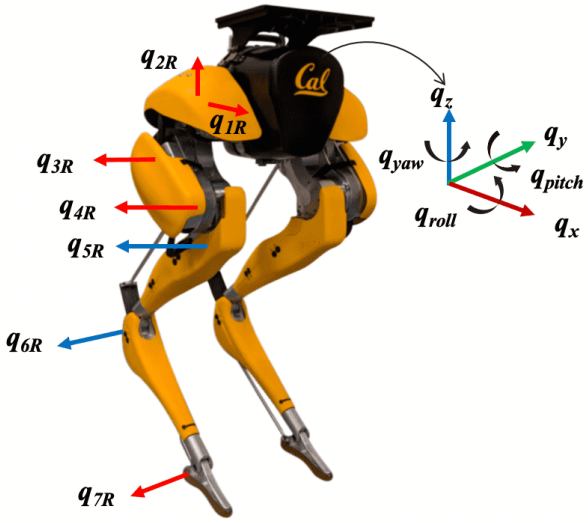


Figure 3.6: Kinematic model of Cassie showing the robot's generalized coordinates in the body frame.

(3.13).

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{bmatrix} = \begin{bmatrix} hip\ roll \\ hip\ yaw \\ hip\ pitch \\ knee\ pitch \\ shin\ pitch \\ tarsus\ pitch \\ toe\ pitch \end{bmatrix}. \tag{3.13}$$

Fig. 3.6 shows the generalized coordinates of Cassie's pelvis and right leg. The generalized coordinates of Cassie's left leg are similar to the right leg states. Each of Cassie's legs has seven DOFs with five of them being actuated: q_1, q_2, q_3, q_4, and q_7. The corresponding motor torques are u_1, u_2, u_3, u_4, and u_5. The other two DOFs, q_5 and q_6, are passive, corresponding to stiff springs.

The dynamics of Cassie can then be expressed in the following Euler-Lagrange dynamics:

$$D(q)\ddot{q} + H(q, \dot{q}) = Bu + J_s^T(q)\tau_s + J_c^T(q)F_c, \tag{3.14}$$

where q is the generalized coordinate vector as defined in (3.12), $D(q)$ is the mass matrix, $H(q, \dot{q})$ contains the centripetal, Coriolis, and gravitation terms, B is the motor torque matrix, u is the motor torque vector of dimension 10 corresponding to the actuators on the two legs, $J_s(q)$ is the Jacobian for the spring torques, τ_s is the spring torque vector, $J_c(q)$ is the Jacobian for the ground contact forces, and F_c is the ground contact force vector.

3.2.1 Gait Library Overview

For this work, we use the gait library previously generated in [112]. Here, a gait library of 1331 gaits were generated using C-FROST for the following gait parameters:

Figure 3.7: Snapshot of an experiment of the bipedal robot Cassie autonomously riding on Hovershoes. Experimental videos are at https://youtu.be/b2fKBb_0iTo.

- 11 forward/backward walking speeds V_x in the range $[-1,\ 1]$ m/s;
- 11 left/right side-stepping speeds V_y in the range $[-0.3,\ 0.3]$ m/s; and
- 11 stepping heights SH in the range $[-0.15, 0.15]$ m per step.

Using their open-source code, we replicated this gait library. In addition to the above-mentioned desired gait parameters, additional constraints are enforced to obtain stability and respect physical limitations of the robot. For completeness, we summarize them below:

- Step Duration is fixed at 0.4 s;
- Swing foot clearance at mid-step is at least 15 cm;
- Ground reaction forces must respect the friction cone and ZMP conditions defined in [113];
- Left and right stance components must be symmetric for straight-line walking;
- Hip abduction and rotation motors have tighter bounds for walking straight; and
- The torso pitch and roll must remain within $[-3,\ 3]$ degree range.

Using this gait library, we develop a learning-based control policy, called *GaitNet* in the next section.

3.2.2 Cassie Robot with Hovershoes

Legged locomotion is efficient when traveling over rough or discrete terrain, as discussed in the last section. On the other hand, wheeled locomotion is more efficient when traveling over flat continuous terrain [114]. Infact, since the discovery of wheel, we have significantly altered the landscape around us to make it very wheel-friendly. Now, Humans are able to optimize locomotion efficiency

using a healthy mix of riding micro-mobility platforms, such as Segways and Hovershoes along with running and walking. Enabling legged robots to autonomously ride on various personal mobility platforms will offer multi-modal locomotion capabilities to these robots which could be very useful in a time-sensitive scenario.

Autonomous robots with multi-modal locomotion capabilities can be game-changing in application ranging from package delivery to search and rescue [115]. Recently, the Cassie bipedal robot developed by Agility Robotics demonstrated autonomous Hovershoes riding capabilities [20] (see Fig. 3.7) using simple PD controllers applied to decoupled elements of the system dynamics.

3.3 Summary

In this chapters, we briefly summarized dynamical modeling and HZD-based control design for the ATRIAS and Cassie robots. Both of them use gait libraries, or a library of gaits, to achieve robust and versatile walking motions. Examples of versatility include accurate foot placement to walk on discrete terrain, speed regulation to walk on uneven surfaces, etc. In Chapter 8, a novel deep learning architecture is introduced to learn a unified policy from a gait library. Its utility is validated using the Cassie robot model and gait library. Further, in Chapter 9, a Deep Perception Module is developed and integrated with the gait-library-based controller built for ATRIAS robot to automate discrete terrain walking.

Part II

Geometric Control for Dynamic Legged Robots

Chapter 4

Euler v/s Geometric Formulations for 3D Pendulum Modeling and Control

4.1 3D Pendulum

Pendular systems are widely studied in the robotics and control community to discover and characterize nonlinear dynamical phenomena like symmetries, bifurcations, orbital stability, etc. ([116, 117, 118]). The deeper understanding of these behaviors helps control theorists devise nonlinear control techniques for effective stabilization and tracking ([119, 120, 121]). The impact of pendular systems in robotics cannot be understated. Most complex robotic systems commonly apply pendular abstractions to model and control their dominant behaviors. For example, see pendulum-like models in robotic manipulation ([122, 123, 124]), inverted pendulum models in legged locomotion ([3, 125, 126]), and multi-link pendular models in brachiation ([127, 128]). To date, multi-link pendula, remain the best nonlinear system models for benchmarking performance of new control algorithms on typical real-world challenges like underactuation, model uncertainity, stochasticity, etc.

Traditionally, nonlinear control design techniques used for stabilization or swinging up of a $3D$ or *spherical* pendulum (a.k.a $3D$ pendulum that cannot yaw) used Euler angles to define the pendulum configuration ([121, 129, 130]). However, more recently, coordinate-free formulations have been proposed to define pendulum configuration and corresponding dynamical models ([117, 118, 131]) and suitable control laws ([132, 133]) have been devised. These globally-defined dynamical formulations are *singularity-free*, i.e., they are devoid of kinematic singularities like gimbal-lock seen in locally-defined formulations like Euler angles (For example, the ZXY-ordering Euler-angles ($[\phi\,\theta\,\psi]^T$) exhibit a singularity when $\phi = \pi/2$). The singularity-free property of global formulations has implications in control design, particularly, recovery from large angle disturbances is possible and the controllers can be designed to be *almost-globally* attractive (as opposed to weaker local attractivity properties of Euler-based control designs). A new sub-field in nonlinear control, called *geometric control* has emerged that exploits these control laws and its been heavily applied in UAV literature, see [66, 67, 134, 135].

Most of this prior literature on geometric control has primarily focused on providing mathematical rigor and experimental validation to geometric control, specifically for almost-global attractiv-

ity property and the freedom from singularities. While these two properties are more meaningful for developing UAV maneuvers to quickly recover from large disturbances, for legged robots and other systems, the use of geometric control needs further motivation. Large angle recoveries are uncommon (restricted to acrobatic motions) and may be impractical during regular locomotion tasks like walking, running, etc. due to limited control authority, finite time (to impact) and narrow region-of-attraction.

4.2 Mathematical Modeling

To define the mathematical model of the 3D pendulum, we first need to define a stationary fixed frame (inertial frame) about which the pendulum configuration is measured. Denote this frame as $\{I\}$ and fix it at the origin, which is also the pivot for the pendulum. Second, we attach a moving frame (body frame), denoted as $\{B\}$, to the center-of-mass (CoM) of the pendulum. Having defined the frames, the next step is to choose a suitable parametrization to represent a *3D rotation* - this translates to estimating the orientation of $\{B\}$ w.r.t. $\{I\}$, as shown in Fig. 4.1. In this work, we use two methods, 1) Euler-based (ZYX ordering): $q \in \mathbb{R}^3$, and 2) Geometric: $R \in \mathbb{SO}(3)$. Other model parameters used for modeling and control design are defined and summarized in Table 4.1.

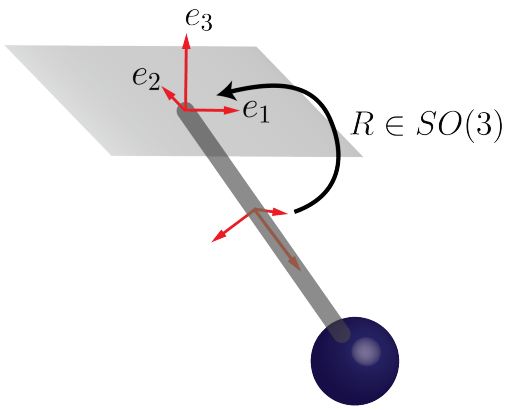


Figure 4.1: A 3D pendulum is a rigid body pinned at one end (pivot) restricting its motion to be purely rotational.

4.2.1 3D Pendulum Dynamics

Pendulum dynamics using both Euler and $\mathbb{SO}(3)$ formulations can be compactly expressed as,

$$\boldsymbol{D_e(q)\ddot{q} + C_e(q,\dot{q})\dot{q} + G_e(q) = B_e u_e}, \tag{4.1}$$

where $q = [\phi\ \theta\ \psi]^T$, $\dot{q} = [\dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T$, and $u_e \in \mathbb{R}^3$;

$$\boldsymbol{D_s(R)\dot{\Omega} + C_s(R,\Omega)\Omega + G_s(R) = B_s u_s}, \tag{4.2}$$

where $R \in \mathbb{SO}(3)$, $\Omega \in T_R\mathbb{SO}(3)$, and $u_s \in T_R^*\mathbb{SO}(3)$. Here, $T_R\mathbb{SO}(3)$ and $T_R^*\mathbb{SO}(3)$ are *tangent* and *co-tangent* spaces on $\mathbb{SO}(3)$ at the configuration R.

$\{I\}$	Inertial frame fixed to the pivot.
$\{B\}$	Body frame fixed at the CoM of pendulum.
$e_3 \in \mathbb{R}^3$	Standard unit vector along the gravity direction (pointing upward).
$m \in \mathbb{R}$	Mass of the pendulum link.
$J \in \mathbb{R}^{3\times3}$	Inertia of the pendulum expressed in frame $\{B\}$.
$l \in \mathbb{R}$	Length of the pendulum w.r.t the pivot.
$l_c \in \mathbb{R}$	Length of the pendulum CoM w.r.t the pivot.
$R \in \mathbb{SO}(3)$	Rotation matrix from {B} to {I}.
$\Omega \in \mathbb{R}^3$	Angular velocity of the pendulum expressed in the body frame $\{B\}$.
ϕ, θ, ψ	Roll, Pitch and Yaw angles used in Euler parametrization.
$\widehat{(.)}$	*hat* map is a linear mapping from $\mathbb{R}^3$ to $\mathfrak{so}(3)$.
$(.)^\vee$	*vee* map is a linear mapping from $\mathfrak{so}(3)$ to $\mathbb{R}^3$.

Table 4.1: Symbols and parameter definitions for $3D$ Pendulum modeling and control design.

The Euler dynamics can be easily derived and we omit it here for brevity. However, $D_e(q)$ will be used later to check for singularities and to define suitable mappings to go back and forth between Euler and geometric inputs, etc. $\mathbb{SO}(3)$ dynamics equations of motion are compact, as shown below:

$$\dot{R} = R\widehat{\Omega}, \tag{4.3}$$

$$\dot{\Omega} = J^{-1}(-\widehat{\Omega}J\Omega + mg\widehat{l_c}R^T e_3 + u). \tag{4.4}$$

For the derivation of these equations, see [71] and [117]. Note that the dynamics cannot be directly compared and we need to define suitable mapping between the Euler angles, rates and inputs to their counterparts in $\mathbb{SO}(3)$.

4.2.2 Transfer Maps

Using the ZXY ordering of Euler angles, we define $\mathcal{R}(q) : \mathbb{R}^3 \to \mathbb{SO}(3)$ whose expression is given as,

$$\mathcal{R}(q) = \begin{pmatrix} c\theta\, c\psi - s\phi\, s\theta\, s\psi & -c\phi\, s\psi & c\psi\, s\theta + c\theta\, s\phi\, s\psi \\ c\theta\, s\psi + c\psi\, s\phi\, s\theta & c\phi\, c\psi & s\theta\, s\psi - c\theta\, c\psi\, s\phi \\ -c\phi\, s\theta & s\phi & c\phi\, c\theta \end{pmatrix}. \tag{4.5}$$

Here, we compact $cos(\alpha)$ and $sin(\alpha)$ as $c\alpha$ and $s\alpha$, respectively, and α is a placeholder for any Euler angle in q.

Next, we define $\mathcal{T}_{\dot{q}}(q) : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ to convert Euler rates ($\dot{q}$) to angular velocities (Ω) as shown

below:

$$\mathcal{T}_{\dot{q}}(q) = \begin{pmatrix} c\theta & 0 & -c\phi\, s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi\, c\theta \end{pmatrix}, \qquad \text{s.t. } \Omega = \mathcal{T}_{\dot{q}}(q)\dot{q}. \tag{4.6}$$

Finally, to convert Euler input (u_e) to $\mathbb{SO}(3)$ input (u_s), we define $\mathcal{T}_u(q, R) : \mathbb{R}^3 \times \mathbb{SO}(3) \to \mathbb{R}^{3\times 3}$ as,

$$\mathcal{T}_u(q, R) = (D_s^{-1}B_s)^{-1}\mathcal{T}_{\dot{q}}(D_e^{-1}B_e), \qquad \text{s.t } u_s = \mathcal{T}_u u_e\ . \tag{4.7}$$

$\mathcal{T}_{\dot{q}}$ Derivation:

Since we are using ZXY Euler sequencing, for $w \in [Z\ X\ Y]^T$, $i \in [3\ 1\ 2]^T$, and $\alpha \in [\psi\ \phi\ \theta]^T$, we define $R_w(\alpha) : \mathbb{R} \to \mathbb{SO}(3)$ as a mapping from an Euler angle α to its corresponding axis-specific rotation matrix R_w. Accordingly, we have $\dot{R}_w = R_w\widehat{\dot{\alpha}e_i}$. Using this notation, we express R as a product of its individual axis-wise rotation elements $R_z(\psi)$, $R_x(\phi)$, and $R_y(\theta)$. We can then compute its first derivative, $\dot{R}$ as a function of Euler angles q and their Euler rates $\dot{q}$, as shown below:

$$\begin{aligned} R &= R_z R_x R_y, \\ R &= \dot{R}_z R_x R_y + R_z \dot{R}_x R_y + R_z R_x \dot{R}_y, \\ \dot{R} &= R_z\widehat{\dot{\psi}e_3}R_x R_y + R_z R_x\widehat{\dot{\phi}e_1}R_y + R_z R_x R_y\widehat{\dot{\theta}e_2}. \end{aligned}$$

From equation (4.3)), we know that $\Omega = (R^T\dot{R})^\vee$. Therefore,

$$\begin{aligned} \widehat{\Omega} &= R_y^T R_x^T R_z^T R_z\widehat{\dot{\psi}e_3}R_x R_y + R_y^T R_x^T R_z^T R_z R_x\widehat{\dot{\phi}e_1}R_y + R_y^T R_x^T R_z^T R_z R_z R_x R_y\widehat{\dot{\theta}e_2}, \\ &= R_y^T R_x^T\widehat{\dot{\psi}e_3}R_x R_y + R_y^T\widehat{\dot{\phi}e_1}R_y + \widehat{\dot{\theta}e_2}, \\ &= \widehat{R_y^T R_x^T\dot{\psi}e_3} + \widehat{R_y^T\dot{\phi}e_1} + \widehat{\dot{\theta}e_2}. \end{aligned}$$

Finally, we apply a *vee map* on both sides to get,

$$\Omega = [R_y^T e_1\ \ e_2\ \ Ry^T Rx^T e_3]\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}, \tag{4.8}$$

$$\mathcal{T}_{\dot{q}}(q) = [R_y^T e_1\ \ e_2\ \ Ry^T Rx^T e_3] = \begin{pmatrix} c\theta & 0 & -c\phi\, s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi\, c\theta \end{pmatrix}$$

$\mathcal{T}_u$ Derivation:

We can obtain $\mathcal{T}_u$ by taking a time derivative of equation (4.6), then substituting the dynamics from equations (4.10) and (4.11) for $\ddot{q}$ and $\dot{\Omega}$. Finally, we equate the input vector fields on both sides to

finish the derivation.

$$\begin{aligned}\Omega &= \mathcal{T}_{\dot{q}}(q)\dot{q},\\ \dot{\Omega} &= \dot{\mathcal{T}}_{\dot{q}}\dot{q} + \mathcal{T}_{\dot{q}}\ddot{q}\\ \underbrace{f_s}_{=:A_e} + \underbrace{g_s u_s}_{=:B_e} &= \underbrace{\dot{\mathcal{T}}_{\dot{q}}\dot{q} + \mathcal{T}_{\dot{q}} f_e}_{=:A_s} + \underbrace{\mathcal{T}_{\dot{q}}(g_e u_e)}_{=:B_s}\end{aligned} \tag{4.9}$$

Note that, the equality in (4.9) must hold for all u_e and u_s. Setting $u_e = u_s = \mathbf{0}$ results in $A_e = A_s$. Eliminating, A_e and A_s, we can define,

$$\begin{aligned}u_s &= \underbrace{(g_s^{-1}\mathcal{T}_{\dot{q}} g_e)}_{\mathcal{T}_u} u_e\\ \mathcal{T}_u &= (D_s^{-1}B_s)^{-1}\mathcal{T}_{\dot{q}}(D_e^{-1}B_e)\end{aligned}$$

Using these transfer maps it is easy to compare the Euler and geometric control laws which are going to be defined in the next section.

4.2.3 Pendulum State Sampling:

In the following sections, we empirically evaluate kinematic and dynamical defects like singularities, control design attributes like error metrics, input profiles, etc. for both $\mathbb{SO}(3)$ and Euler models for a wide range of pendulum configurations/states. The main emphasis of this work is to use this comprehensive empirical evaluation to highlight the benefits of *geometric control* and complement mathematically rich previous works.

The states are sampled from a $[-\pi, \pi]$ range of ϕ, θ, and ψ values with a resolution of $\pi/4$. The corresponding $\mathbb{SO}(3)$ states can be obtained using transfer map $\mathcal{R}$. For dynamic and kinematic studies, these configurations can be treated as a potential intermediate state in the pendulum motion trajectory. For control studies, they are used as initial conditions from which the pendulum is stabilized to its hanging equilibrium position. Some of these state samples are shown in Fig. 4.2 along-with the sample numbers to be used in plots that follow.

***Remark* 4.2.1.** *Note that, both $\mathcal{T}_{\dot{q}}$ and D_e lose rank $\phi = \pi/2$. This is the singularity issue that plagues Euler parametrization. Singular states are littered all over the configuration space, as shown in Fig. 4.2, making large disturbance recovery challenging using Euler parametrization.*

4.3 Control Laws

In this work, we choose Feedback Linearization (FL) for pendulum stabilization. We begin by rewriting equations (4.1) and (4.2) in the control-affine form as,

$$\dot{x}_e = f_e(x_e) + g_e(x_e)u_e, \tag{4.10}$$
$$\dot{x}_s = f_s(x_s) + g_s(x_s)u_s, \tag{4.11}$$

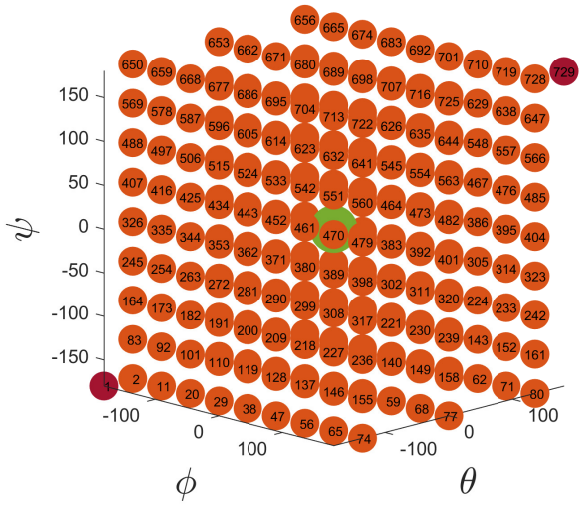


Figure 4.2: The figure shows a coarse grid of the sampling points used in this study along with their *sample number*. The starting and final experiment indices are highlighted in **red**, and correspond to ($\phi = \theta = \psi = -\pi$) and ($\phi = \theta = \psi = \pi$), respectively. To better visualize the control studies, the desired final position (hanging equlibrium) is also shown in **green** and corresponds to ($\phi = \theta = \psi = 0$).

where, $x_e = [q\ \dot{q}]^T$ and $x_s = [R\ \Omega]^T$. The Feedback Linearization schemes for both models are summarized below:

Euler:

For the Euler case, it is fairly straightforward to derive an appropriate feedback linearizing policy by defining a suitable output as $y_e = h_e(q) = q - q_d$. Here, $q_d(t)$ can be time-varying and the control problem transitions to tracking from regulation. The output is relative degree 2. The control goal is to drive $h_e(q)$

$$y_e := h_e(q) = q - q_d(t) \to 0, \tag{4.12}$$

$$\dot{y}_e := L_{f_e} h_e = \dot{q} - \dot{q}_d(t), \tag{4.13}$$

$$\ddot{y}_e := L_{f_e}^2 h_e + (L_{g_e} L_{f_e} h_e) u_e = \ddot{q} - \ddot{q}_d(t), \tag{4.14}$$

where, $L_f h(q)$ is the lie derivative and $L_f h = \frac{d}{dt} h(q) = \frac{\partial h}{\partial q} f$, assuming $\dot{q} = f(q)$. Now lets define suitable *feedforward* and *feedback* terms,

$$u_e^{ff} := -(L_{g_e} L_{f_e} h_e)^{-1} (L_{f_e}^2 h_e), \tag{4.15}$$

$$u_e^{fb} := -(L_{g_e} L_{f_e} h_e)^{-1} (K_p y_e + K_d \dot{y}_e). \tag{4.16}$$

Applying $u_e := u_e^{ff} + u_e^{fb}$ in equation (4.14) results in a closed-loop linear system,

$$\ddot{y}_e + K_d \dot{y}_e + K_p y_e = 0. \tag{4.17}$$

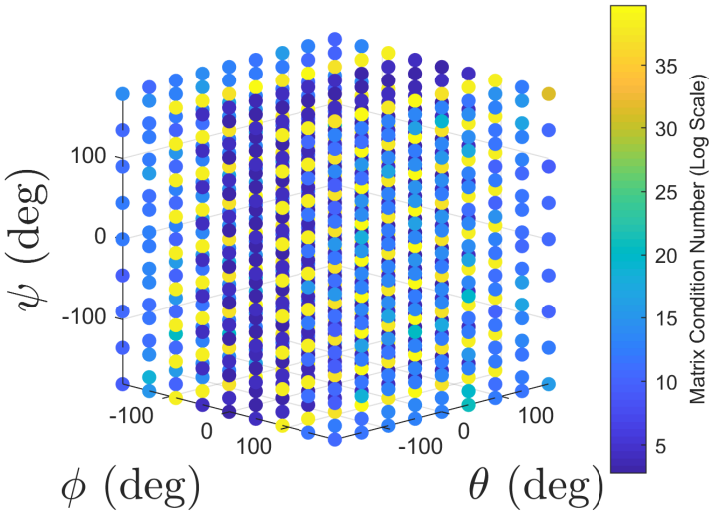


Figure 4.3: A scatter plot of $D_s(q)$ matrix condition number (on the log scale) for all the state samples from Section 4.2.3. All the **yellow** points are singular.

$\mathbb{SO}(3)$:

Feedback Linearization for $\mathbb{SO}(3)$ dynamics is non-trivial. Here we use geometric PD error functions previously introduced in [136, 133] and presented in (4.18). Here e_R is the configuration error akin to position error in the euclidean space. Similarly, e_Ω is angular velocity error. The extra term $(R^T R_d)$ in e_Ω is called a *transport map*. Since the tangent space on $\mathbb{SO}(3)$ changes with configuration ($\Omega \in T_R\mathbb{SO}(3)$ & $\Omega_d \in T_{R_d}\mathbb{SO}(3)$), the transport map helps project desired angular velocities onto the tangent space at the current configuration ($T_{R_d}\mathbb{SO}(3) \to T_R\mathbb{SO}(3)$) for correct comparison.

Error Functions on $\mathbb{SO}(3)$

$$e_R = \frac{1}{2}[R_d^T R - R^T R_d]^\vee \tag{4.18a}$$

$$e_\Omega = \Omega - (R^T R_d)\Omega_d \tag{4.18b}$$

$$\dot{e}_R = \underbrace{\frac{1}{2}[tr(R^T R_d)I - R^T R_d]}_{=:T(R,R_d)} e_\Omega \tag{4.18c}$$

Similar to the Euler case, we design a relative degree two output, y_s using the e_R error function. We can feedback linearize the systems to result in a desired closed-loop dynamics of the form

$\ddot{y}_s + K_d\dot{y}_s + K_p y_s = 0$ (or) $\ddot{e}_R + K_d\dot{e}_R + K_p e_R = 0$. Accordingly, we have the control goal as,

$$y_s = h_s(R) := e_R(R, R_d) \to 0, \tag{4.19}$$

$$\dot{y}_s := L_f h_s = \dot{e}_R = T(R, R_d)e_\Omega. \tag{4.20}$$

At this point, it's worth noting that $||\dot{e}_R|| <= ||e_\Omega||$ as $||T|| <= 1$ (for proof, see [137] Appendix 1(a)). Therefore, it is sufficient to choose only e_Ω as the relative degree one output instead of $\dot{e}_R$. Note that this is a conservative approximation and $e_\Omega \to 0$ guarantees $\dot{e}_R \to 0$. Now, we take the second derivative,

$$\ddot{y}_s = L_f^2 h_s + (L_g L_f h_s)u_s = T\dot{e}_\Omega. \tag{4.21}$$

Lastly, we define

$$u_s^{ff} := -(L_g L_f h_s)^{-1}(L_f^2 h_s), \tag{4.22}$$

$$u_s^{fb} := -(L_g L_f h_s)^{-1}(K_p e_R + K_d e_\Omega), \tag{4.23}$$

$$u_s = u_s^{ff} + u_s^{fb}. \tag{4.24}$$

Here, $L_f^2 h_s = f_s - (R^T R_d)\dot{\Omega}_d - (\dot{R}R_d)\Omega_d$ and $L_g L_f h_s = g_s$. Applying u_s results in a closed-loop linear system,

$$\ddot{y}_s + K_d\dot{y}_s + K_p y_s = 0, \tag{4.25}$$

$$\implies \dot{e}_\Omega + K_d e_\Omega + K_p e_R = 0$$

Error Metrics: Before executing the proposed controllers and testing their performance, it is worth defining suitable error metrics for the euclidean and $\mathbb{SO}(3)$ spaces to measure error growth between desired and actual pendulum states as they spread apart. This metric is directly proportional to the control expense involved. This helps us visualize apriori configurations that are harder to control.

Euler: Define non-negative function $\Psi_e : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ as,

$$\Psi_e = \frac{1}{2}\sqrt{(q - q_d)^T(q - q_d)} \tag{4.26}$$

$\mathbb{SO}(3)$ Define $\Psi_s : \mathbb{SO}(3) \times \mathbb{SO}(3) \to \mathbb{R}$ as,

$$\Psi_s = \frac{1}{2}tr[I - R_d^T R] \tag{4.27}$$

Assuming the hanging equilibrium configuration is desired, we pick q from all the states defined in the earlier section (Fig. 4.2) and plot Ψ_e in Fig. 4.5 and Ψ_s in Fig. 4.4.

From the two figures, it is clear that all the configurations where either one or two Euler angles are $\approx \pi$ (antipodal configurations), have higher errors. These configurations are the cross-type bands around face centers on the samples cube in Fig. 4.4. While this makes sense, as these are the farthest points from the desired configuration, in Fig. 4.5 however, the highest error points are configurations with all the three Euler angles $\approx \pi$ (depicted as the edges of the samples cube). This is very non-intuitive as these points are actually very close to the desired configuration and should require minimal effort to reach q_d. Even before evaluating the controller performance, the error metric plotting exercise gives an insight into the potential failure modes of Euler-based controller.

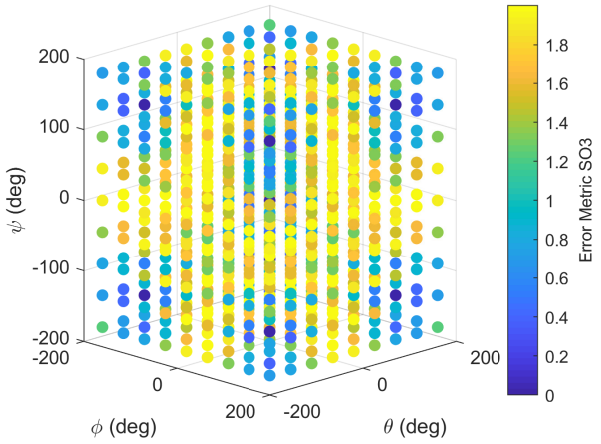


Figure 4.4: Ψ_s for $R_d = I$ and for all R in Section 4.2.3.

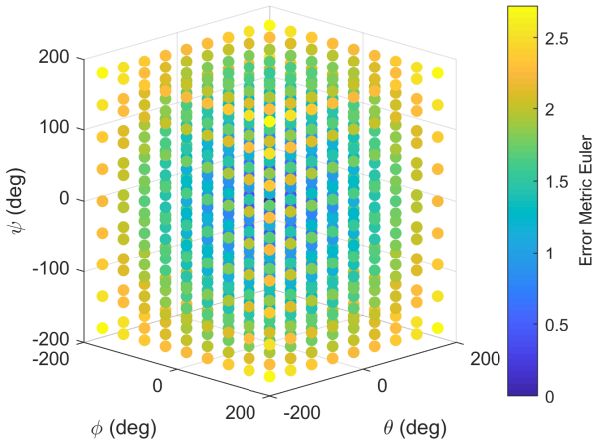


Figure 4.5: Ψ_e for $q_d = [0\ 0\ 0]^T$ and for all q in Section 4.2.3.

4.4 Results and Discussion

The control objective in this study is to stabilize the $3D$ pendulum to its hanging equilibrium position, i.e. $q_d = [0\ 0\ 0]^T$ or $R_d = I$. Velocities $\dot{q}_d$ and Ω_d are both randomized for each trial with a max of $4\ rads/s$ per state. The initial condition of the pendulum could be any state from the samples defined in Section 4.2.3. All the experiments are run for a fixed time T. For the pendulum model, we chose, mass as $m = 1\ kg$, length as $l = 0.5\ m$, and inertia w.r.t body-frame as $J = diag(0.1625,\ 0.1625,\ 0.01)\ kg\ m^2$. The controller gains are kept same for both the Euler and geometric controllers (as $K_p = 100,\ K_d = 20$) to better visualize and compare performance. We applied an Euler controller for the Euler dynamics and a geometric controller for the $\mathbb{SO}(3)$ dynamics and logged key performance metrics like, input integral over time, power integral over time, max input, max power, etc.

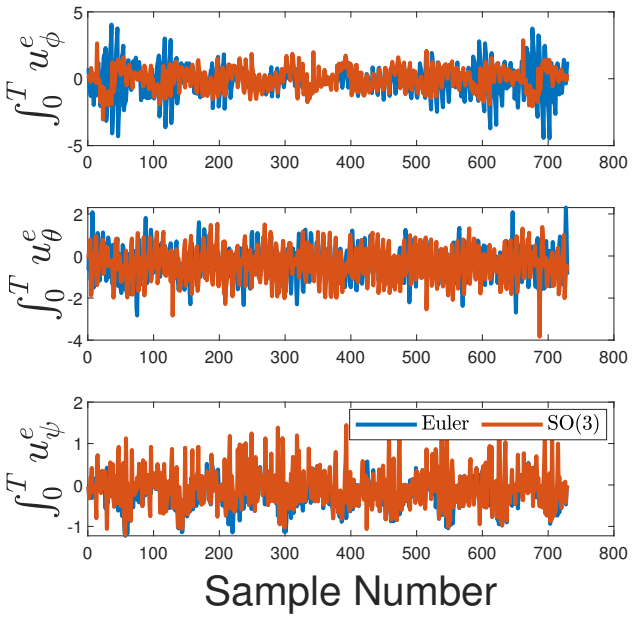


Figure 4.6: Individual joint input integrals for each experiment.

How to compare the two? After applying the two controllers and logging performance data across a range of initial conditions, we compare both of them in the Euler-space. We use the maps defined in Section 4.2.2 to map $R(t)$, $\Omega(t)$, and $u_s(t)$ to $\tilde{q}(t)$, $\dot{\tilde{q}}(t)$, and $\tilde{u}_e$, respectively. Therefore,

$$\tilde{q} = \mathcal{R}^{-1}(R),\ \dot{\tilde{q}} = \mathcal{T}_{\dot{q}}^{-1}(\Omega),\ \tilde{u}_e = \mathcal{T}_u^{-1}(u_s). \tag{4.28}$$

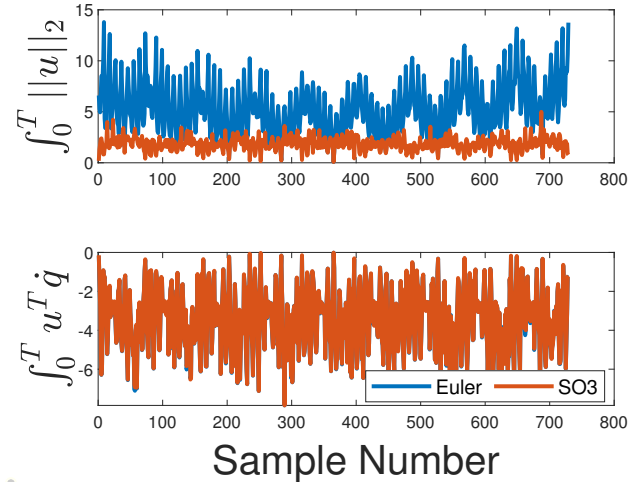


Figure 4.7: The top plot shows normed input integral and the bottom plot shows power integral for each experiment. Note that *orange* points indicate $\tilde{u}_e$ and *blue* points indicate u_e.

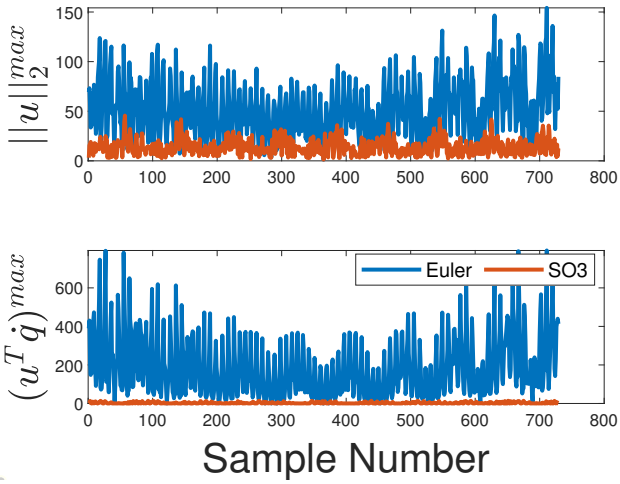


Figure 4.8: The top plot shows max input norm and the bottom plot shows max power for each experiment.

4.4.1 Geometric Control is more efficient

Having logged both u_e and $\tilde{u}_e$, we plot joint-wise integrals for each experiment in Fig. 4.6. Note that, $\tilde{u}_\phi$ is particularly very efficient compared to u_ϕ. The existence of singularity in the ϕ direction impacts all trajectories in the Euler model with large ϕ errors. Next, we evaluate metrics for full attitude control performance. We show normed input integral and power integral plots in Fig. 4.7 and max input and max power plots in Fig. 4.8. Note that both over time and in magnitude the geometric controllers are consistently more efficient.

However, it's worth noting that the power integral is comparable between the two. The error functions e_R and e_Ω have a peculiar issue that is also highlighted in [135]. The farther the current configuration from the desired the lower proportional controller stiffness causing the pendulum to stall for too long and accumulate error and spiking up velocities. This could be the cause for power integral to be comparable even though lower input was being used. In [135], alternate error function choices are proposed to mitigate this issue. Also, note that the max input and power values for Euler controller are particularly high in some cases. This is mainly caused while crossing the singularity configuration and the numerical integrator applying arbitrarily large inputs. However, on a real system with strict input saturations the difference would be less severe.

4.5 Summary

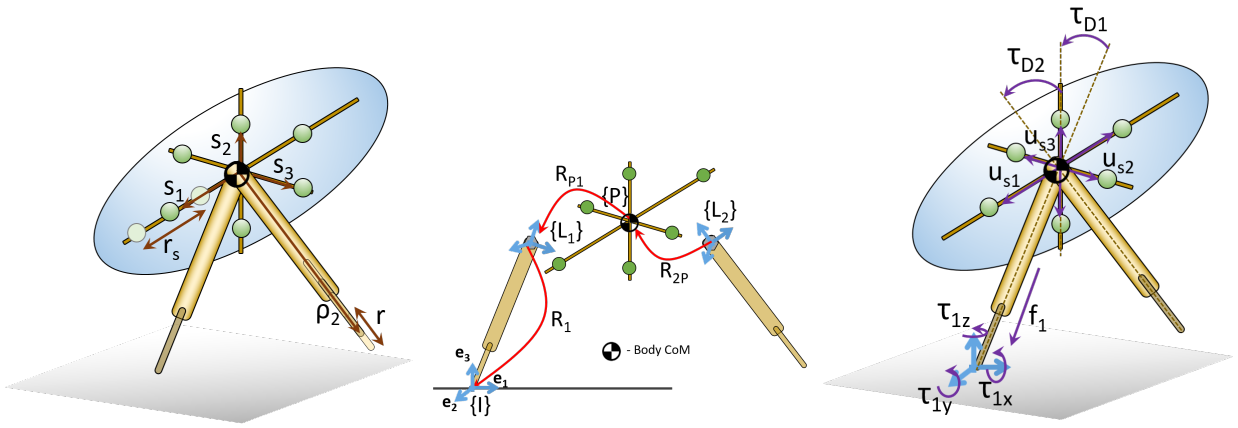
In this work, we presented two formulations for modeling and control of a $3D$ pendulum - one is Euler-parameterized and the other is coordinate-free and evolves on the manifold space $\mathbb{SO}(3)$. Moreover, through comprehensive empirical evaluation, we demonstrate that geometric controller is generally more efficient than the traditional Euler-parametrized one for all configurations and not just for large error recovery as was mainly shown so far.

Chapter 5

Geometric Modeling and Control of a Reaction Mass Biped

5.1 Reaction Mass Biped Model

Inverted Pendulum based reduced-order models offer many valuable insights into the much harder problem of bipedal locomotion. While they help in understanding leg behavior during walking, they fail to capture the natural balancing ability of humans that stems from the variable rotational inertia on the torso. In an attempt to overcome this limitation, we introduce a new Reaction Mass Biped (RMB) model. It is a generalization of the previously introduced *Reaction Mass Pendulum (RMP)*, which is a multi-body inverted pendulum model with an extensible leg and a variable rotational inertia torso. The dynamical model for the RMB is hybrid in nature, with the roles of stance leg and swing leg switching after each cycle. It is derived using a variational mechanics approach and is therefore coordinate-free. The RMB model has thirteen degrees of freedom, all of which are assumed to be actuated.



(a) RMB Model (b) Rotations (c) Actuation

Figure 5.1: A schematic of the RMB model.

Physical description of the RMB Model

As shown in Fig.5.1(a), RMB consists of two extensible legs whose lengths are ρ_1 and ρ_2 as measured from the RMB CoM. The CoM of both the legs is assumed to coincide with the RMB CoM. Moreover,for some nominal length ρ_0, the moment of inertia for the legs is given by J_{L0}. The legs can extend up to a max distance r from ρ_0. The torso is made up of three pairs(depicted in green) of reaction masses s_1,s_2,s_3,respectively. Each pair is arranged along an orthogonal axis e_i of the torso's body-fixed frame. With one each on either side of the RMB CoM , they are constrained to move equidistantly so that the torso CoM always coincides with the RMB CoM. In Fig. 5.1(b), the frames of reference used in this study are depicted and they are defined in Table 5.1. We assume full actuation for the RMB model, as shown in Fig. 5.1(c). τ_1 rotates the ankle joint at the stance foot along pitch, roll and yaw directions w.r.t. the inertial frame $\{I\}$. f_1 is the force used to extend the telescopic stance leg. $\tau_{D1} \in \mathbb{R}^3$ rotates torso frame $\{P\}$ w.r.t the stance leg frame $\{L_1\}$. Similarly, $\tau_{D2} \in \mathbb{R}^3$ rotates swing leg frame $\{L_2\}$ w.r.t the torso frame $\{P\}$. Finally, $u_s \in \mathbb{R}^3$ actuates the reaction masses pairs on the torso. It is the motion of these point-masses that induces variability into the torso's inertia.

Stance Dynamics (or) Fixed-base Robot Model

We develop a coordinate-free dynamic model for the stance phase of the Reaction Mass Biped, as shown in Fig. 5.1(a), by using rotation matrices to represent the attitudes of the two legs, R_1, R_2, and the torso, R_P, along with scalars ρ_1, ρ_2 to represent the length of the two legs, and s_i to represent the position of the i^{th} pair of reaction masses. Note that, there are three pairs of reaction-masses, all of which are mutually orthogonal. During the stance phase, the stance-leg is assumed to be pinned to the ground. The configuration manifold of the system is then given by $\mathcal{Q}_s = C \times SO(3) \times SO(3) \times SO(3) \times S \times C$, with $\rho_1, \rho_2 \in C = [0, r]$, $R_1, R_P, R_2 \in SO(3)$, $s = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}^T \in S = [0, r_s] \times [0, r_s] \times [0, r_s]$. The symbols used in this section are tabulated in Table 5.1.

We have the following kinematic relations in the system, $\dot{R}_1 = R_1\Omega_1^\times$, $\dot{R}_{P1} = R_{P1}\Omega_{P1}^\times$, $\dot{R}_P = R_P\Omega_P^\times$, $\dot{R}_{2P} = R_{2P}\Omega_{2P}^\times$, $\dot{R}_2 = R_2\Omega_2^\times$, where, $\Omega_1, \Omega_2, \Omega_P$ are the respective body angular velocities, and are related by

$$\Omega_P = \Omega_{P1} + R_{P1}^T\Omega_1, \quad \text{where,} \quad R_{P1} = R_1^T R_P, \tag{5.1}$$

$$\Omega_2 = \Omega_{2P} + R_{2P}^T\Omega_P, \quad \text{where,} \quad R_{2P} = R_P^T R_2. \tag{5.2}$$

Here, the $(.)^\times$ is called the hat operator and it is used to map angular velocities from $\mathbb{R}^3$ to $\mathfrak{so}(3)$ (the Lie algebra of $\mathbb{SO}(3)$). Next, we derive an expression for the kinetic energy of the system, $\mathcal{T}_s : T\mathcal{Q}_s \to \mathbb{R}$. We do this by first finding the position of the center-of-mass (COM) of the stance leg, b, and the positions of the reaction mass pairs, $p_{i\pm}$, in the inertial frame $\{I\}$ as follows,

$$b = \rho_1 R_1 e_3, \quad p_{i\pm} = b \pm s_i R_P e_i.$$

The dot product of their velocities can then be respectively computed as,

$$||\dot{b}||^2 = \dot{\rho}_1^2 - \rho_1^2\Omega_1^T(e_3^\times)^2\Omega_1 \tag{5.3}$$

$$||\dot{p}_{i+}||^2 + ||\dot{p}_{i-}||^2 = 2\left(||\dot{b}||^2 + \dot{s}_i^2 - s_i^2\Omega_P(e_i^\times)^2\Omega_P\right), \tag{5.4}$$

where $(\cdot)^\times : \mathbb{R}^3 \to \mathfrak{so}(3)$ is the skew operator, defined such that $x^\times y = x \times y, \forall x, y \in \mathbb{R}^3$. The kinetic energy of the system is then given by $\mathcal{T}_s = T_1 + T_P + T_2$, where, T_1, T_2 are the kinetic energies of the two legs respectively, and T_P is the kinetic energy of the torso, computed as,

$$\begin{aligned} T_1 &= \frac{1}{2} m_L ||\dot{b}||^2 + \frac{1}{2} \Omega_1^T J_{L_0} \Omega_1, \\ T_P &= \frac{1}{2} m_P \left(\sum_{i=1}^{3} ||\dot{p}_{i+}||^2 + ||\dot{p}_{i-}||^2 \right) + \frac{1}{2} \Omega_P^T J_{P_0} \Omega_P, \\ T_2 &= \frac{1}{2} m_L ||\dot{b}||^2 + \frac{1}{2} \Omega_2^T J_{L_0} \Omega_2. \end{aligned}$$

Thus, the total kinetic energy of the system is,

$$\mathcal{T}_s = \frac{1}{2} m \dot{\rho}_1^2 + \sum_{i=1}^{3} m_P \dot{s}_i^2 + \frac{1}{2} \Omega_1^T J_1(\rho_1) \Omega_1 + \frac{1}{2} \Omega_P^T J_P(s) \Omega_P + \frac{1}{2} \Omega_2^T J_{L_0} \Omega_2, \tag{5.5}$$

where,

$$\begin{aligned} J_1(\rho_1) = J_{L_0} + K_1(\rho_1), \qquad & K_1(\rho_1) = -m\rho_1^2 (e_3^\times)^2, \qquad m = 2m_L + 6m_P, \\ J_P(s) = J_{P_0} + K_P(s), \qquad & K_P(s) = -2 \textstyle\sum_{i=1}^{3} m_P s_i^2 (e_i^\times)^2. \end{aligned}$$

***Remark* 5.1.1.** *Note that, the length of the swing leg, ρ_2 does not appear in the kinetic energy of the system, and as we will consequently see, it will not appear in the dynamics either. This is because of representing COM of the swing leg at the hip. Moving COM location to half-way along the leg will ensure the swing leg length velocity appears in the kinetic energy, thereby introducing an additional dynamical equation for ρ_2. The dynamical model and the controller developed here can easily be extended to incorporate a variable swing leg length, at the cost of adding another degree of freedom and some complexity to the dynamics. Here we treat the simpler case by assuming the swing leg length to be constant. The swing leg's rotation does appear in the kinetic energy through Ω_2.*

Having developed an expression for the kinetic energy, we next compute the Potential Energy, $\mathcal{U}_s : \mathcal{Q}_s \to \mathbb{R}$, as,

$$\mathcal{U}_s = -mg\rho_1 R_1^T e_3 \cdot e_3. \tag{5.6}$$

Note that the negative sign arises due to our convention of e_3 being along the direction of uniform gravity.

The Lagrangian of the system $\mathcal{L}_s : T\mathcal{Q}_s \to \mathbb{R}$ is then given by $\mathcal{L}_s = \mathcal{T}_s - \mathcal{U}_s$. The equations of motion can then be computed through the Lagrange-d'Alembert principle by writing the variation of the action integral as,

$$\int \left(\delta \mathcal{L}_s + \eta_1 \cdot \tau_1 + \delta\rho_1 f_1 + \eta_{P1} \cdot \tau_{D_1} + \sum_{i=1}^{3} \delta s_i u_{s_i} + \eta_{P2} \cdot \tau_{D_2} \right), \tag{5.7}$$

$m_L \in \mathbb{R}$	Mass of either Leg at Hip Joint
$m_P \in \mathbb{R}$	Mass of Reaction masses
$m \in \mathbb{R}$	Mass of the entire system
$J_{L_0} \in \mathbb{R}^{3\times3}$	Inertia matrix of either leg with respect to the body-fixed frame when leg length is its nominal value ρ_0
$J_{P_0} \in \mathbb{R}^{3\times3}$	Inertia matrix of the torso with respect to the body-fixed frame
$\{I\}$	Inertial frame at the stance foot
$\{L_1\}$	Body frame of the stance leg at the hip joint
$\{P\}$	Body frame of the torso at the hip joint
$\{L_2\}$	Body frame of the swing leg at the hip joint
$\rho_1 \in C \subset \mathbb{R}$	Distance between CoM of the stance leg and its ankle
$\rho_0 \in C \subset \mathbb{R}$	Constant distance between CoM of the swing leg and the hip joint
$R_1 \in SO(3)$	Rotation matrix of the stance leg from the body-fixed frame to the inertial frame {I}
$R_P \in SO(3)$	Rotation matrix of the torso from the body-fixed frame to the inertial frame {I}
$R_2 \in SO(3)$	Rotation matrix of the swing leg from the body-fixed frame to the inertial frame {I}
$R_{P1} \in SO(3)$	Rotation matrix of the torso from the body-fixed frame {P} to the stance leg body-fixed frame $\{L_1\}$
$R_{2P} \in SO(3)$	Rotation matrix of the swing leg from the body-fixed frame $\{L2\}$ to the torso body-fixed frame {P}
$\Omega_1 \in \mathbb{R}^3$	Angular velocity of the stance leg in the body-fixed frame
$\Omega_2 \in \mathbb{R}^3$	Angular velocity of the swing leg in the body-fixed frame
$\Omega_P \in \mathbb{R}^3$	Angular velocity of the torso in the body-fixed frame
$s_i \in S \subset \mathbb{R}$	Position of the i'th reaction mass
$e_3 \in \mathbb{R}^3$	Standard unit vector along the gravity direction (downward) in the inertial frame

Table 5.1: Enumeration of the symbolic notation used to develop RMB Model.

where the first term in the integral represents the variation of the Lagrangian, computed using the following variations on SO(3),

$$\delta R_1 = R_1\eta_1^\times,\ \eta_1 \in \mathbb{R}^3,\ \ \delta\Omega_1 = \Omega_1^\times\eta_1 + \dot{\eta}_1, \tag{5.8}$$

$$\delta R_P = R_P\eta_P^\times,\ \eta_P \in \mathbb{R}^3,\ \ \delta\Omega_P = \Omega_P^\times\eta_P + \dot{\eta}_P, \tag{5.9}$$

$$\delta R_2 = R_2\eta_2^\times,\ \eta_2 \in \mathbb{R}^3,\ \ \delta\Omega_2 = \Omega_2^\times\eta_2 + \dot{\eta}_2, \tag{5.10}$$

and, all the other subsequent terms in the integral representing the infinitesimal virtual work, where,

$$\eta_{P1} = \eta_P - R_P^T R_1\eta_1, \quad \eta_{P2} = \eta_2 - R_2^T R_P\eta_P.$$

For more details and illustrations of the actuators, please refer to Fig. 5.1(c). The dynamical equations of motion can then be obtained by setting the above integral to zero for all possible

variations, resulting in,

$$m\ddot{\rho}_1 = -m\rho_1\Omega_1^T(e_3^\times)^2\Omega_1 + mge_3^T R_1^T e_3 + f_1, \tag{5.11}$$

$$\begin{aligned} J_1(\rho_1)\dot{\Omega}_1 = &-\Omega_1 \times J_1(\rho_1)\Omega_1 + 2m\rho_1\dot{\rho}_1(e_3^\times)^2\Omega_1 \\ &+ mg\rho_1 e_3^\times R_1^T e_3 + \tau_1 - R_1^T R_P \tau_{D_1}, \end{aligned} \tag{5.12}$$

$$\begin{aligned} J_P(s)\dot{\Omega}_P = &-\Omega_P \times J_P(s)\Omega_P - N(s,\dot{s})\Omega_P \\ &+ \tau_{D_1} - R_P^T R_2 \tau_{D_2}, \end{aligned} \tag{5.13}$$

$$2m_P\ddot{s} = -L(s,\Omega_P) + u_s, \tag{5.14}$$

$$J_{L_0}\dot{\Omega}_2 = -\Omega_2 \times J_{L_0}\Omega_2 + \tau_{D_2}, \tag{5.15}$$

where, $N(s,\dot{s}) = \dfrac{\mathrm{d}}{\mathrm{d}t}K_P(s) = 4m_P \,\mathrm{diag}\,\{s_2\dot{s}_2 + s_3\dot{s}_3, s_1\dot{s}_1 + s_3\dot{s}_3, s_1\dot{s}_1 + s_2\dot{s}_2\}$,

$$\text{and, } L(s,\Omega_P) = \frac{\partial}{\partial s}(\frac{1}{2}\,\Omega_P^T\,K_P\,\Omega_P) = 2m_P \begin{bmatrix} s_1(\Omega_{P_2}^2 + \Omega_{P_3}^2) \\ s_2(\Omega_{P_3}^2 + \Omega_{P_1}^2) \\ s_3(\Omega_{P_1}^2 + \Omega_{P_2}^2) \end{bmatrix}.$$

We can rewrite the above dynamical equations in a matrix form (useful for the impact model) by defining $\boldsymbol{q}_s = \big(\rho_1, R_1, R_P, s, R_2\big)$, and $\boldsymbol{\omega}_s = \begin{bmatrix}\dot{\rho}_1 & \Omega_1 & \Omega_P & \dot{s} & \Omega_2\end{bmatrix}^T$. Thus, using $\boldsymbol{q}_s, \boldsymbol{\omega}_s$, the equations of motion can be rewritten as

$$\boldsymbol{D}_s(\boldsymbol{q}_s)\dot{\boldsymbol{\omega}}_s = \boldsymbol{H}_s(\boldsymbol{q}_s, \boldsymbol{\omega}_s) + \boldsymbol{B}_s\boldsymbol{u}_s,$$

where, $\boldsymbol{D}_s(\boldsymbol{q}_s) = diag\,(m, J_1(\rho_1), J_P(s), 2m_P I, J_{L_0})$, and

$$\boldsymbol{H}_s(\boldsymbol{q}_s, \boldsymbol{\omega}_s) = \begin{bmatrix} -m\rho_1\Omega_1^T(e_3^\times)^2\Omega_1 + mge_3^T R_1^T e_3 \\ -\Omega_1 \times J_1(\rho_1)\Omega_1 + 2m\rho_1\dot{\rho}_1(e_3^\times)^2\Omega_1 + mg\rho_1 e_3^\times R_1^T e_3 \\ -\Omega_P \times J_P(s)\Omega_P + 4\sum_{i=1}^3 m_P s_i\dot{s}_i(e_i^\times)^2\Omega_P \\ -L(s,\Omega_P) \\ -\Omega_2 \times J_{L_0}\Omega_2 \end{bmatrix},$$

$$\boldsymbol{B}_s(\boldsymbol{q}_s) = \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & -R_1^T R_P & 0 & 0 \\ 0 & 0 & I & 0 & -R_P^T R_2 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}, \boldsymbol{u}_s = \begin{bmatrix} f_1 \\ \tau_1 \\ \tau_{D1} \\ u_s \\ \tau_{D2} \end{bmatrix}.$$

***Remark* 5.1.2.** *Note that the stance dynamics of the Reaction Mass Biped is fully actuated due to the ankle torque τ_1 at the stance foot.*

Extended Dynamics or Floating-base Robot Model

Having derived the stance dynamics of the Reaction Mass Biped system, where the stance leg is pinned to the ground, we now develop the extended model, where the foot is no longer pinned to the ground. This model is required to formulate the discrete-time impact model that captures the

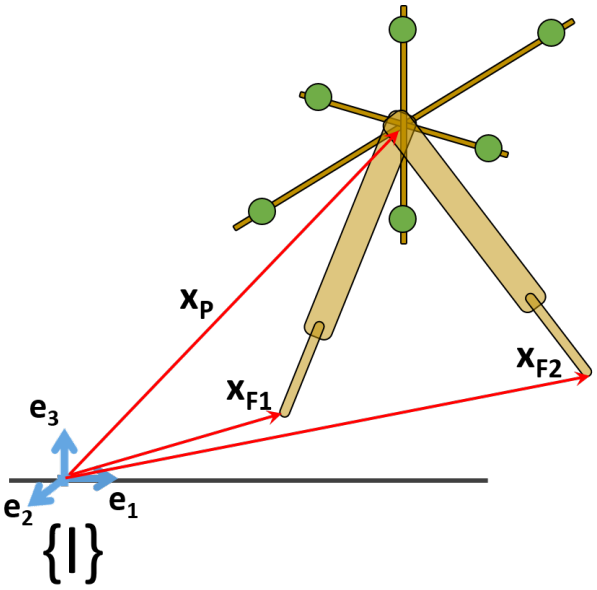


Figure 5.2: Reaction Mass Biped Model for the Extended Dynamics or the Floating base case. Here, x_P is the hip position while x_{F1}, x_{F2} are the stance and swing leg positions, respectively. In the flight phase, we assume that both $\rho_1 = \rho_2 = \rho_0$ i.e., both legs don't extend but only rotate.

dynamics of swing foot impact with the ground. The extended model is illustrated in Figure 5.2, and has the configuration manifold $\mathcal{Q}_e = \mathbb{R}^3 \times SO(3) \times SO(3) \times S \times SO(3)$. The kinetic and potential energies, $\mathcal{T}_e : T\mathcal{Q}_e \to \mathbb{R}, \mathcal{U}_e : Q_e \to \mathbb{R}$ can be derived in a similar manner as in the stance dynamics, resulting in,

$$\mathcal{T}_e = \frac{1}{2} m \dot{x}_P \cdot \dot{x}_P + \sum_{i=1}^{3} m_P \dot{s}_i^2 + \frac{1}{2} \Omega_1^T J_{L_0} \Omega_1 + \frac{1}{2} \Omega_P^T J_P(s) \Omega_P + \frac{1}{2} \Omega_2^T J_{L_0} \Omega_2, \tag{5.16}$$

$$\mathcal{U}_e = -mg x_P \cdot e_3. \tag{5.17}$$

The dynamics of motion can be obtained through application of the Lagrange-d'Alembert principle as outlined earlier. We will directly write this in matrix form by first defining, $\boldsymbol{q}_e = \left(R_1, R_P, s, R_2, x_P\right)$, and $\boldsymbol{\omega}_e = \begin{bmatrix} \Omega_1 & \Omega_P & \dot{s} & \Omega_2 & \dot{x}_P \end{bmatrix}^T$, where x_P is the position of the hip in the inertial frame. We then have,

$$\boldsymbol{D}_e(\boldsymbol{q}_e)\dot{\boldsymbol{\omega}}_e = \boldsymbol{H}_e(\boldsymbol{q}_e, \boldsymbol{\omega}_e) + \boldsymbol{B}_e(\boldsymbol{q}_e)\boldsymbol{u}_e,$$

where, $\boldsymbol{D}_e(\boldsymbol{q}_e) = diag\left(J_{L_0}, J_P(s), 2m_P I, J_{L_0}, m\right)$,

$$\boldsymbol{H}_e(\boldsymbol{q}_e, \boldsymbol{\omega}_e) = \begin{bmatrix} -\Omega_1 \times J_{L_0}\Omega_1 \\ -\Omega_P \times J_P(s)\Omega_P + 4\sum_{i=1}^{3} m_P s_i \dot{s}_i (e_i^{\times})^2 \Omega_P \\ -L(s, \Omega_P) \\ -\Omega_2 \times J_{L_0}\Omega_2 \\ mge_3 \end{bmatrix}, \tag{5.18}$$

$$\boldsymbol{B}_e(\boldsymbol{q}_e) = \begin{bmatrix} -R_1^T R_P & 0 & 0 \\ I & 0 & -R_P^T R_2 \\ 0 & I & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{u}_e = \begin{bmatrix} \tau_{D1} \\ u_s \\ \tau_{D2} \end{bmatrix}. \tag{5.19}$$

***Remark* 5.1.3.** *Note that the extended dynamical model of the reaction mass biped is underactuated. This is in contrast to the stance dynamical model, which is fully actuated.*

Impact Model

We will next develop the discrete-time impact model that captures the impact of the swing foot with the ground. The impact model results in an instantaneous change in the joint velocities of the system. In order to capture this, we will first need to map the stance coordinates to the extended coordinates, perform the impact in the extended coordinates, map the extended coordinates to the stance coordinates while accounting for the relabeling that occurs as the old swing leg becomes the new stance leg.

First, to map the stance coordinates to the extended coordinates, we need to find $x_P, \dot{x}_P$ in terms of the stance coordinates. Since the stance foot is on the ground, $x_P = \rho_1 R_1 e_3$. From this we obtain, $\dot{x}_P = \dot{\rho}_1 R_1 e_3 + \rho_1 R_1 \Omega_1^{\times} e_3$. We will write this as the following map,

$$\boldsymbol{q}_e = \Upsilon^{\boldsymbol{q}}_{s\to e}(\boldsymbol{q}_s), \quad \boldsymbol{\omega}_e = \Upsilon^{\boldsymbol{\omega}}_{s\to e}(\boldsymbol{\omega}_s).$$

For later use, we will denote the map from the extended coordinates to the stance coordinates (assuming the first leg's foot is in contact with the ground) as,

$$\boldsymbol{q}_s = \Upsilon^{\boldsymbol{q}}_{e\to s}(\boldsymbol{q}_e), \quad \boldsymbol{\omega}_s = \Upsilon^{\boldsymbol{\omega}}_{e\to s}(\boldsymbol{\omega}_e).$$

This map essentially computes ρ_1 from x_P as, $\rho_1 = ||x_P||$.

Next we model the impact map. By considering $(\boldsymbol{q}_e^-, \boldsymbol{\omega}_e^-)$ to be the state prior to impact, and $(\boldsymbol{q}_e^+, \boldsymbol{\omega}_e^+)$ to be the state post impact, and F_{ext} representing the external force, we have the following relation from [138],

$$\boldsymbol{D}(\boldsymbol{q}_e^+)\boldsymbol{\omega}_e^+ - \boldsymbol{D}(\boldsymbol{q}_e^-)\boldsymbol{\omega}_e^- = F_{ext}.$$

Further, the swing foot position and velocity are given as,

$$x_{F_2} = x_P + \rho_2^{td} R_2 e_3, \quad \dot{x}_{F_2} = \dot{x}_P - \rho_2^{td} R_2 (e_3)^{\times} \Omega_2,$$

where ρ_2^{td} is the value of ρ_2 at touchdown (note that ρ_2^{td} is a constant and has no dynamics since ρ_2 can instantaneously change.) We require the post impact swing foot velocity $\dot{x}_{F_2}^+ = 0$, since this foot now becomes the new stance foot. This can be expressed as $A\boldsymbol{\omega}_e^+ = 0$, where,

$$A = \begin{bmatrix} 0 & 0 & 0 & -\rho_2^{td} R_2^+ (e_3)^{\times} & I \end{bmatrix}.$$

Further, denoting I_R as the impact force at the swing foot, we have $F_{ext} = A^T I_R$. The above equations can then be expressed in matrix form to solve for $\boldsymbol{\omega}_e^+$ and I_R,

$$\begin{bmatrix} \boldsymbol{\omega}_e^+ \\ I_R \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}_e(\boldsymbol{q}_e^+) & -A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\boldsymbol{D}_e(\boldsymbol{q}_e^-)\boldsymbol{\omega}_e^- \\ 0 \end{bmatrix}. \tag{5.20}$$

We can then define a map Γ such that $\boldsymbol{\omega}_e^+ = \Gamma(\boldsymbol{\omega}_e^-)$.

The impact map can then be defined by the map $\Delta_{s\to s} : \mathcal{S} \to TQ$, where $\mathcal{S} = \{x_s \in TQ_s \mid (\rho_2 R_2 e_3 - \rho_1 R_1 e_3) \cdot e_3 = 0\}$ is the switching surface representing the contact of the swing leg toe with the ground. We have,

$$\Delta_{s\to s} := \begin{bmatrix} \Delta^{\boldsymbol{q}}_{s\to s} \\ \Delta^{\boldsymbol{\omega}}_{s\to s} \end{bmatrix},$$

where, the components $\Delta^{\boldsymbol{q}}_{s\to s}$ and $\Delta^{\boldsymbol{\omega}}_{s\to s}$ define the transition maps for the configuration variables and their velocities, respectively. These are obtained from the above equations as follows:

$$\begin{aligned} \Delta^{\boldsymbol{q}}_{s\to s} &:= \Upsilon^{\boldsymbol{q}}_{e\to s} \circ \mathcal{R} \circ \Upsilon^{\boldsymbol{q}}_{s\to e}, \\ \Delta^{\boldsymbol{\omega}}_{s\to s} &:= \Upsilon^{\boldsymbol{\omega}}_{e\to s} \circ \mathcal{R} \circ \Gamma \circ \Upsilon^{\boldsymbol{\omega}}_{s\to e}, \end{aligned}$$

where $\mathcal{R}$ represents a coordinate relabeling transformation such that the old swing leg is labeled as the new stance leg and vice-versa.

Hybrid System Model

The hybrid model for walking is based on the stance dynamics and the impact model developed in the previous sections, and can be represented as follows:

$$\Sigma : \begin{cases} \boldsymbol{D}_s \dot{\boldsymbol{\omega}}_s = \boldsymbol{H}_s(\boldsymbol{q}_s, \boldsymbol{\omega}_s) + \boldsymbol{B}_s(\boldsymbol{q}_s)\boldsymbol{u}_s, & (\boldsymbol{q}_s^-, \boldsymbol{\omega}_s^-) \notin \mathcal{S}, \\ (\boldsymbol{q}_s^+, \boldsymbol{\omega}_s^+) = \Delta_{s\to s}(\boldsymbol{q}_s^-, \boldsymbol{\omega}_s^-), & (\boldsymbol{q}_s^-, \boldsymbol{\omega}_s^-) \in \mathcal{S}. \end{cases}$$

5.1.1 Discrete Mechanics and Variational Integrator for RMB

In general, for hybrid dynamical models like the RMB, conventional numerical integrators, based on explicit Runge-Kutta method, are used for determining the system's flow based on the continuous-time Euler-Lagrange equations that were derived in Section 5.1. However, this procedure results in the loss of some fundamental geometric properties of the system such as, inherent manifold structure, symplecticity, and the momentum map. Special integrators exist to either preserve manifold structure of the configuration space [139, 140] or simplecticity [141, 142]. Recently, Lee et al [143, 144] integrated these two techniques and devised Geometric Variational Integrators (GVI) that are capable of preserving both geometry and structure of the discrete flow. In this section, we develop a discrete mechanics model of the RMB by taking variations of the corresponding discrete action sum. The resulting update rules form the discrete equations of motion and they used to construct a GVI for the RMB system.

Discrete Lagrangian

The Lagrangian is discretized with a fixed step size, $h \geq 0$, and the subscript k determines the value at any iteration, as $t_k = kh$. Therefore, the configuration manifold of the RMB at any time t_k is given as $\mathcal{Q}_s = C \times SO(3) \times SO(3) \times SO(3) \times S \times C$, with configuration variables $\rho_{1_k}, \rho_{2_k} \in C = [0, r]$, $R_{1_k}, R_{P_k}, R_{2_k} \in SO(3)$, $s_k = \begin{bmatrix} s_{1_k} & s_{2_k} & s_{3_k} \end{bmatrix}^T \in S$.

For the discrete-time kinematic relations, the linear velocity $\dot{x}_k$ at t_k can be approximated as shown

$$\dot{x}_k \approx \frac{\Delta x_k}{h} = \frac{x_{k+1} - x_k}{h}. \tag{5.21}$$

Similarly, from the rotational kinematic relations in Section 5.1, the discrete analogue of angular velocity is

$$F_k = e^{h\Omega_k^\times}, \text{ so that } R_{k+1} = R_k F_k. \tag{5.22}$$

Using (5.22) and (5.21), we get the following first order discrete equations for the RMB,

$$R_{1_{k+1}} = R_{1_k} F_{1_k},\ R_{P_{k+1}} = R_{P_k} F_{P_k},\ R_{2_{k+1}} = R_{2_k} F_{2_k}, \tag{5.23}$$

$$\rho_{1_{k+1}} = \rho_{1_k} + h\dot{\rho}_{1_k},\ s_{k+1} = s_k + h\dot{s}_k. \tag{5.24}$$

$F_{1_k} \in SO(3)$	Rotation matrix that shifts the stance leg from configuration R_{1_k} to $R_{1_{k+1}}$ during k^{th} time-step
$F_{P_k} \in SO(3)$	Rotation matrix that shifts the torso from configuration R_{P_k} to $R_{P_{k+1}}$ during k^{th} time-step
$F_{2_k} \in SO(3)$	Rotation matrix that shifts the swing leg from configuration R_{2_k} to $R_{2_{k+1}}$ during k^{th} time-step

Table 5.2: Notations used in the discrete mechanics of RMB

The symbols specific to the discrete mechanics are tabulated in Table 5.2. We derive the discrete versions of both kinetic ($\mathcal{T}_s$) and potential ($\mathcal{U}_s$) energies of the system next. The discrete version of potential energy $\mathcal{U}_{s_k} : \mathcal{Q}_s \to \mathbb{R}$ is given as $\mathcal{U}_{s_k} = -mg\rho_{1_k} e_3^T R_{1_k}^T e_3$, and the kinetic energy, defined as $\mathcal{T}_{s_k} : T\mathcal{Q}_s \to \mathbb{R}$, is given as,

$$\mathcal{T}_{s_k} = \frac{1}{2}(m\dot{\rho}_{1_k}^2 + \sum_{i=1}^{3} m_P \dot{s}_{i_k}^2 + \Omega_{1_k}^T J_1(\rho_{1_k})\Omega_{1_k} + \quad \Omega_{P_k}^T J_P(s_k)\Omega_{P_k} + \Omega_{2_k}^T J_{L_0}\Omega_{2_k}). \tag{5.25}$$

The discrete Lagrangian $\mathcal{L}_k$ approximates the path of least action, which is obtained by integrating the Lagrangian along the exact solution of the equations of motion for a single time step,

$$\mathcal{L}_k \approx \int_0^h L dt = L(\rho_{1_k}, s_k, \Omega_{1_k}, \Omega_{P_k}, \Omega_{2_k})h = \mathcal{T}_{s_k} - \mathcal{U}_{s_k}. \tag{5.26}$$

Substituting (5.25) in (5.26) gives

$$\mathcal{L}_k = \frac{h}{2}[\dot{\rho}_{1_k}\ \dot{s}_k\ \Omega_{1_k}\ \Omega_{P_k}\ \Omega_{2_k}]^T \begin{bmatrix} m & & & & & \\ & 2m_P I & & & & \\ & & J_1(\rho_{1_k}) & & & \\ & & & J_P(s_k) & & \\ & & & & J_{L_0} & \end{bmatrix} \begin{bmatrix} \dot{\rho}_{1_k} \\ \dot{s}_k \\ \Omega_{1_k} \\ \Omega_{P_k} \\ \Omega_{2_k} \end{bmatrix} + hmg\rho_{1_k} e_3^T R_{1_k}^T e_3, \tag{5.27}$$

where $J_1(\rho_{1_k}) = J_{L_0} - m\rho_{1_k}^2 (e_3^\times)^2$, $J_P(s_k) = J_{P_0} - 2\sum_{i=1}^{3} m_P s_{i_k}^2 (e_i^\times)^2$. Similar to its continuous-time counterpart, the discrete-time version of the Lagrange-d'Alembert principle states that the action sum, which approximates the action integral, is invariant to the first order of all possible

variations, as shown in (5.28). Integrators that maintain this invariance are called Variational Integrators. Additionally, if they also maintain the structure of the configuration manifold, they are called GVIs.

$$\delta \mathcal{S}_d = \sum_{k=0}^{N-1} \delta \mathcal{L}_k + \delta \mathcal{W}_k = 0, \quad \text{where} \quad N = \frac{t_F - t_0}{h}. \tag{5.28}$$

Here, t_0, t_F are the start and end times of the integration, respectively. To compute the above, we have to first determine the infinitesimal variations for R_k and Ω_k, as shown in [70],

$$\delta R_k = \lim_{\epsilon \to 0} R_k \exp\left(\epsilon \eta_k^\times\right) = R_k \eta_k^\times, \tag{5.29}$$

$$\delta F_k = h \delta \Omega_k^\times \exp\left(h \Omega_k^\times\right) = h \delta \Omega_k^\times F_k, \tag{5.30}$$

$$\implies \delta \Omega_k^\times = \frac{1}{h} \delta F_k F_k^T = \frac{1}{h}((F_k \eta_{k+1})^\times - \eta_k^\times),$$

$$\therefore \delta \Omega_k = \frac{1}{h}((F_k \eta_{k+1}) - \eta_k). \tag{5.31}$$

Additionally, the variations of $\dot{\rho}_{1_k}$ and $\dot{s}_k$ are,

$$\delta \dot{\rho_{1_k}} = \frac{\delta \rho_{1_{k+1}} - \delta \rho_{1_k}}{h}, \; \delta \dot{s_k} = \frac{\delta s_{k+1} - \delta s_k}{h}. \tag{5.32}$$

The infinitesimal virtual work done and Lagrangian can be discretized using the discrete infinitesimal variations obtained above as follows:

$$\delta \mathcal{W}_k = h(f_{1_k} \delta \rho_{1_k} + u_{s_k}^T \delta s_k + \tau_{1_k}^T \eta_{1_k} + \tau_{D_{1_k}}^T \eta_{P1_k} + \tau_{D_{2_k}}^T \eta_{P2_k}),$$

$$\delta \mathcal{L}_k = h[\dot{\rho}_{1_k} \; \dot{s}_k \; \Omega_{1_k} \; \Omega_{P_k} \; \Omega_{2_k}]^T \begin{bmatrix} m & & & & & \\ & 2m_P I & & & & \\ & & J_1(\rho_{1_k}) & & & \\ & & & J_P(s_k) & & \\ & & & & J_{L_0} \end{bmatrix} \begin{bmatrix} \delta \dot{\rho}_{1_k} \\ \delta \dot{s}_k \\ \delta \Omega_{1_k} \\ \delta \Omega_{P_k} \\ \delta \Omega_{2_k} \end{bmatrix}$$

$$-hm\rho_{1_k} \Omega_{1_k}^T (e_3^\times)^2 \Omega_{1_k} \delta \rho_{1_k} - h2m_P \textstyle\sum_{i=1}^3 s_{i_k} \Omega_{P_k}^T (e_i^\times)^2 \Omega_{P_k} \delta s_{i_k}$$

$$+hmg \delta \rho_{1_k} e_3^T R_{1_k}^T e_3 + hmg \rho_{1_k} e_3^T \delta R_{1_k}^T e_3. \tag{5.33}$$

Now, we map all the velocities to their corresponding momentum terms and continue the rest of this derivation in terms of the momenta. Let, $p_{\rho_{1_k}} = m\dot{\rho}_{1_k}$, $p_{s_k} = 2m_P \dot{s}_k$, $\Pi_{1_k} = J_1(\rho_{1_k})\Omega_{1_k}$, $\Pi_{P_k} = J_P(s_{P_k})\Omega_{P_k}$, and $\Pi_{2_k} = J_{L_0}\Omega_{2_k}$. Accordingly, the discrete action sum in (5.28) can be rewritten using (5.33) as,

$$\delta \mathcal{S}_d = \textstyle\sum_{k=0}^{N-1} [\Pi_{1_k}^T \delta \Omega_{1_k} + M_k^T \delta R_{1_k} + \Pi_{P_k}^T \delta \Omega_{P_k} + \Pi_{2_k}^T \delta \Omega_{2_k} +$$

$$p_{\rho_{1_k}}(\delta \rho_{1_{k+1}} - \delta \rho_{1_k}) - hm\rho_{1_k} \Omega_{1_k}^T (e_3^\times)^2 \Omega_{1_k} \delta \rho_{1_k} + N_k \delta \rho_{1_k} +$$

$$\textstyle\sum_{i=1}^3 p_{s_{i_k}}(\delta s_{i_{k+1}} - \delta s_{i_k}) - h2m_P s_{i_k} \Omega_{P_k}^T (e_i^\times)^2 \Omega_{P_k} \delta s_{i_k} + \delta \mathcal{W}_k] = 0. \tag{5.34}$$

where, $M_k = \frac{\partial \mathcal{U}_{s_k}}{\partial R_{1_k}}$ and $N_k = \frac{\partial \mathcal{U}_{s_k}}{\partial \rho_{1_k}}$. We can substitute (5.33), and the variations (5.29),(5.31), and (5.32) in (5.34) to obtain the discrete action sum in terms of the variations $\delta \upsilon_k := [\eta_{1_k}\ \eta_{P_k}\ \eta_{2_k}\ \delta\rho_{1_k}\ \delta s_k]$. The fact that variations vanish at end points, i.e., $\delta\upsilon_k = 0$ if $k = \{0, N\}$, and an appropriate re-indexing of terms allows us to reformulate (5.34) as,

$$\begin{aligned}
\sum_{k=1}^{N-1}[(F_{1_{k-1}}^T \Pi_{1_{k-1}} - \Pi_{1_k} + M_k^T R_{1_k} + h\tau_{1_k} - hR_{P1_k}\tau_{D1_k})^T \eta_{1_k} \\
(F_{P_{k-1}}^T \Pi_{P_{k-1}} - \Pi_{P_k} + h\tau_{D1_k} - hR_{P2_k}\tau_{D2_k})^T \eta_{p_k} + \\
(F_{2_{k-1}}^T \Pi_{2_{k-1}} - \Pi_{2_k} + h\tau_{D2_k})^T \eta_{2_k} + \\
(p_{\rho_{1_{k-1}}} - p_{\rho_{1_k}} - hm\rho_{1_k}\Omega_{1_k}^T (e_3^\times)^2 \Omega_{1_k} + N_k + hf_{1_k})\delta\rho_{1_k} + \\
\sum_{i=1}^{3}(p_{s_{i_{k-1}}} - p_{s_{i_k}} - h2m_P s_{i_k}\Omega_{P_k}^T (e_i^\times)^2 \Omega_{P_k} + hu_{i_k})\delta s_{i_k}] = 0.
\end{aligned} \tag{5.35}$$

Since (5.35) is true for any $\delta\upsilon_k$, we require that the expressions each of in the parentheses to be equal to zero. They are indeed the discrete-time equations of motion for the RMB in terms of the momenta. Finally, we can map back from the momentum terms to the velocity terms to get the equations of motion in terms of the velocities as shown below:

$$J_{1_{k+1}}\Omega_{1_{k+1}} = F_{1_k}^T(J_{1_k}\Omega_{1_k}) + hmg\rho_{1_{k+1}}e_3^\times R_{1_{k+1}}^T e_3 + h\tau_{1_{k+1}} - hR_{P1_{k+1}}\tau_{D1_{k+1}} \tag{5.36}$$

$$J_{P_{k+1}}\Omega_{P_{k+1}} = F_{P_k}^T(J_{P_k}\Omega_{P_k}) + h\tau_{D1_{k+1}} - hR_{P2_{k+1}}\tau_{D2_{k+1}} \tag{5.37}$$

$$J_{L_0}\Omega_{2_{k+1}} = F_{2_k}^T(J_{L_0}\Omega_{2_k}) + h\tau_{D2_{k+1}} \tag{5.38}$$

$$m\dot{\rho}_{1_{k+1}} = m\dot{\rho}_{1_k} - hm\rho_{1_{k+1}}\Omega_{1_{k+1}}^T(e_3^\times)^2\Omega_{1_{k+1}} + hmge_3^T R_{1_{k+1}}^T e_3 + hf_{1_{k+1}} \tag{5.39}$$

$$2m_P\dot{s}_{k+1} = 2m_P\dot{s}_k - hL(s_{k+1}, \Omega_{P_{k+1}}) + hu_{k+1} \tag{5.40}$$

The discrete-time Lagrangian flow map takes us from
$(\Omega_{1_k}\ \Omega_{P_k}\ \Omega_{2_k}\ \dot{\rho}_{1_k}\ \dot{s}_k) \mapsto (\Omega_{1_{k+1}}\ \Omega_{P_{k+1}}\ \Omega_{2_{k+1}}\ \dot{\rho}_{1_{k+1}}\ \dot{s}_{k+1})$, and this process is repeated for N steps. Note that, unlike in [143, 144], this is an explicit method and doesn't require custom Rodrigues formula-based gradient descent methods, and is therefore faster.

Advantages of Geometric Variational Integrators are listed below.

1. GVIs preserve important mechanical properties like energy conservation (for conservative systems), momentum conservation (where there is symmetry), while ensuring that the dynamics evolves in the configuration manifold of the system.

2. They can be easily implemented in hardware and the equations are inherently discrete-time.

3. This structure preserving property is also useful when building controllers based on energy-like Lyapunov functions, as shown in this work.

4. Moreover, the performance does not degrade even for long simulation times.

5.1.2 Motion Planning and Control

Motion Planning for Moving between Ground Locations

Consider a trajectory connecting two ground points with known initial and final velocities; there are many ways to generate this trajectory while avoiding fixed obstacles. The motion of the torso center of mass, when projected on the horizontal (ground) plane, should closely follow this generated trajectory. Assuming that this trajectory is known a priori, a stride length that is optimal (or natural) for the RMB is used to determine the number of steps required to cover the path length. If l_s is the optimal stride length and p_l is the path length of the trajectory, then the nearest integer to p_l/l_s can be used as the number of strides required to cover this trajectory.

Desired trajectories (motion primitives) for variables associated with the RMB legs in time interval $[0, T]$ are:

$$\begin{aligned} \rho_1^d &= \rho_0 + \bar{\rho}\sin(\omega t),\ \omega = \frac{\pi}{T},\ \rho_0 > \bar{\rho} > 0, \\ R_1^d &= R_{1_0}\exp\left(\zeta_1^\times \sin(\omega t/2)\right), \\ \rho_2^d &= \rho_0, \\ R_2^d &= R_{2_0}\exp\left(\zeta_2^\times \sin(\omega t/2)\right). \end{aligned} \tag{5.41}$$

Note that the constant vectors $\zeta_1, \zeta_2 \in \mathbb{R}^3$ for the leg rotations could be equal, and something similar could be said for $R_{1_0}, R_{2_0} \in \mathrm{SO}(3)$ when the biped is standing erect. Also, ρ_0 and $\bar{\rho}$ are related to the optimal stride length for the biped. The desired trajectories for variables associated with the torso over the time interval $[0, T]$ are:

$$\begin{aligned} R_P^d &= R_1^d \exp\left(\gamma \log\left((R_1^d)^T R_2^d\right)\right),\ \gamma \in [0, 1], \\ s^d &= s_0 + \bar{s}\sin(\omega t), \end{aligned} \tag{5.42}$$

where $s_0, \bar{s} \in \mathbb{R}^3$ are designed to have the appropriate inertia distribution for the torso as mentioned earlier with $|s_{0i}| > |\bar{s}_i|$, $\log : \mathrm{SO}(3) \to \mathfrak{so}(3)$ is the logarithm map that is inverse of the exponential map (given by the matrix exponential), and γ is a weight factor. Note that $R_P^d = R_1^d$ when $\gamma = 0$ and $R_P^d = R_2^d$ when $\gamma = 1$. The reasoning behind introducing these weights is to mimic human bipedal gait, where the body (torso) becomes more closely aligned with the alignment of the stance leg as the speed of bipedal motion increases. By making γ and ω time-varying, one can even transition between different speeds of bipedal motion. This is one of the future goals of this research.

The stride length is given by the horizontal distance traversed by the ankle joint of the swing leg in one cycle. The desired stride length can be obtained from the above desired motions for the swing leg, considering that the inertial position of the ankle/foot of the swing leg at an instant is given by

$$a_{L2} = a_{L1} + \rho_1 R_1 e_3 - \rho_2 R_2 e_3, \tag{5.43}$$

where a_{L1}, a_{L2} denote the positions of the ankles of the stance and swing leg, respectively. With the coordinate frames as illustrated in Fig. 5.1 and substituting equation (5.41) for the desired

motion trajectories, the starting and end positions of the swing leg ankle during a cycle are:

$$\begin{aligned} a^s_{L2} &= a_{L1} + \rho_0 R_{1_0} e_3 - \rho_0 R_{2_0} e_3, \\ a^e_{L2} &= a_{L1} + \rho_0 R_{1_0} \exp(\zeta_1^\times) e_3 - \rho_0 R_{2_0} \exp(\zeta_2^\times) e_3 \end{aligned} \tag{5.44}$$

Therefore the stride length is given by

$$a^e_{L2} - a^s_{L2} = \rho_0 R_{1_0} \big(\exp(\zeta_1^\times) - I \big) e_3 + \rho_0 R_{2_0} \big(I - \exp(\zeta_2^\times) \big) e_3. \tag{5.45}$$

Using Rodrigues' rotation formula, the above expression can be simplified to

$$\begin{aligned} v^d_s = \rho_0 R_{1_0} \big\{ \hat{\zeta}_1^\times \sin \|\zeta_1\| + \big(\hat{\zeta}_1^\times\big)^2 (1 - \cos \|\zeta_1\|) \big\} e_3 \\ - \rho_0 R_{2_0} \big\{ \hat{\zeta}_2^\times \sin \|\zeta_2\| + \big(\hat{\zeta}_2^\times\big)^2 (1 - \cos \|\zeta_1\|) \big\} e_3, \end{aligned} \tag{5.46}$$

where v^d_s denotes the desired stride vector, and $\hat{\zeta}_1, \hat{\zeta}_2$ denote the unit vectors along ζ_1, ζ_2 respectively. This sets the desired stride length to $l^d_s = \|v^d_s\|$. Note that the constraint $e_3^T v^d_s = 0$ must be satisfied, which imposes certain constraints on R_{1_0}, R_{2_0}, ζ_1 and ζ_2. Substituting (5.46) for v^d_s, this constraint is expressed as

$$\begin{aligned} &\Gamma_{1_0}^T \big\{ \hat{\zeta}_1^\times \sin \|\zeta_1\| + \big(\hat{\zeta}_1^\times\big)^2 (1 - \cos \|\zeta_1\|) \big\} e_3 = \\ &\Gamma_{2_0}^T \big\{ \hat{\zeta}_2^\times \sin \|\zeta_2\| + \big(\hat{\zeta}_2^\times\big)^2 (1 - \cos \|\zeta_1\|) \big\} e_3, \\ &\text{where } \Gamma_{1_0} = R_{1_0}^T e_3, \ \Gamma_{2_0} = R_{2_0}^T e_3. \end{aligned} \tag{5.47}$$

Expression (5.47) can be satisfied by setting

$$\zeta_1 = \zeta_2 \text{ and } R_{2_0} = \exp(\theta e_3^\times) R_{1_0}, \tag{5.48}$$

for $\theta \in \mathbb{S}^1$. The second equality in (5.48) guarantees that $\Gamma_{1_0} = \Gamma_{2_0}$; physically, it means that the initial orientations of the stance and swing legs during start of a cycle are related by a rotation about the inertial third axis that points up. Substituting (5.48) in (5.42) to simplify the expression for R^d_P in (5.42), one obtains:

$$\begin{aligned} (R^d_1)^T R^d_2 &= \exp\big(- c(t) \zeta_1^\times \big) R_{1_0}^T R_{2_0} \exp\big(c(t) \zeta_1^\times \big) \\ &= \exp\big(- c(t) \zeta_1^\times \big) \exp\big(\theta (R_{1_0}^T e_3)^\times \big) \exp\big(c(t) \zeta_1^\times \big) \\ &= \exp\Big(\theta \big(\exp(-c(t) \zeta_1^\times) R_{1_0}^T e_3 \big)^\times \Big), \end{aligned} \tag{5.49}$$

where $c(t) = \sin(\omega t/2)$. The above simplification uses the following relation multiple times:

$$R^T \exp(\phi e^\times) R = \exp\big(\phi (R^T e)^\times \big),$$

where $e \in \mathbb{S}^2$ is a unit vector. This leads to the following simplified expression for R^d_P:

$$R^d_P = R^d_1 \exp\Big(\gamma \theta \big(\exp(-c(t) \zeta_1^\times) R_{1_0}^T e_3 \big)^\times \Big), \tag{5.50}$$

which can then be expanded using Rodrigues' formula.

Trajectory Tracking Control Scheme

Define the trajectory tracking errors:

$$\begin{aligned}
&\tilde{\rho}_1 = \rho_1 - \rho_1^d, \;\; \dot{\tilde{\rho}}_1 = \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\rho}_1, \\
&Q_1 = R_1 (R_1^d)^T, \;\; \tilde{\Omega}_1 = \Omega_1 - \Omega_1^d, \\
&Q_P = R_P (R_P^d)^T, \;\; \tilde{\Omega}_P = \Omega_P - \Omega_P^d, \\
&\tilde{s} = s - s^d, \;\; \dot{\tilde{s}} = \frac{\mathrm{d}}{\mathrm{d}t}\tilde{s}, \\
&Q_2 = R_2 (R_2^d)^T, \;\; \tilde{\Omega}_2 = \Omega_2 - \Omega_2^d.
\end{aligned} \tag{5.51}$$

The trajectory tracking control scheme is a generalization of the control scheme in [145]. The Lyapunov function candidate for the stance leg is:

$$\begin{aligned}
V_{L_1}(\rho_1, \tilde{\rho}_1, Q_1, \dot{\tilde{\rho}}_1, \tilde{\Omega}_1) = &\frac{1}{2} m \dot{\tilde{\rho}}_1^2 + \frac{1}{2} \tilde{\Omega}_1^T J_1(\rho_1) \tilde{\Omega}_1 + \frac{1}{2} k \tilde{\rho}_1^2 \\
&+ \Phi\big(\mathrm{tr}(A - A Q_1)\big),
\end{aligned} \tag{5.52}$$

where $k > 0$, $A = \mathrm{diag}(a_1, a_2, a_3)$ with $a_1 > a_2 > a_3 > 0$, and $\Phi : \mathbb{R} \to \mathbb{R}$ is a $\mathcal{C}^2$ function that satisfies $\Phi(0) = 0$ and $\Phi'(x) > 0$ for all $x \in \mathbb{R}^+$. Furthermore, let $\Phi'(\cdot) \leq \alpha(\cdot)$, where $\alpha(\cdot)$ is a Class-$\mathcal{K}$ function [146]. The Lyapunov function candidate for the torso of the RMB is:

$$\begin{aligned}
V_P(Q_P, \tilde{\Omega}_P, s, \tilde{s}, \dot{\tilde{s}}) = &\frac{1}{2} \tilde{\Omega}_P^T J_P(s) \tilde{\Omega}_P + m_P \dot{\tilde{s}}^T \dot{\tilde{s}} + \frac{1}{2} \tilde{s}^T P \tilde{s} \\
&+ \Phi\big(\mathrm{tr}(A - A Q_P)\big).
\end{aligned} \tag{5.53}$$

The Lyapunov function candidate for the swing leg is:

$$V_{L_2}(Q_2, \tilde{\Omega}_2) = \frac{1}{2} \tilde{\Omega}_2^T J_{L_0} \tilde{\Omega}_2 + \Phi\big(\mathrm{tr}(A - A Q_2)\big), \tag{5.54}$$

where J_{L_0} is the inertia of swing leg at its nominal length (ρ_0), which is kept constant during swing phase, and P, Q_1, Q_P, Q_2 are suitable positive definite matrices that are use to build valid Lyapunov functions. The time derivative of these Lyapunov functions along the stance dynamics of the RMB system are evaluated next.

The time derivative of V_{L_1} along dynamics (5.11)-(5.12) is:

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t} V_{L_1}(\rho_1, \tilde{\rho}_1, Q_1, \dot{\tilde{\rho}}_1, \tilde{\Omega}_1) = \dot{\tilde{\rho}}_1 \Big[f_1 - m \rho_1 \Omega_1^T \big(e_3^\times\big)^2 \Omega_1 \\
&+ m g e_3^T \Gamma_1 - m \ddot{\rho}_1^d + k \tilde{\rho}_1 \Big] \\
&+ \tilde{\Omega}_1^T \Big[- \Omega_1 \times J_1(\rho_1) \Omega_1 + 2 m \rho_1 \dot{\rho}_1 \big(e_3^\times\big)^2 \Omega_1 + m g \rho_1 e_3^\times \Gamma_1 \\
&+ \tau_1 - R_1^T R_P \tau_{D_1} - J_1(\rho_1) \dot{\Omega}_1^d - m \rho_1 \dot{\rho}_1 \big(e_3^\times\big)^2 \tilde{\Omega}_1 \\
&+ \Phi'\big(\mathrm{tr}(A - A Q_1)\big) (R_1^d)^T S(Q_1) \Big],
\end{aligned} \tag{5.55}$$

where $\Gamma_1 = R_1^T e_3$ is the inertial z-axis direction (upwards) in the stance leg's body-fixed frame and $S : \mathrm{SO}(3) \to \mathbb{R}^3$ is defined by

$$S(Q) = \sum_{i=1}^{3} a_i Q^T e_i \times e_i. \tag{5.56}$$

After some partial cancellations of terms, this expression can be rewritten as

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}V_{L_1}(\rho_1, \tilde{\rho}_1, Q_1, \dot{\tilde{\rho}}_1, \tilde{\Omega}_1) &= \dot{\tilde{\rho}}_1\Big[f_1 - m\ddot{\rho}_1^d - m\rho_1\Omega_1\big(e_3^\times\big)^2\Omega_1 + mge_3^T\Gamma_1 + k\tilde{\rho}_1\Big] \\ &+ \tilde{\Omega}_1^T\Big[\tau_1 - R_1^T R_P \tau_{D_1} - \Omega_1^d \times J_1(\rho_1)\Omega_1 + m\rho_1\dot{\rho}_1\big(e_3^\times\big)^2\big(\Omega_1 + \Omega_1^d\big) \\ &+ mg\rho_1 e_3^\times\Gamma_1 + \Phi'\big(\mathrm{tr}(A - AQ_1)\big)(R_1^d)^T S(Q_1)\Big].\end{aligned} \tag{5.57}$$

The time derivative of V_P along the dynamics (5.13)-(5.14) is:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}V_P(Q_P, \tilde{\Omega}_P, s, \tilde{s}, \dot{\tilde{s}}) &= \tilde{\Omega}_P^T\Big[J_P(s)\Omega_P \times \Omega_P + \tau_{D_1} - N(s,\dot{s})\Omega_P - R_P^T R_2 \tau_{D_2} \\ &- J_P(s)\dot{\Omega}_P^d + \frac{1}{2}N(s,\dot{s})\tilde{\Omega}_P + \Phi'\big(\mathrm{tr}(A - AQ_P)\big)(R_P^d)^T S(Q_P)\Big] \\ &+ \dot{\tilde{s}}^T\Big[L(s,\Omega_P) - 2m_P\ddot{s}^d + P\tilde{s} + u_s\Big],\end{aligned} \tag{5.58}$$

where $N(s,\dot{s}) = \frac{\mathrm{d}}{\mathrm{d}t}K_P(s)$ and $\dot{s}^T L(s,\Omega_P) = \frac{1}{2}\Omega_P^T N(s,\dot{s})\Omega_P$. After some partial cancellation of terms, one can simplify expression (5.58) to

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}V_P(Q_P, \tilde{\Omega}_P, s, \tilde{s}, \dot{\tilde{s}}) &= \tilde{\Omega}_P^T\Big[\tau_{D_1} - R_P^T R_2\tau_{D_2} - J_P(s)\dot{\Omega}_P^d - \Omega_P^d \times J_P(s)\Omega_P \\ &- \frac{1}{2}N(s,\dot{s})\big(\Omega_P + \Omega_P^d\big) + \Phi'\big(\mathrm{tr}(A - AQ_P)\big)(R_P^d)^T S(Q_P)\Big] \\ &+ \dot{\tilde{s}}^T\Big[u_s + L(s,\Omega_P) - 2m_P\ddot{s}^d + P\tilde{s}\Big].\end{aligned} \tag{5.59}$$

Finally, the time derivative of V_{L_2} along dynamics (5.15) is

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}V_{L_2}(Q_2, \tilde{\Omega}_2) = \frac{1}{2}\tilde{\Omega}_2^T\Big(&- \Omega_2^d \times J_{L_0}\Omega_2 + \tau_{D_2} - J_{L_0}\dot{\Omega}_2^d \\ &+ \Phi'\big(\mathrm{tr}(A - AQ_2)\big)(R_2^d)^T S(Q_2)\Big).\end{aligned} \tag{5.60}$$

Theorem 5.1.1. *Let $\ell > 0$ and let $D_1, D_P, D_2, P, Q \in \mathbb{R}^{3\times 3}$ be positive definite matrices. Then*

the tracking control laws

$$f_1 = m\ddot{\rho}_1^d + m\rho_1\Omega_1^T\left(e_3^\times\right)^2\Omega_1 - mge_3^T\Gamma_1 - k\tilde{\rho}_1 - \ell\dot{\tilde{\rho}}_1, \tag{5.61}$$

$$\begin{aligned}\tau_1 &= R_1^T R_P \tau_{D_1} + J_1(\rho_1)\dot{\Omega}_1^d + \Omega_1^d \times J_1(\rho_1)\Omega_1 - mg\rho_1 e_3^\times \Gamma_1 \\ &\quad - m\rho_1\dot{\rho}_1\left(e_3^\times\right)^2\left(\Omega_1 + \Omega_1^d\right) - \Phi'\big(tr(A - AQ_1)\big)(R_1^d)^T S(Q_1) - D_1\tilde{\Omega}_1,\end{aligned} \tag{5.62}$$

$$\begin{aligned}\tau_{D_1} &= R_P^T R_2 \tau_{D_2} + \Omega_P^d \times J_P(s)\Omega_P + J_P(s)\dot{\Omega}_P^d - D_P\tilde{\Omega}_P \\ &\quad + \frac{1}{2}N(s,\dot{s})\left(\Omega_P + \Omega_P^d\right) - \Phi'\big(tr(A - AQ_P)\big)(R_P^d)^T S(Q_P),\end{aligned} \tag{5.63}$$

$$u_s = 2m_P\ddot{s}^d - L(s,\Omega_P) - P\tilde{s} - Q\dot{\tilde{s}}, \tag{5.64}$$

$$\tau_{D_2} = \Omega_2^d \times J_{L_0}\Omega_2 + J_{L_0}\dot{\Omega}_2^d - D_2\tilde{\Omega}_2 - \Phi'\big(tr(A - AQ_2)\big)(R_2^d)^T S(Q_2), \tag{5.65}$$

asymptotically stabilize a desired state trajectory of the form given by equations (5.41)-(5.42)*. Further, the trajectory's domain of convergence is almost global in the state space in the absence of control constraints, force/torque disturbances.*

Proof: Consider the Lyapunov function

$$\begin{aligned}V(\tilde{\rho}_1, Q_1, \dot{\tilde{\rho}}_1, \tilde{\Omega}_1, Q_P, \Omega_P, s, \dot{s}, Q_2, \tilde{\Omega}_2) &= V_{L_1}(\rho_1, \tilde{\rho}_1, Q_1, \dot{\tilde{\rho}}_1, \tilde{\Omega}_1) \\ &\quad + V_P(Q_P, \tilde{\Omega}_P, s, \tilde{s}, \dot{\tilde{s}}) + V_{L_2}(Q_2, \tilde{\Omega}_2),\end{aligned} \tag{5.66}$$

which captures the three coupled components: the stance leg, torso, and swing leg, of the RMB. The time derivative of this Lyapunov function is obtained by substituting expressions (5.67), (5.68) and (5.69) for the time derivatives of V_{L_1}, V_P, and V_{L_2} respectively. Further substitutions of the control laws (5.61)-(5.65) in these expressions gives the time derivatives along trajectories of the feedback tracking system

$$\dot{V}_{L_1}(\dot{\tilde{\rho}}_1, \tilde{\Omega}_1) = -\ell\dot{\tilde{\rho}}_1^2 - \tilde{\Omega}_1^T L_1 \tilde{\Omega}_1, \tag{5.67}$$

$$\dot{V}_P(\tilde{\Omega}_P, \dot{\tilde{s}}) = -\tilde{\Omega}_P^T L_P \tilde{\Omega}_P - \dot{\tilde{s}}^T Q \dot{\tilde{s}}, \tag{5.68}$$

$$\dot{V}_{L_2}(\tilde{\Omega}_2) = -\tilde{\Omega}_2^T L_2 \tilde{\Omega}_2. \tag{5.69}$$

This makes the time derivative of the overall Lyapunov function negative semi-definite:

$$\dot{V}(\dot{\tilde{\rho}}_1, \tilde{\Omega}_1, \tilde{\Omega}_P, \dot{\tilde{s}}, \tilde{\Omega}_2) = -\ell\dot{\tilde{\rho}}_1^2 - \tilde{\Omega}_1^T L_1 \tilde{\Omega}_1 - \tilde{\Omega}_P^T L_P \tilde{\Omega}_P - \dot{\tilde{s}}^T Q \dot{\tilde{s}} - \tilde{\Omega}_2^T L_2 \tilde{\Omega}_2. \tag{5.70}$$

Assuming that the desired motion trajectories are bounded and continuous, as is the case with the desired motions given by (5.41)-(5.42), then V as given by (5.66) is positive definite and is bounded above and below by suitably chosen positive definite functions of the trajectory tracking error states. Therefore, invoking invariance-like principle given by Theorem 8.4 in [146], one can conclude that $\dot{V}$ converges asymptotically to zero. Therefore, the positive limit set for the feedback tracking control system is a subset of

$$\begin{aligned}\dot{V}^{-1}(0) = \big\{&(\tilde{\rho}_1, Q_1, \dot{\tilde{\rho}}_1, \tilde{\Omega}_1, Q_P, \Omega_P, \dot{\tilde{s}}, Q_2, \tilde{\Omega}_2) \,:\, \dot{\tilde{\rho}}_1 = 0, \\ &\tilde{\Omega}_1 = 0,\, \tilde{\Omega}_P = 0,\, \dot{\tilde{s}} = 0,\, \tilde{\Omega}_2 = 0\big\}.\end{aligned} \tag{5.71}$$

The feedback dynamics can be expressed in terms of the tracking errors as follows:

$$m\ddot{\tilde{\rho}}_1 = -\ell\dot{\tilde{\rho}}_1 - k\tilde{\rho}_1, \tag{5.72}$$

$$\begin{aligned} J_1(\rho_1)\dot{\tilde{\Omega}}_1 &= -\tilde{\Omega}_1 \times J_1(\rho_1)\Omega_1 + m\rho_1\dot{\rho}_1\big(e_3^{\times}\big)^2\tilde{\Omega}_1 - D_1\tilde{\Omega}_1 \\ &\quad - \Phi'\big(\mathrm{tr}(A - AQ_1)\big)(R_1^d)^T S(Q_1), \end{aligned} \tag{5.73}$$

$$\begin{aligned} J_P(s)\dot{\tilde{\Omega}}_P &= -\tilde{\Omega}_P \times J_P(s)\Omega_P - \frac{1}{2}N(s,\dot{s})\tilde{\Omega}_P - D_P\tilde{\Omega}_P \\ &\quad - \Phi'\big(\mathrm{tr}(A - AQ_P)\big)(R_P^d)^T S(Q_P), \end{aligned} \tag{5.74}$$

$$2m_P\ddot{\tilde{s}} = -Q\dot{\tilde{s}} - P\tilde{s}, \tag{5.75}$$

$$\begin{aligned} J_{L_0}\dot{\tilde{\Omega}}_2 &= -\tilde{\Omega}_2 \times J_{L_0}\Omega_2 - D_2\tilde{\Omega}_2 \\ &\quad - \Phi'\big(\mathrm{tr}(A - AQ_2)\big)(R_2^d)^T S(Q_2). \end{aligned} \tag{5.76}$$

Therefore in the set $\dot{V}^{-1}(0)$, the feedback dynamics is restricted to

$$\begin{aligned} &\tilde{\rho}_1 = 0,\ \Phi'\big(\mathrm{tr}(A - AQ_1)\big) = 0,\ \Phi'\big(\mathrm{tr}(A - AQ_P)\big) = 0, \\ &\tilde{s} = 0, \text{ and } \Phi'\big(\mathrm{tr}(A - AQ_2)\big) = 0, \end{aligned} \tag{5.77}$$

which characterizes the positive limit set of the feedback tracking system. Note that within the set of four critical points $\mathcal{E}_c$ of $\Phi\big(\mathrm{tr}(A - AQ)\big)$, it can be shown, as in [69, 68, 147], that $Q = I$ is the minimum, while the other points ($Q \in \mathcal{E}_c \setminus I$) are non-degenerate critical points. Therefore, as $\dot{V} \leq 0$ along the trajectories of the feedback system, the only stable subset of the positive limit set is when the actual motion is tracking the desired motion, i.e.,

$$\tilde{\rho}_1 = 0,\ Q_1 = I,\ Q_P = I,\ \tilde{s} = 0, \text{ and } Q_2 = I. \tag{5.78}$$

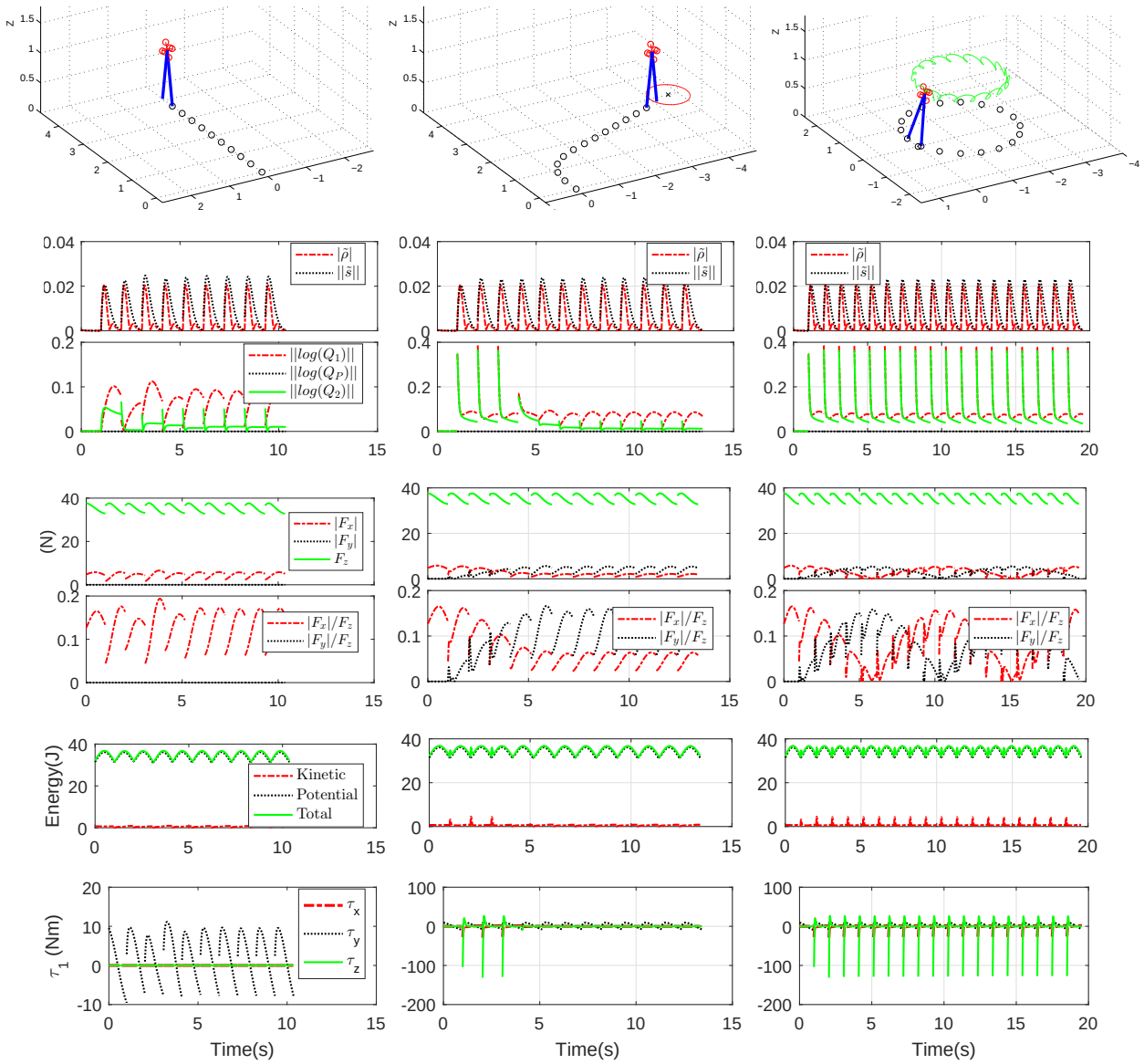
The other subsets (corresponding to $Q_1, Q_P, Q_2 \in \mathcal{E}_c \setminus I$) are unstable, although they may have stable subsets. Except for trajectories that start on these stable subsets of the positive limit set, all other trajectories in the state space converge asymptotically to the desired state trajectory. This means that the set $\mathcal{S}_L$ is asymptotically stable and its domain of attraction is almost-global. □

Note that, this trajectory tracking control scheme can be applied in general to track all $\mathcal{C}^2$ desired state trajectories, provided that actuator constraints are not violated. In practice, the desired state trajectories can be designed keeping in mind known actuator constraints for the RMB or for a humanoid robot being modeled by the RMB.

5.2 Numerical Results

Having developed a geometric controller for asymptotically tracking trajectories, we now validate the proposed controller through a numerical simulation of the hybrid model developed in Section 5.1.

To illustrate the capability of the controller, we will demonstrate (a) walking in a straight line, (b) walking towards a goal location, and (c) walking in a circle. In all cases, we choose a constant desired torso angle leaning forward, this is in contrast to (5.42) to simplify velocity and acceleration



(a) Walking Straight (b) Walking to a Goal (c) Walking in a Circle

Figure 5.3: Numerical simulations of the controller for (a) Walking along a straight line, (b) Walking towards a goal location by changing the yaw-angle in an event-based step-to-step manner, and (c) Walking in a circle while leaning inwards, with the hip trajectory shown in green. For all these cases, first row shows simulation snapshots. The second row shows error plots to study controller behavior. The errors include those defined in (5.51). Third row shows ground reaction force plots. Since the legs have point contact with the ground, we assume the friction forces (F_x and F_y) to be isotropic, and $\frac{|F_x|}{F_z} \leq 0.6$ and $\frac{|F_y|}{F_z} \leq 0.6$. Assuming the coefficient of static friction to be greater than 0.6, RMB satisfies the no slip condition at the stance leg for all the three trajectories. Fourth row shows the energy plots for the closed-loop dynamics of the RMB walking. Finally, in the the fifth row, we show the ankle torque (τ_1) generated for the three walking trajectories. The x-axis for all the plots is Time(in seconds).

computation. The mass and inertia properties of the reaction mass biped are chosen to be similar to that of a NAO robot, as done in [145], in particular,

$$m_L = 0.882\text{kg}, J_{L_0} = 0.5diag\{0.98, 0.91, 0.63\}\text{kg-m}^2,$$
$$m_P = 0.32\text{kg}, J_{P_0} = \begin{bmatrix} 0.2126 & 0.0004 & -0.0002 \\ 0.0004 & 0.2042 & 0.0010 \\ -0.0002 & 0.0010 & 0.2246 \end{bmatrix} \text{kg-m}^2.$$

Motion Primitive Parameters
$\rho_0 = 0.9, \quad \bar{\rho} = 0.1, \quad s = 0.125, \quad \bar{s} = 0.025.$
Controller Tuning Parameters
$\epsilon = 0.25, \quad k = \frac{16}{\epsilon^2}, \quad l = \frac{8}{\epsilon}, \quad A = \frac{1}{\epsilon^2}\text{diag}(1.2, 1.5, 1.8), \quad L = (1.5)(1.2)\text{diag}(1,1,1),$
$D1 = \frac{2}{\epsilon}\text{diag}(1,1,1), \quad D2 = \frac{0.5}{\epsilon}\text{diag}(1,1,1), \quad P = \frac{1.2}{\epsilon^2}\text{diag}(1,1,1), \quad Q = \frac{1.5}{\epsilon}\text{diag}(1,1,1).$

Table 5.3: List of various tuning parameters used in the Motion Primitive and Controller designs.

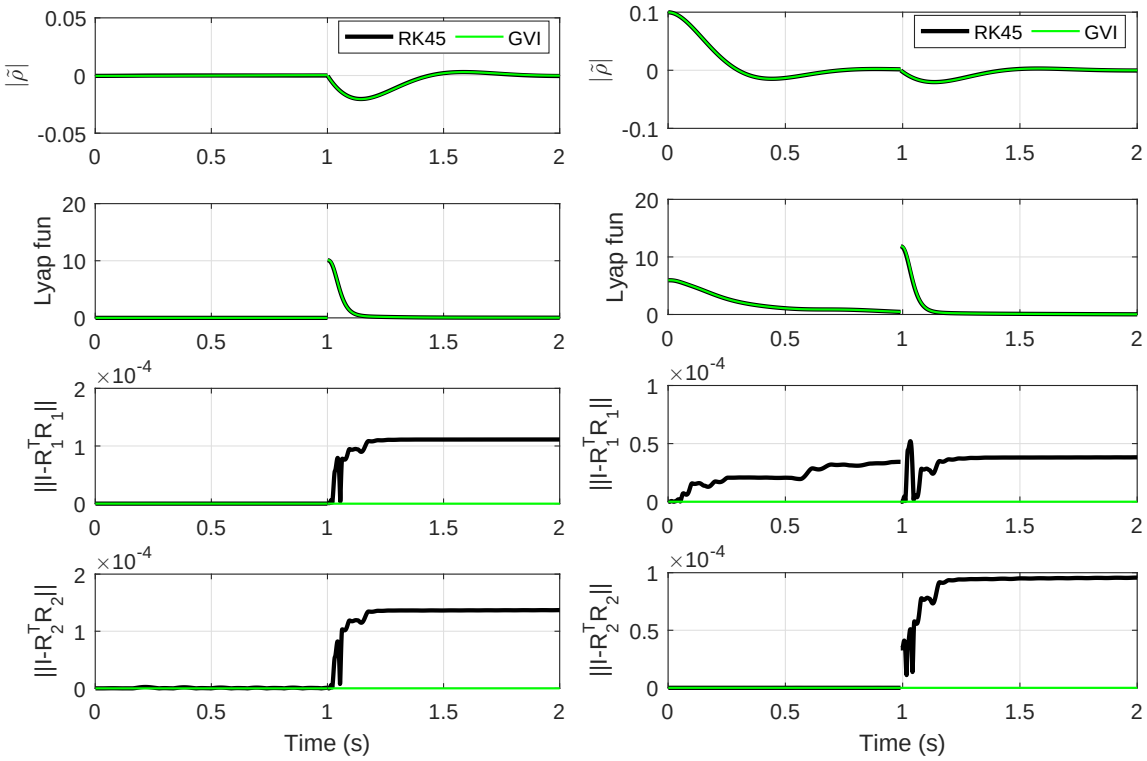
Walking in a straight line: We chose $\zeta_1 = \zeta_2 = e_2, R_{10} = R_{20} = I$, and $T = 1$s as in (5.41). Moreover, we introduce a constant phase offset in the angles for R_1^d, R_2^d to enable the swing legs to swing from -15° to 15°. For all other motion design and controller gain parameters, see Table 5.3. Figure 5.3a illustrates a snapshot and the tracking errors.

Walking towards a goal: We employ the walking in a straight line controller as above, however, we perform an event-based modification of R_{10}, R_{20} at each impact to change the heading of the biped. Figure 5.3b illustrates a snapshot and the tracking errors. Note that at each impact, the desired yaw instantaneously changes and the controller is able to regulate the errors asymptotically within a step.

Walking in a circle: We employ the walking towards a goal controller as above, however we modify R_{10}, R_{20} by a fixed amount at each impact. Moreover, R_{10}, R_{20} are also chosen to lean the body into the turn. Figure 5.3c illustrates a snapshot (along with the hip position demonstrating the body lean) and the tracking errors. Note that instead of modifying R_{10}, R_{20}, we could have modified ζ_1, ζ_2 too.

For all these motions, it is important to verify that the unilateral ground contact constraints and the friction constraints are satisfied during the walking, i.e., we need to ensure $|F_x| \leq \mu F_z$ and $|F_y| \leq \mu F_z$, where μ is the coefficient of static friction. The ground reaction forces were computed as, $\mathbf{F}_G = m\ddot{x}_{cm} - mge_3$, where $\mathbf{F}_G := [F_x \ F_y \ F_z]$ and x_{cm} is the center-of-mass of the RMB. It is equally critical to verify these constraints for the impact forces (I_R in Section 2.3) generated at the end of every step. Note that, the above-mentioned three motions namely a) 'walking in a straight line', b) 'walking towards a goal' and c) 'walking in a circle', take 10, 13, 19 steps respectively. In all these impact situations, we note that I_{R_z} is always positive ($\geq 1.6019\ N$). Moreover, the maximum values of $\frac{|I_{R_x}|}{I_{R_z}}$ and $\frac{|I_{R_y}|}{I_{R_z}}$ are 0.5872 and 0.5888, respectively, both occurring while walking in a circle.

Combining the impact-force data with the stance leg ground reaction force information, as shown in the third row of Fig. 5.3a, 5.3b, 5.3c, we can conclude that, for $\mu \geq 0.6$, the ground reaction force and the impact force respect the unilateral and friction cone constraints, thereby validating the assumptions that 1) stance leg is pinned to the ground during stance-phase and 2) No slip occurs at impact.



(a) Starting from desired configuration

(b) Starting with some configuration error

Figure 5.4: Comparing the performance of GVI with Runge-Kutta 45 based Integrator. In (a) RMB starts along the desired walking trajectory and a nominal controller is in action for tracking purposes. However, in (b), RMB starts with an initial error in its configuration.

In addition to testing the trajectory tracking controller on the continuous-time dynamics model of the RMB, it was also tested on the Discrete-time model developed in Section 5.1.1. Figure 5.4 shows the performance of the GVI in comparison to the traditional Runge-Kutta(4,5)-based integrator (RK45). RK45 is a very popular numerical integration algorithm based on the explicit Runge-Kutta formula [148]. It is also part of Matlab's ODE suite [149].

In Fig. 5.4(a), the two integrators are compared for a two-step walking scenario where the robot is initialized along the desired nominal trajectory(here, we chose the 'walking in a circle' trajectory) given by (5.41) and (5.42): $q_{s_0} = (\rho_0, R_{1_0}, R_{P_0}, s_0, R_{2_0})$. On the other hand, in 5.4(b), we start with a perturbed initial conditions: $q_{s_0}^{pert} = (\rho_0 + 0.1, R_x(\pi/10)R_{1_0}, R_{P_0}, s_0, R_{2_0})$ and $\omega_{s_0}^{pert} = (\dot{\rho}_0, \Omega_{1_0}, \Omega_{P_0} + 0.01, \dot{s}_0 - 0.03, \Omega_{2_0})$. Here $R_x(\theta)$ denotes a rotation along the x-direction by an angle θ. For both these cases, we plot (a) the desired leg extension tracking error ($\bar{\rho}$) as

obtained from (5.51) , (b) discrete Lyapunov function (obtained by discretizing (5.52), (5.53) and (5.54)) given by (5.79), and (c)-(d) R_1, R_2 norm errors. The norm errors are computed as $||I - R_i^T R_i|| \quad \forall i \in \{1, 2\}$. If the group structure ($R \in \mathbb{SO}(3)$) of R_1 and R_2 is preserved during the numerical integrations, the norm errors must be closer to zero.

$$V_k(\tilde{\rho}_{1_k}, Q_{1_k}, \dot{\tilde{\rho}}_{1_k}, \tilde{\Omega}_{1_k}, Q_{P_k}, \Omega_{P_k}, \tilde{s}_k, \dot{\tilde{s}}_k, Q_{2_k}, \tilde{\Omega}_{2_k}) = V_{L_{1_k}}(\rho_{1_k}, \tilde{\rho}_{1_k}, Q_{1_k}, \dot{\tilde{\rho}}_{1_k}, \tilde{\Omega}_{1_k}) + V_{P_k}(Q_{P_k}, \tilde{\Omega}_{P_k}, s_k, \tilde{s}_k, \dot{\tilde{s}}_k) + V_{L_{2_k}}(Q_{2_k}, \tilde{\Omega}_{2_k}) \tag{5.79}$$

where,

$$\begin{aligned} V_{L_{1_k}}(\rho_{1_k}, \tilde{\rho}_{1_k}, Q_{1_k}, \dot{\tilde{\rho}}_{1_k}, \tilde{\Omega}_{1_k}) &= \frac{1}{2} m \dot{\tilde{\rho}}_{1_k}^2 + \frac{1}{2} \tilde{\Omega}_{1_k}^T J_1(\rho_{1_k}) \tilde{\Omega}_{1_k} + \frac{1}{2} k \tilde{\rho}_{1_k}^2 + \Phi\big(\text{tr}(A - AQ_{1_k})\big), \\ V_{P_k}(Q_{P_k}, \tilde{\Omega}_{P_k}, s_k, \tilde{s}_k, \dot{\tilde{s}}_k) &= \frac{1}{2} \tilde{\Omega}_{P_k}^T J_P(s_k) \tilde{\Omega}_{P_k} + m_P \dot{\tilde{s}}_k^T \dot{\tilde{s}}_k + \frac{1}{2} \tilde{s}_k^T P \tilde{s}_k + \Phi\big(\text{tr}(A - AQ_{P_k})\big), \\ V_{L_{2_k}}(Q_{2_k}, \tilde{\Omega}_{2_k}) &= \frac{1}{2} \tilde{\Omega}_{2_k}^T J_{L_0} \tilde{\Omega}_{2_k} + \Phi\big(\text{tr}(A - AQ_{2_k})\big). \end{aligned} \tag{5.80}$$

Note that, for this study the step size chosen for the GVI and RK45 was $h = 10^{-3}$. From the third and fourth rows of Fig. 5.4, it is can be noted that the variational integrator maintained the group structure much better than the RK45 Integrator. On further examination, it was found that the GVI kept the norm error within 10^{-12} which is orders of magnitude better than RK45. Moreover, other parameters like configuration errors($\bar{\rho}$), energies, etc., as computed using the discrete GVI-based system model, track the continuous dynamics as accurately as the discrete-model based on RK45, if not better.

5.3 Summary

In this chapter, we introduced a new reduced-order legged robot model, called the RMB, to study bipedal motions that leverage variable torso inertia to exhibit non-trivial and human-like rotational maneuvers in $3D$. A Hybrid Geometric Model is developed by applying variational principles directly on the configuration manifold of the robot. The resulting dynamics are coordinate-free with no singularity issues. The variable torso inertia helps to capture the dynamics involving torso rotation, arm movements, etc. which are omitted in existing reduced-order models. We also outlined a geometric and variational discretization procedure for the RMB that guarantees preservation of critical structural properties during integration even for long simulation cycles. The preserved properties include physical ones like energy conservation (for conservative systems), momentum conservation (when there is symmetry in the Lagrangian), and geometric ones like the manifold structure of the robot's configuration space. This structure preserving property is also useful when building controllers based on energy-like Lyapunov functions, as shown in this work. On the control design front, we defined geometric motion plans for the RMB model to walk straight and in a circle (with a significant torso lean-in angle of 30^0). We also developed tracking controllers to achieve these desired motions using control lyapunov functions and showed that they are asymptotically stable. Additionally, the geometric control policy is also discretized variationally to highlight the preservation of the energy-like Lyapunov function structure that is critical to the controller's performance. Finally, we note that the controllers introduced for the

RMB can only be applied to robot models assuming full actuation, i.e. any flat-footed humanoid robot.

Chapter 6

Geometric Modeling and Control of Cassie Robot

Cassie is one of the more popular bipedal robot platforms in active research today. Recently, researchers have developed controllers for standing and $3D$ walking [150] for this robot. The latest in this line of work is multi-modal locomotion using Cassie [20]. Here, the objective is speed-up Cassie on flat terrains using wheeled platforms with an eye towards time-sensitive applications. This approach is also more energy-efficient. Walking is therefore limited to climbing steps or navigating unstructured terrain. Our current objective is to develop suitable model-based controllers to realize a diverse range of transverse plane motions using this wheeled-leg-platform.

6.1 Geometric Model of Cassie Robot

$m_b \in \mathbb{R}$	Pelvis Mass
$m_{hr} \in \mathbb{R}$	Mass at the right hip
$m_{hl} \in \mathbb{R}$	Mass at the left hip
$m_{kr} \in \mathbb{R}$	Mass at the right knee
$m_{kl} \in \mathbb{R}$	Mass at the left knee
$m \in \mathbb{R}$	Total robot mass $m = m_b + m_{hr} + m_{hl} + m_{kr} + m_{kl}$
$J_b \in \mathbb{R}^{3\times3}$	Pelvis Inertia
$J_r \in \mathbb{R}^{3\times3}$	Inertia of right leg
$J_l \in \mathbb{R}^{3\times3}$	Inertia of left leg
$e_1 \in \mathbb{R}^3$	Unit vector along X-axis in the Inertial frame
$e_2 \in \mathbb{R}^3$	Unit vector along Y-axis in the Inertial frame
$e_3 \in \mathbb{R}^3$	Unit vector along Z-axis in the Inertial frame

Table 6.1: List of notations used in this section and their definitions

As a first step, we develop a free-floating, coordinate-free bipedal model with *almost* the same kinematic structure as Cassie, as shown in Fig. 6.1. We strike a balance between geometric mod-

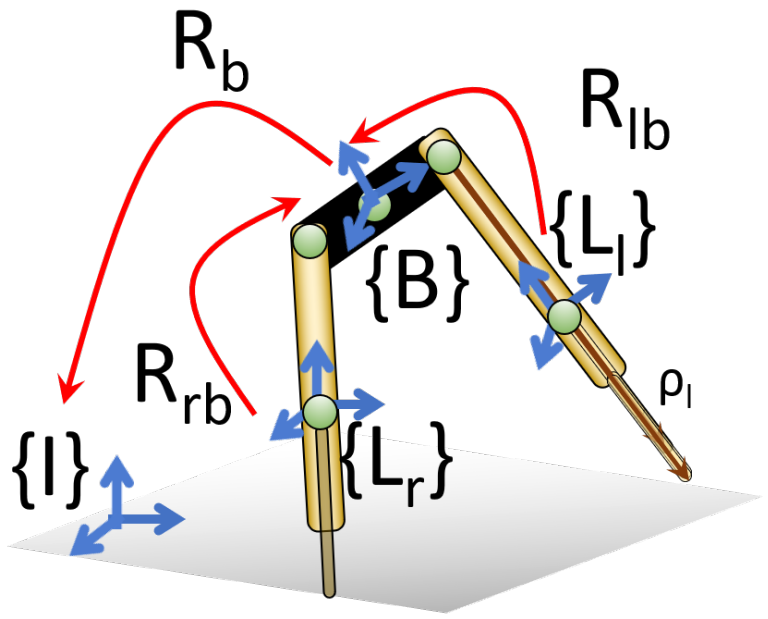


Figure 6.1: The geometric model schematic of Cassie robot.

eling complexity and real world applicability by modeling knees as prismatic joints, similar to the Linear Inverted Pendulum model [3]. Second, we do not model the toe joints and feet as their mass and inertia are negligible. Following the mass-less toe assumption, we fix the leg length to be twice that of the leg center of mass length (from the hip). The leg lengths as denoted as ρ_r and ρ_l for the right and left legs, respectively. Next, we denote the base center-of-mass position as x_b.

The leg and pelvis (base) orientations are represented using rotation matrices, R_r, R_l, R_b, respectively, where the subscripts r and l stand for right and left legs. Accordingly, The configuration manifold of the robot is $\mathcal{Q} = \mathrm{SE}(3) \times \mathbb{SO}(3) \times \mathbb{SO}(3) \times \mathbb{R}^+ \times \mathbb{R}^+$. The robot has 14 degrees of freedom in total of which only 8 are actuated. Mass and inertia parameters are defined with notations in Table. 6.1

6.1.1 Floating-base Dynamics

Having described the salient attributes of this robot, we now derive it's kinematics,

$$\dot{R}_b = R_b \widehat{\Omega}_b, \quad \dot{R}_{rb} = R_{rb} \widehat{\Omega}_{rb}, \quad \dot{R}_{lb} = R_{lb} \widehat{\Omega}_{lb}, \tag{6.1}$$

where, Ω_b, Ω_r, Ω_l are the respective body angular velocities. Ω_{lb} is the body angular rate of the left leg relative to the base. It is given by,

$$\Omega_{lb} = \Omega_l - R_{lb}^T \Omega_b, \text{where, } R_{rb} = R_l^T R_r. \tag{6.2}$$

Here, R_{lb}^T is called the *transport map* and is used to correctly compare two velocities that are evolving on different tangent spaces. This is characteristic to non-euclidean manifolds.

Using these kinematic relations, and using forward kinematics, we compute hip (p_{hr}/p_{hl}) and knee positions (p_{kr}/p_{kl}) and velocities for each leg. Having obtained the position kinematics of all the states with masses, we can compute the kinetic and potential energies accordingly and finally formulate the system Lagrangian. Using calculus of variations on the $\mathbb{SO}(3)$ manifold space and invoking the Lagrange-d'Alembert principle, we can obtain the equations of motion. The equations can be compacted by defining $q = [x_b\ R_b\ R_r\ R_l\ \rho_r\ \rho_l]$, $\omega = [\dot{x}_b\ \Omega_b\ \Omega_r\ \Omega_l\ \dot{\rho}_r\ \dot{\rho}_l]^T$, and $u = [\tau_{rb}\ \tau_{lb}\ f_r\ f_l]^T$, to get,

$$D(q)\dot{\omega} = C(q,\omega) + G(q) + B(q)u, \tag{6.3}$$

where,

$$\boldsymbol{B} = \begin{pmatrix} \mathbf{0}, & \mathbf{0}, & \mathbf{0}, & \mathbf{0} \\ -\mathrm{R}_{rb}^T, & -\mathrm{R}_{lb}^T, & \mathbf{0}, & \mathbf{0} \\ \mathrm{I}, & \mathbf{0}, & \mathbf{0}, & \mathbf{0} \\ \mathbf{0}, & \mathrm{I}, & \mathbf{0}, & \mathbf{0} \\ 0, & 0, & 1, & 0 \\ 0, & 0, & 0, & 1 \end{pmatrix}, \quad \boldsymbol{G} = \begin{pmatrix} mge_3 \\ \mathbf{0}_{3\times 3} \\ m_{kr} g \rho_{kr} \widehat{e_3} R_r^T e_3 \\ m_{kl} g \rho_{kl} \widehat{e_3} R_l^T e_3 \\ m_{kr} g e_3^T R_r e_3 \\ m_{kl} g e_3^T R_l e_3 \end{pmatrix},$$

$$\boldsymbol{C} = \begin{pmatrix} m_{kr}\rho_{rc} R_r \widehat{\Omega}_r \widehat{e}_3^T \Omega_r + m_{kl}\rho_{lc} R_l \widehat{\Omega}_l \widehat{e}_3^T \Omega_l + 2m_{kr}\dot{\rho}_{rc} R_r \widehat{e}_3^T \Omega_r + 2m_{kl}\dot{\rho}_{lc} R_l \widehat{e}_3^T \Omega_l, \\ \widehat{\Omega}_b \tilde{J}_b \Omega_b + 2m_{kr}\rho_b \dot{\rho}_{rc} \widehat{e}_2 R_{rb} \widehat{e}_3^T \Omega_r - 2m_{kl}\rho_b \dot{\rho}_{lc} \widehat{e}_2 R_{lb} \widehat{e}_3^T \Omega_l + \\ m_{kr}\rho_b \rho_{rc} \widehat{e}_2 R_{rb} \widehat{\Omega}_r \widehat{e}_3^T \Omega_r - m_{kl}\rho_b \rho_{lc} \widehat{e}_2 R_{lb} \widehat{\Omega}_l \widehat{e}_3^T \Omega_l, \\ \widehat{\Omega}_r J_r \Omega_r + m_{kr}\rho_{rc}^2 \widehat{e}_3 \widehat{\Omega}_r \widehat{e}_3^T \Omega_r + 2m_{kr}\rho_{rc}\dot{\rho}_{rc} \widehat{e}_3 \widehat{e}_3^T \Omega_r + m_{kr}\rho_{rc}\rho_b \widehat{e}_3 R_r^T R_b \widehat{\Omega}_b \widehat{e}_2^T \Omega_b, \\ \widehat{\Omega}_l J_l \Omega_l + m_{kl}\rho_{lc}^2 \widehat{e}_3 \widehat{\Omega}_l \widehat{e}_3^T \Omega_l + 2m_{kl}\rho_{lc}\dot{\rho}_{lc} \widehat{e}_3 \widehat{e}_3^T \Omega_l + m_{kl}\rho_{lc}\rho_b \widehat{e}_3 R_l^T R_b \widehat{\Omega}_b \widehat{e}_2^T \Omega_b, \\ m_{kr}\rho_{rc} \Omega_r^T \widehat{e}_3^2 \Omega_r + m_{kr}\rho_b e_3^T R_r^T R_b \widehat{\Omega}_b \widehat{e}_2^T \Omega_b \\ m_{kl}\rho_{lc} \Omega_l^T \widehat{e}_3^2 \Omega_l + m_{kl}\rho_b e_3^T R_l^T R_b \widehat{\Omega}_b \widehat{e}_2^T \Omega_b \end{pmatrix},$$

$$\boldsymbol{D} = \begin{pmatrix} m\mathrm{I}, & \mathbf{0}_{3\times 3}, & D_{13}, & D_{14}, & D_{15}, & D_{16} \\ \mathbf{0}_{3\times 3}, & \tilde{J}_b, & D_{23}, & D_{24}, & D_{25}, & D_{26} \\ D_{13}^T, & D_{23}^T, & \tilde{J}_r, & \mathbf{0}_{3\times 3}, & \mathbf{0}_{3\times 1}, & \mathbf{0}_{3\times 1} \\ D_{14}^T, & D_{24}^T, & \mathbf{0}_{3\times 3}, & \tilde{J}_l, & \mathbf{0}_{3\times 1}, & \mathbf{0}_{3\times 1} \\ D_{15}^T, & D_{25}^T, & \mathbf{0}_{1\times 3}, & \mathbf{0}_{1\times 3}, & m_{kr}, & 0 \\ D_{16}^T, & D_{26}^T, & \mathbf{0}_{1\times 3}, & \mathbf{0}_{1\times 3}, & 0, & m_{kl} \end{pmatrix},$$

$$\begin{aligned} D_{13} &= m_{kr}\rho_{rc} R_r \widehat{e}_3^T, & D_{23} &= m_{kr}\rho_b \rho_{rc} \widehat{e}_2 R_{rb} \widehat{e}_3^T, \\ D_{14} &= m_{kl}\rho_{lc} R_l \widehat{e}_3^T, & D_{24} &= m_{kl}\rho_b \rho_{lc} \widehat{e}_2^T R_{lb} \widehat{e}_3^T, \\ D_{15} &= m_{kr} R_r e_3, & D_{25} &= m_{kr}\rho_b \widehat{e}_2 R_{rb} e_3, \\ D_{16} &= m_{kl} R_l e_3, & D_{26} &= m_{kl}\rho_b \widehat{e}_2^T R_{lb} e_3. \end{aligned}$$

Here, $\tilde{J}_b = J_b + m\rho_b^2 \widehat{e}_2 \widehat{e}_2^T$, $\tilde{J}_r = J_r + m_{kr}\rho_{kr}^2 \widehat{e}_3 \widehat{e}_3^T$, $\tilde{J}_l = J_l + m_{kl}\rho_{kl}^2 \widehat{e}_3 \widehat{e}_3^T$ are the *effective inertias* experienced by the pelvis, right and left legs, respectively.

6.1.2 Holonomic Constraints

After obtaining the dynamical model, we define suitable holonomic constraints to restrict the dynamics to satisfy the unilateral ground contact constraints. These constraints need to be applied on the floating-base dynamics for model both walking and riding (on Hovershoes) motions.

Foot Positions and Velocities

To obtain the constraint jacobian, we use forward kinematics to first obtain foot positions and velocities. They are,

$$\begin{aligned} p_{fr} &= x_b - \rho_b R_b e_2 + \rho_r R_r e_3, \\ p_{fl} &= x_b + \rho_b R_b e_2 + \rho_l R_l e_3, \\ \dot{p}_{fr} &= \dot{x}_b - \rho_b R_b \widehat{e}_2^T \Omega_b + \dot{\rho}_r R_r e_3 + \rho_r R_r \widehat{e}_3^T \Omega_r, \\ \dot{p}_{fl} &= \dot{x}_b + \rho_b R_b \widehat{e}_2^T \Omega_b + \dot{\rho}_l R_l e_3 + \rho_l R_l \widehat{e}_3^T \Omega_l. \end{aligned} \tag{6.5}$$

Next, we define $\dot{p}_f = [\dot{p}_{fr}\ \dot{p}_{fl}]^T$ and express the forward kinematics in the matrix form as,

$$\dot{p}_f = \underbrace{\begin{bmatrix} \mathrm{I} & -\rho_b R_b \widehat{e}_2^T & \rho_r R_r \widehat{e}_3^T & \mathbf{0} & 2R_r e_3 & 0 \\ \mathrm{I} & \rho_b R_b \widehat{e}_2^T & \mathbf{0} & \rho_l R_l \widehat{e}_3^T & 0 & 2R_l e_3 \end{bmatrix}}_{=: J_c} \omega, \tag{6.6}$$

$$\therefore \dot{p}_f = J_c\, \omega. \tag{6.7}$$

Ground Contact Constraints

For the robot to always remain on the ground, we need that the z-component of all the positions in p_f to be zero. This can be imposed as,

$$h_c(q) := e_3^T p_f \equiv 0, \tag{6.8}$$

$$\dot{h}_c := e_3^T J_c\, \omega = 0, \tag{6.9}$$

$$\ddot{h}_c := e_3^T (\dot{J}_c\, \omega + J_c\, \dot{\omega}) = 0, \tag{6.10}$$

$$\ddot{h}_c := \dot{J}_{cz}\, \omega + \dot{J}_{cz}\, \dot{\omega} = 0, \quad \text{s.t. } J_{cz} = e_3^T J_c. \tag{6.11}$$

Upon imposing the ground contact constraints, the constrained dynamics have the form,

$$D(q)\dot{\omega} = C(q, \omega) + G(q) + B(q)u + J_{cz}^T \lambda. \tag{6.12}$$

By substituting (6.11) in (6.12), we can calculate λ in terms of $q,\ \omega,\ u$,

$$\lambda = -(J_{cz} D^{-1} J_{cz}^T)^{-1} (J_{cz} D^{-1} (C(q, \omega) + G(q) + B(q)u) + \dot{J}_{cz}\, \dot{\omega}). \tag{6.13}$$

Note that, we also need to ensure that $\lambda \geq 0$ as the constraint is *unilateral*. To guarantee that our inputs preserve these holonomic constraints, we only use the constrained dynamics in control design. The constrained dynamics are,

$$D_c(q)\dot{\omega} = C_c(q,\omega) + G_c(q) + B_c(q)u, \tag{6.14}$$

where, $D_c = J_{cz}D$, $C_c = J_{cz}C$, $G_c = J_{cz}G$, $B_c = J_{cz}B$, respectively. In other words, we operate strictly in the *null space* of J_{cz}^T.

Having defined the coordinate-free constrained dynamical model of Cassie, in the next section, we define suitable propulsion forces at the foot to complete our Cassie-Hovershoes modeling.

6.1.3 Cassie with Hovershoes

Here, we first briefly summarize the Hovershoe dynamics which will be used later to model the combined system.

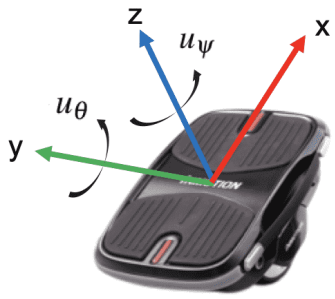


Figure 6.2: Hovershoe model with pitch(u_θ) and yaw(u_ψ) actuation.

Dynamical Model of Hovershoes

Hovershoe is a sensitive wheeled platform. It has an internal controller that regulates the pitch to zero degrees when external force is zero. The internal parameters and states of the Hovershoe are unknown. Moreover, system identification was not possible to estimate model parameters. Keeping this in mind, we mainly focused on identifying the structure of its dynamics to use in the control design. To develop a closed-loop dynamical model, we consider θ, ψ, u_θ, u_ψ to be the pitch and yaw angles and the corresponding torques in the body frame, and x, y to be the x, y positions in the global frame, and lastly, v to be the scalar speed along the e_1 direction in the body frame. Accordingly the dynamics are,

$$J_\theta\ddot{\theta} = -c_1\theta - c_2\dot{\theta} + u_\theta, \tag{6.15}$$
$$J_\psi\ddot{\psi} = -c_3\dot{\psi} + u_\psi, \tag{6.16}$$
$$\dot{x} = v\cos(\psi), \tag{6.17}$$
$$\dot{y} = v\sin(\psi), \tag{6.18}$$
$$m\dot{v} = c_4\theta. \tag{6.19}$$

Here, the parameters (c_1, c_2, c_3) correspond to the Hovershoe internal controller, and c_4 corresponds to the contact model between the Hovershoe and the ground. The same dynamics are used for both the left and right Hovershoes.

Nonholonomic Constraints

In addition to the unilateral ground contact constraints, the Hovershoes attachment adds two more constraints to the full system dynamics. At the feet, no instantaneous velocities are allowed in the lateral direction. Unlike ground contacts, these constraints are *nonholonomic* and *bilateral*. Using the foot kinematics defined in (6.5), we express these velocity constraints as,

$$h_{nh}(q,\omega) := e_2^T \begin{bmatrix} R_r^T \\ R_l^T \end{bmatrix} \dot{p}_f \equiv 0. \tag{6.20}$$

Similar to (6.11), in order to impose these constraints, we also set $\dot{h}_{nh}$ to zero and derive the corresponding constraint forces λ_{nh} and the jacobian J_{nh} that must be added to the dynamics in (6.14). Therefore we have,

$$\dot{h}_c := \underbrace{\left(\frac{\partial h_{nh}}{\partial q}\right)}_{=:\partial_q J_{nh}} \dot{q} + \underbrace{\left(\frac{\partial h_{nh}}{\partial \omega}\right)}_{=:J_{nh}} \dot{\omega}. \tag{6.21}$$

Here, $J_{nh} = \frac{\partial h_{nh}}{\partial \omega}$ or the term that corresponds to $\dot{\omega}$. Observe that, the first partial derivative, $\partial_q J_{nh}$, in (6.21) is *non trivial*. Unlike in (6.11), this extra term is not a function of ω but of $\dot{q}$. Here, $\dot{q}$ depends on both q and ω i.e., $\dot{q} = [\dot{x}_b \; R_b\widehat{\Omega}_b \; R_{rb}\widehat{\Omega}_{rb} \; R_{lb}\widehat{\Omega}_{lb} \; \dot{\rho}_r \; \dot{\rho}_l]^T$. It has the same state size as q. Further upon deriving the Jacobians, we have,

$$\begin{aligned}
J_{nh} &= e_2^T \begin{bmatrix} \mathrm{R}_\mathrm{r}^\mathrm{T} & -\rho_b R_{rb}^T \widehat{e}_2^T & \rho_r \widehat{e}_3^T & \mathbf{0} & 2e_3 & 0 \\ \mathrm{R}_\mathrm{l}^\mathrm{T} & \rho_b R_{lb}^T \widehat{e}_2^T & \mathbf{0} & \rho_r \widehat{e}_3^T & 0 & 2e_3 \end{bmatrix}, \\
\partial_q J_{nh} &= \begin{bmatrix} \mathbf{0} & -\rho_b(\widehat{\Omega}_b e_2)^T \otimes R_r^T & -\rho_b (e_2)^T \otimes (\widehat{\Omega}_b^T R_r^T) & \mathbf{0}_{3\times 9} & 2\widehat{\Omega}_r e_3 & \mathbf{0} \\ \mathbf{0} & \rho_b(\widehat{\Omega}_b e_2)^T \otimes R_l^T & \mathbf{0}_{3\times 9} & \rho_b (e_2)^T \otimes (\widehat{\Omega}_b^T R_l^T) & \mathbf{0} & 2\widehat{\Omega}_l e_3 \end{bmatrix}.
\end{aligned} \tag{6.22}$$

Here, $\otimes$ operator is a kronecker product between a vector in $\mathbb{R}^3$ and a matrix $\mathbb{R}^{3\times 3}$. We can calculate λ_{nh} as,

$$\lambda_{nh} = -(J_{nh} D^{-1} J_{nh}^T)^{-1} (J_{nh} D^{-1} (C(q,\omega) + G(q) + B(q)u) + \partial_q J_{nh} \dot{q}). \tag{6.23}$$

Finally with a slight abuse of notation, we redefine the constrained dynamics as in (6.14). However, the new constraint jacobian used is $J_c = [J_{cz}; J_{nh}]$. After applying both the holonomic and nonholonomic constraints the Cassie-Hovershoes system's *effective* degrees-of-freedom reduces from 14 to 10. As a next step, we add propulsive forces at Cassie's feet to abstract Hovershoes actuation in the body-frame e_1 direction.

Propulsive Forces at the Feet

We add Hovershoes propulsion forces to Cassie based on Assumption 6.1.1.

***Assumption* 6.1.1.** *Assume a rigid contact between Cassie's feet and the Hovershoes such that Cassie's toe pitch and the hip rotation inputs can be equated to u_θ, u_ψ of the Hovershoe. This assumption is valid when $\lambda > 0$ and there is no slip at the Cassie-Hovershoe contact interface.*

Note that, our Cassie geometric model in (6.3) doesn't consider either feet or toe joint actuation. We add this to the model in the form of a *force* at the foot. This is inspired from equation (6.19) in the Hovershoe dynamics. This force, f_v, is directly proportional to toe pitch angle from equation (6.19) assuming a rigid contact between Cassie's foot and the Hovershoe. Accordingly, we define two forces f_{vr} and f_{vl} for the right and left feet. Note that, these forces are always acting along the x-direction (along e_1) in the Hovershoe local frame. By including these additional forces, we define an augmented input vector $\bar{u} = [\tau_{rb}\ \tau_{lb}\ f_r\ f_l\ f_{vr}\ f_{vl}]^T$ and the corresponding new input matrix $\bar{B} = [B\ \bar{J}_c^T]$, where $\bar{J}_c = e_1^T J_c$. Finally, the constrained Cassie-Hovershoe dynamics are,

$$D_c(q)\dot{\omega} = C_c(q,\omega) + G_c(q) + \bar{B}_c(q)\bar{u}, \quad s.t.\ \bar{B}_c = J_{cz}\bar{B} \tag{6.24}$$

In the next section, we use this dynamical model to develop a geometric controller that can be used to both stabilize Cassie on Hovershoes or track desired transverse plane trajectories.

6.2 Motion Planning and Control

The Cassie-Hovershoes system can be treated as a car-like system capable of all transverse-plane motions. Additionally, it can also change body height (crouch to pass through a low ceiling) and step-width (to ride over low-height ground obstacles). Simple PD controllers that exhibit these behaviors have been presented in [20]. Another key-feature desired from Cassie-Hovershoes system is the ability to *turn in place* to retrace its path when it stumbles upon a dead-end. This is useful while performing exploratory mapping tasks and the current controllers fail to turn in place. Moreover, current methods [20] control joints independently and mainly rely on heavy gain-tuning and accurate initialization to achieve expected performance. In contrast, a systematic control design for whole-body motion of the Cassie-Hovershoes system with convergence guarantees and ability to recover from large initialization errors is attempted here.

6.2.1 Motion Planning

For turning in place, we define a suitable turning velocity for the pelvis along the e_3 axis as Ω_{bz}^d, keeping the angular velocities about e_1 and e_2 zero. Therefore, we have $\Omega_b^d = [0\ 0\ \Omega_{bz}^d]^T$, where Ω_{bz}^d can be proportional to number of turns desired in a given time duration. Further, we desire that the relative angular velocities of both legs w.r.t the pelvis remain zero i.e., $\Omega_{rb}^d = \Omega_{lb}^d = \mathbf{0}$. We keep the leg lengths, pelvis and leg configurations constant during the motion. Therefore, the desired trajectories (or motion primitives) for the Cassie-Hovershoes systems turning in place in a time interval $[0,\ T]$ are:

$$\Omega_b = [0\ 0\ \frac{2\pi k}{T}]^T, \Omega_{rb} = \mathbf{0}, \Omega_{lb} = \mathbf{0}, \tag{6.25}$$

Additionally, we keep $R_b^d, R_{rb}^d, R_{lb}^d, x_b^d, \dot{x}_b^d$ constant. Having defined suitable motion plans to generate in place turning motions, we can design a suitable controller that can track them next.

6.2.2 Control Design

As a fist step, we transform (6.24) into its control-affine form as,

$$\dot{\omega} = \underbrace{D_c^{-1}(C_c + G_c)}_{=:f_c} + \underbrace{D_c^{-1}(\bar{B}_c)}_{=:g_c}\bar{u} \tag{6.26}$$

where, f_c and g_c are defined above to represent the dynamics in the control affine form. According to the motion plans specified in (6.25) a total for 10 outputs are defined, one for pelvis height and three each for the pelvis, right leg and left leg angular velocities, respectively, as summarized below:

$$y := h(q,\ q^d) = H(q) - H^d(q^d) \to \mathbf{0}. \tag{6.27}$$

where, $h = [h_1,\ h_2,\ h_3,\ h_4]$, and each h_i is defined as follows:

- Desired Vertical Height: $h_1 = e_3^T(x_b - x_b^d)$,
- Desired Pelvis Angular Velocity: $h_2 = \Omega_b - (R_b^T R_b^d)\Omega_b^d$,
- Desired Right Leg Angular Velocity w.r.t Pelvis: $h_3 = \Omega_{rb} - (R_{rb}^T R_{rb}^d)\Omega_{rb}^d$,
- Desired Left Leg Angular Velocity w.r.t Pelvis: $h_4 = \Omega_{lb} - (R_{lb}^T R_{lb}^d)\Omega_{lb}^d$.

Note that, to define the outputs h_2, h_3, and h_4, we used the geometric error functions introduced earlier in Section 4.3 (see (4.18)). We can use the same here to complete the control design defining a geometric PD scheme for tracking outputs y.

$$y = v,\ s.t.\ v = \begin{bmatrix} k_{p_1}(x_b - x_b^d) + k_{d_1}(\dot{x}_b - \dot{x}_b^d) \\ k_{p_2} e_R(R_b, R_b^d) + k_{d_2} e_\Omega(\Omega_b, \Omega_b^d) \\ k_{p_3} e_R(R_{rb}, R_{rb}^d) + k_{d_3} e_\Omega(\Omega_{rb}, \Omega_{rb}^d) \\ k_{p_4} e_R(R_{lb}, R_{lb}^d) + k_{d_4} e_\Omega(\Omega_{lb}, \Omega_{lb}^d) \end{bmatrix}, \tag{6.28}$$

where, $e_R = \frac{1}{2}[R_d^T R - R^T R_d]^\vee$ and $e_\Omega = \Omega - (R^T R_d)\Omega_d$ are geometric configuration and angular rate errors introduced in [136, 133, 137]. Note that, by choosing a PD controller over a feedback linearization based controller, we trade-off efficiency for greater robustness [13, 58]. This is particularly useful as we have sufficient model uncertainty in the form unknown Hovershoes internal model parameters and simplified leg dynamics of Cassie (i.e., using piston-like legs instead of knees).

6.2.3 Simulation Results

We numerically test the model and controller performance by using Cassie robot model params obtained from [151]. We desire a rest-to-rest in place turning manuever which can be made time-varying. Accordingly, set $\Omega_b^d(t) = \frac{2\pi k}{T}t$ where k is the number of turns we desire. Similarly, we can also vary body height over time to make the motion more dynamic, $x_b^d = 1 - 0.4\sin(\frac{2\pi t}{T})$. The

total simulation time T is set to 2.5 seconds. Simulation snapshots are shown in Fig. 6.3. Base states and their time evolution is plotted in Fig. 6.4. Note the exponential convergence to Ω_b^d. The time evolution of leg states was as expected and visualized in Fig. 6.3 and the plots are omitted for brevity.

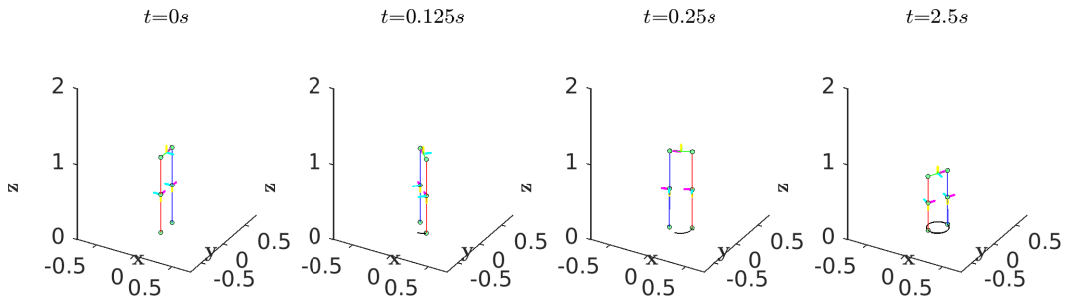


Figure 6.3: Snapshots of the robot turning in place.

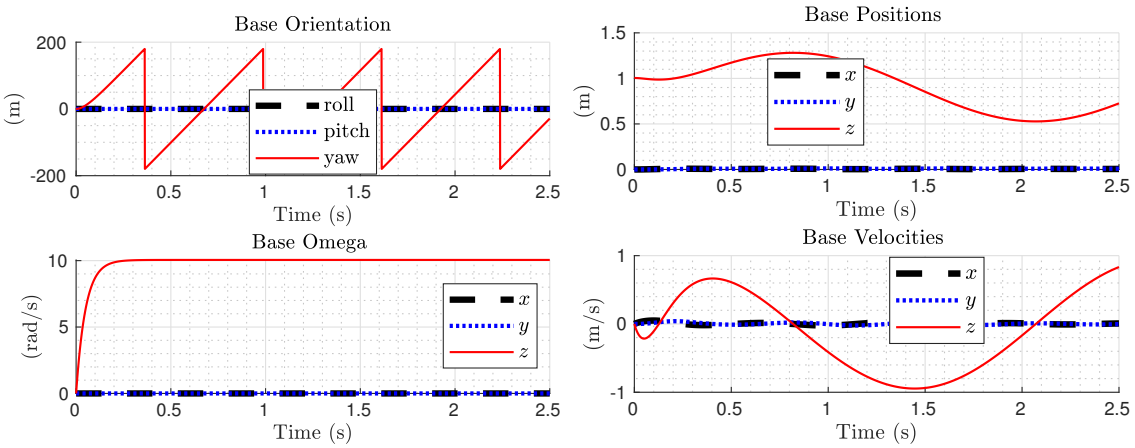


Figure 6.4: Base trajectories evolution in time: base orientation R_b(expressed using Euler angles) and Ω_b are shown on the *left* while position x_b and velocity $\dot{x}_b$ trajectories are shown on the *right*.

Having validated the controller performance in simulation, we will discuss experimental testing and results in the next section.

6.3 Robot Experiments

This section briefly summarizes preliminary experimental results using geometric controllers on the Cassie Robot and the Cassie-Hovershoes System.

6.3.1 State Estimation

We use the onboard IMU to obtain the pelvis (base) angular rates expressed in the body-frame i.e., Ω_b. We don't use any sensing for base configuration. Stock controllers are used to initialize

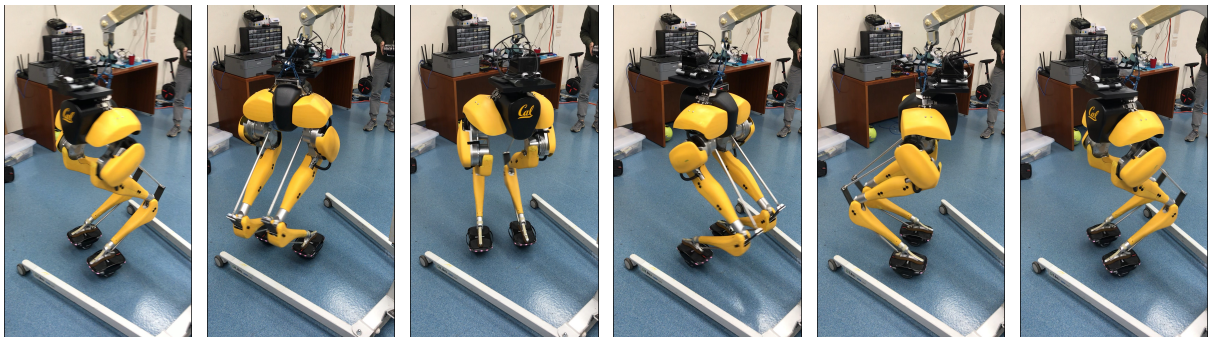
the robot at a desired standing pose and therefore we assume zero base configuration error at initialization. Further, joint encoders are used to measure all the leg joint angles q defined in Section 3.2 equation (3.13). To convert the desired leg heights ρ_r and ρ_l into corresponding knee pitch (q_{4R}, q_{5R}) and shin (q_{4L}, q_{5L}) pitch angles, we use the *inverse kinematics* mappings provided in the Cassie software release [151].

To convert the leg orientations and rates (w.r.t. base) from their Euler parametrizations to coordinate-free representations we use the transfer maps $\mathcal{R}$ and $\mathcal{T}_{\dot{q}}$ defined in Section 4.2.2, equations (4.5) and (4.6).

6.3.2 Cassie Standing in place

As a first experiment, we test the geometric control laws on Cassie for the task of standing in place. For this task, we only use the feedback laws defined in (6.28). Here, the desired leg lengths and toe pitch angles are set to their nominal values, the configurations as $R_b^d = R_{rb}^d = R_{lb}^d = \mathbf{I}$, and the angular velocities as $\Omega_b^d = \Omega_{rb}^d = \Omega_{lb}^d = \mathbf{0}$. A snapshot of the robot standing using this control implementation is shown in Fig. 6.5. Controller robustness was tested by providing minor pushes to the robot in the lateral and forward directions. Note that, the same gains used in the earlier Euler-parametrized controllers [20] were used here to compare sensitivity to perturbations. For our pushing experiments, we noted that the geometric PD controller had a greater error tolerance range than its Euler-parametrized counterpart. This simple experiment served as an experimental sanity check for geometric controller use.

Figure 6.5: Cassie standing.



(a) t = 0s (b) t = 2s (c) t = 4s (d) t = 6s (e) t = 8s (f) t = 10s

Figure 6.6: Snapshots of Cassie making two turns in place.

6.3.3 Cassie-Hovershoes turning in place

For the Cassie riding Hovershoes experiment, we use the same control structure mentioned above. We modify the desired Ω_{bz}^d to 1.5 radians per second. Further, the

desired toe pitch angles are proportional to the desired tangential velocities of the Hovershoes. The velocity magnitudes are same but applied in opposite directions. For example, for turning clockwise, left toe pitch is positive while the right toe pitch negative and vice versa. Finally, $|v^d| = p^d_{fy}\Omega^d_{bz}$, where p^d_{fy} is the desired constant y-distance from the base to each foot in the base-frame. Snapshots for the robot turning in place are shown in Fig. 6.6. Note that, for successful execution of turns, the robot was very carefully made to stand on the Hovershoes such that its line feet align perfectly with the center axes of the Hovershoes. The system is particularly sensitive to these initialization errors as there is no means for the robot to correct them on-the-fly. Note that we assume the legs to be pinned to the Hovershoes. There was also some body oscillation around the standing pose while turning. The controller gains need to be tuned further to dampen this behavior.

6.4 Summary

A new Geometric Model is developed for the Cassie-Hovershoes multimodal robot to capture motion dynamics that involve dramatic orientation changes like turning in place. Note that, this is a challenging manuever to execute using a high center-of-gravity bipedal robot standing on Hovershoes. Essentially, the system is a like a pair of $3D$ cart-poles connected by the pelvis. Motion plans are devised in the transverse plane for the turning in place and geometric controllers are designed to stabilize the erect robot pose while executing dynamic transverse plane motions. These controllers are numerically and experimentally realized.

Chapter 7

Part II Conclusions and Future Work

In Part II, Geometric Control for Dynamic Legged Robots, we first highlighted the advantages of using *geometric* or coordinate-free formulations in modeling and control of robots. Specifically, through exhaustive empirical testing of geometric and Euler-parametrized controllers for set-point regulation of a $3D$ pendulum, we demonstrated that geometric controllers are generally more efficient. This was presented in Chapter 4. Building on this result, we introduced two new legged robot models, (a) the Reaction Mass Biped (RMB) model and (b) the Cassie geometric model.

In Chapter 5, the RMB model was introduced to study bipedal motions that leverage variable torso inertia to exhibit non-trivial and human-like rotational maneuvers in $3D$. A Hybrid Geometric Model was developed by applying variational principles directly on the configuration manifold of the robot. The resulting dynamics are coordinate-free with no singularity issues. The variable torso inertia helps to capture the dynamics involving torso rotation, arm movements, etc. which are omitted in existing reduced-order models. We also outlined a geometric and variational discretization procedure for the RMB that guarantees preservation of critical structural properties during integration even for long simulation cycles. The preserved properties include physical ones like energy conservation (for conservative systems), momentum conservation (when there is symmetry in the Lagrangian), and geometric ones like the manifold structure of the robot's configuration space. This structure preserving property is also useful when building controllers based on energy-like Lyapunov functions, as shown in this work. On the control design front, we defined geometric motion plans for the RMB model to walk straight and in a circle (with a significant torso lean-in angle of 30^0). We also developed tracking controllers to achieve these desired motions using control lyapunov functions and showed that they are asymptotically stable. Additionally, the geometric control policy is also discretized variationally to highlight the preservation of the energy-like Lyapunov function structure that is critical to the controller's performance. Finally, we note that the controllers introduced for the RMB can only be applied to robot models assuming full actuation, i.e. any flat-footed humanoid robot.

Later, in Chapter 6, a geometric model was developed for the Cassie robot riding a pair of Hovershoes. We used this model to plan and control challenging transverse-plane motions like turning in place standing on Hovershoes. Note that the robot has a high center-of-gravity and the it's full $3D$ pose has to be stabilized while turning. The system is akin to a pair of $3D$ cart-poles connected through the pelvis. The in place tuning motion plan and the geometric controllers

designed to track it are both numerically and experimentally validated.

7.1 Future Directions

Several interesting directions emerge for extending this work. They are briefly summarized below:

7.1.1 Fast Navigation through Cluttered Environments

The RMB model can be used to generate walking motion plans involving body leans, aggressive turning, etc. for humanoids like HRP [46], ATLAS [33] and ASIMO [44]. Similarly, the Cassie robot model paired with Hovershoes can be used to generate faster and aggressive transverse plane motions to quickly navigate through narrow corridors and other cluttered indoor environments.

7.1.2 Extend to other Legged Motions

In addition to bipedal robots, geometric modeling and control design can be applied to quadrupedal robots as well to realize acrobatic motions like the back flip, somersault, etc. In [152], the authors an MPC controller using a geometric lumped rigid body model of a quadruped using the ground reaction forces as input. This can be expanded by modeling leg orientations as well (similar to the bipedal robot models introduced in the previous chapters). This allows mid-air reorientation planning and control unlike [152].

7.1.3 Trajectory Optimization

In Part II, the main emphasis was on modeling and control design. Motion plans were hand-designed to highlight the advantages of our novel modeling choice and the corresponding control design. However, to actually utilize these models and controllers on robots navigating in the real world while avoiding obstacles, we require a more systematic approach. A promising line of future work is trajectory optimization on manifolds for motion planning.

Trajectory optimization on Manifolds is currently an active area of research. It was applied for optimal collision-free motion planing of a free-floating rigid body in [153] and subsequently extended to UAVs in [154]. Parallely, it has received much attention in the manipulation community as well [155, 156].

However, extending these ideas to legged robot motion planning is non-trivial. Bipedal robots are hybrid models, not fixed-based and inherently unstable. These additional constraints could severely impact optimization convergence.

7.1.4 Geometric Control to Underactuated Legged Robots

The RMB model is fully actuated and the Cassie model, though underactuated in general, is fully actuated for the specific case of riding on Hovershoes (after imposing the necessary holonomic and nonholonomic constraints). However, in general, the continuous phase of dynamic walking can be fully or partially underactuated (when there is a point or a line contact with the ground). This makes control design and trajectory optimization even more challenging. HZD controllers [51] leverage this underactuation to design and track *virtual constraints* that render the resulting zero dynamics stable leading to limit cycle walking behaviors.

These HZD controllers can be enhanced using geometric control formulations. For example, consider a $3D$ legged robot with point feet and a torso. During stance phase, we can only control the relative angles between the torso and either leg but not the absolute torso orientation. Instead of treating the unactuated torso pitch and roll as independently evolving states, we can model the torso orientation as a single state and stabilize it. We have already demonstrated that this is more efficient using the $3D$ pendulum model in Chapter 4. The main challenge in extending geometric control to HZD is the trajectory optimization part which is used to discover the lower dimensional embedding where the zero dynamics is stable.

Part III

Learning for Dynamic Legged Robots

Chapter 8

Learning a Unified Walking Policy from Gait Libraries

8.1 Introduction

Control design to realize dynamic and versatile motions still remains a challenge for bipedal robots. Specifically, explicit modeling of ground contact is not possible. Even if the robot can be stabilized within-stride (while stepping forward), the impact at the end of the step could potentially destabilize the robot. Further, any within-stride perturbation or a drastic change in the desired velocity aggravates this problem. However, we have made considerable progress in developing stable *limit-cycle walkers* ([7]). i.e., robots that can maintain the commanded velocity or step length, in addition to stabilizing while sustaining impacts. The objective of this work is to leverage this prior work and enhance the ability of limit-cycle walkers to smoothly transit between velocity-specific limit cycles, and thus, realize a continuum of velocities. This is achieved via learning a smooth policy by regressing over the finite set of pre-optimized gaits. The resulting learned policy can be used by a high-level planner to achieve full range navigation. Before going into the details of our implementation, we first define the terms *gait* and *gait parameter* which will be used frequently in the rest of this paper.

***Definition* 8.1.1.** *A gait parameter is a higher-level motion descriptor used for bipedal robot path planning. Forward and lateral velocities, step length, step height, etc. are commonly used gait parameters for legged robots. In this work, the chosen gait parameters are forward velocity, lateral velocity and step height.*

***Definition* 8.1.2.** *A gait ([157]) is any locomotion behavior that is characterized by periodic motion patterns. These motion patterns encode desired robot configurations that evolve in time. Typically, gaits are parametrized using smooth curves, and particularly in this work, we use a bézier curve parametrization for our gait.*

In this work, we restrict our attention to *hybrid zero dynamics (HZD)* based gait design and control which was introduced for $3D$ bipedal robots in [158]. This technique uniquely offers a

general and mathematically rigorous procedure to generate asymptotically stable, fast and energy-efficient gaits using the full robot model. It has been widely applied to a range of bipedal robots like RABBIT ([32]), MABEL ([7]), DURUS ([159]), ATRIAS ([160]) and Cassie ([161]). For a thorough review of this topic, see [113]. Note that, these periodic gaits are designed to achieve either a fixed forward velocity or step-length while walking and running on a flat-ground. However, to achieve robust walking over an uneven terrain, we desire a robust policy that can modulate velocity by transiting between gaits. Recently, *gait library* method was proposed in [160] and [162]. Similarly, to navigate on discrete terrain this technique was used to modulate step-length in [163], both step-length and step-width in [164], and both step-length and step-height in [13]. In the gait library method, a set of gaits are designed offline using a trajectory optimization toolbox like FROST ([165]) and C-FROST ([112]) and the interpolation is used online to generate a continuum of gaits from this set.

It was first proposed in [166] to learn a continuous control policy from gait libraries. Starting from learning individual policies to command desired forward velocities, step-heights and lateral velocities, they eventually learned a unified policy and successfully demonstrated robust walking on uneven terrain using the ATRIAS bipedal robot in [166]. Additionally, they performed rigorous analysis to offer stability and robustness guarantees on the learned policy in [167].

8.2 Learning-based Gait Policy

Cassie robot model and gait library design are already summarized in Section 3.2. In this section, we build on that to present our gait library learning process. The objective is to learn a control policy, $\mathcal{GN} : \mathbb{R}^3 \rightarrow \mathbb{R}^{120}$, to generate a dynamically feasible gait for the full robot model that is parametrized using 120 bézier parameters. We call them *gait features* in the rest of this section. The inputs are 3 gait parameters namely forward velocity, V_x, lateral velocity, V_y, and step height, SH. These parameters encode a two-step gait, for each of the ten actuated robot joints using a degree six bézier curve each. In addition to generating the gait prediction, Gait-Net is also jointly trained to accurately reconstruct the gait parameters using the learned gait prediction. This serves two purposes: 1) the reconstruction process serves as a *regularization* to the gait prediction process as both errors are back-propagated during training, and, 2) the reconstruction loss at test-time serves as a metric to evaluate gait prediction quality.

For clarity, we denote the gait parameters and gaits used for training as $\mathbf{p_i}$ and $\mathbf{b_i}$, respectively. Accordingly, we denote the gait prediction and the reconstructed gait parameters as $\tilde{\mathbf{p}}_\mathbf{i}$ and $\tilde{\mathbf{b}}_\mathbf{i}$, respectively. In both cases, the subscript i denotes the sample index. Finally, the training dataset is denoted as $\mathbf{P} = [\mathbf{p_1}\ \mathbf{p_i} \dots \mathbf{p_n}]$ for input data and $\mathbf{B} = [\mathbf{b_1}\ \mathbf{b_i} \dots \mathbf{b_n}]$ for ground truth gait data. Here, n is used to denote dataset size.

8.2.1 Gait-Net

The proposed autoencoder-based network architecture of the Gait-Net is shown in Fig. 8.1. The *fifth layer* output of the Gait-Net is used as gait prediction and the *last layer* reconstructs the input. A total of four layers are used to incrementally *decode* the resulting gait from the low-dimensional gait parameters and symmetric four layer structure is used to *compress* the gait to its gait parameter definition.

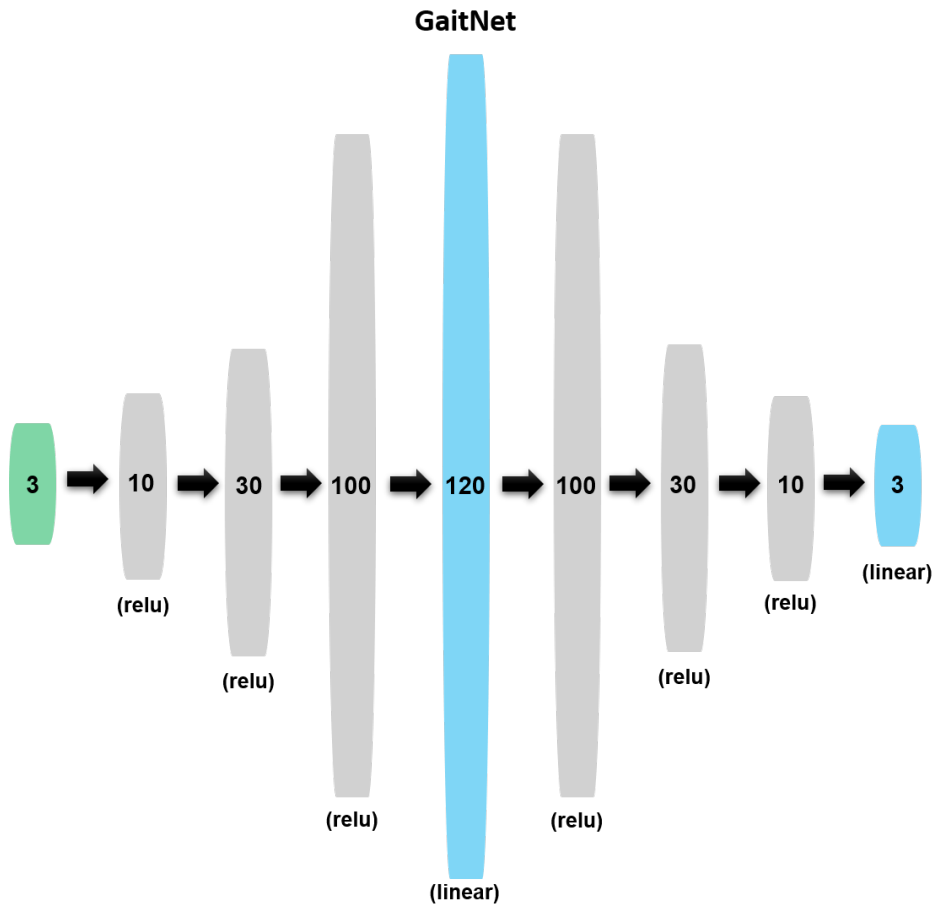


Figure 8.1: Gait-Net Outline: The input layer is highlighted in *green* while the two jointly trained output layers are highlighted in *blue*.

***Remark* 8.2.1.** *We use* relu *activations for all the intermediate layers except the two output layers as both the gait features and gait parameters admit negative values. Therefore, no non-linear activation is used for there two layers.*

8.2.2 Performance Analysis

We benchmark Gait-Net's accuracy and inference time against three baselines described below:

- Simple Network (SN): SN is a fully-connected network built using only the first five layers of the Gait-Net. This denotes traditional supervised regression without the additional gait parameter reconstruction component. Simple fully connected neural network was previously used for gait learning in [166] and shown to be better than Interpolation.
- Principle Component Analysis (PCA): PCA is performed on the gait data $\mathbf{B}$ to obtain a lower dimensional embedding $\widehat{\mathbf{B}} \in \mathbb{R}^3$. Further, a linear transformation $\mathbf{W}$ is obtained using

ordinary least squares regression to fit $\widehat{\mathbf{B}} = \mathbf{WP}$. Therefore, we get $\mathbf{W} = (\widehat{\mathbf{B}}^{\mathbf{T}}\widehat{\mathbf{B}})^{-1}\widehat{\mathbf{B}}^{\mathbf{T}}\mathbf{P}$. Finally, for a given any $\mathbf{p_i} \in \mathbf{P}$, we reconstruct the corresponding $\mathbf{b_i} \in \mathbf{B}$ using $\mathbf{Wp_i} \in \widehat{\mathbf{B}}$.

- Interpolation: Interpolation is a popularly used technique to generate a continuous policy from gait libraries in [166, 163, 164, 13, 161].

	Gait-Net	**SN**	**PCA**	**Interpolation**
***Interpolation* MSE**	0.00036	0.00032	0.00225	0.00023
***Interpolation* Max Error**	0.06646	0.06547	0.11864	0.07070
***Extrapolation* MSE**	0.00065	0.00056	0.00281	0.00678
***Extrapolation* Max Error**	0.07683	0.07321	0.11791	23.2368
Average Inference Time (s)	0.00172	0.00113	0.00154	0.00038

Table 8.1: Benchmark of Online Gait Generation of Squared Error (SE) and Consumed Time

All the methods are evaluated in two tasks:

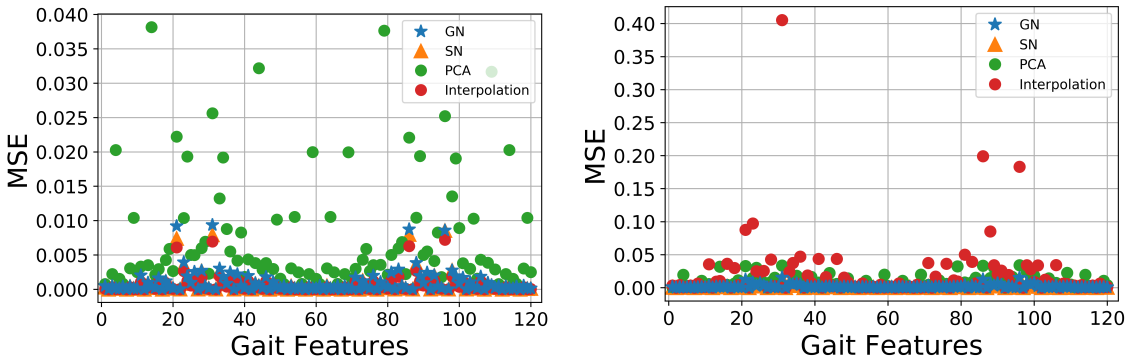
- *Interpolation*[1]: The test-set gait parameters are within the training-set range; and
- *Extrapolation*: The test-set is chosen to be outside the training-set range, specifically, test-set is $\mathbf{P}^{Test} = (V_x^{Test}, V_y^{Test}, SH^{Test}) \in \{\pm 1, \pm 0.8\} \times \{\pm 0.3, \pm 0.27\} \times \{\pm 0.15, \pm 0.12\}$ while the training-set is its complement.

We compare all the methods on two metrics for *extrapolation* and *interpolation*, 1) mean squared error (MSE), and, 2) maximum test error. Additionally, we measure average inference time for each method over 1000 runs. These results are tabulated in Table 8.1.

***Remark* 8.2.2.** *Note that, across all metrics, Gait-Net performance is close to that of the Simple Network. While taking only a minor hit in prediction performance, Gait-Net offers an additional metric to evaluate prediction quality. Note that, it still outperforms Interpolation method.*

For both *interpolation* and *extrapolation* both PCA and Interpolation are on average an order of magnitude worse than Gait-Net. While the dataset shows mostly linear trends, sharp nonlinearities exist in some sub-spaces. The nonlinear function approximation ability of Gait-Net is able to better approximate them. The maximum error values of PCA, and of Interpolation in particular, underscore this fact. Finally, note that, Gait-Net is ten times faster than Interpolation. The time complexity of Interpolation is $O(k \log(n))$, where k denotes number of gait parameters and n is the number of samples per gait parameter. Moreover, the space complexity is $O(n^k)$ which is worse, especially on hardware. Clearly, with increase in features or data size in $\mathbf{P}$, the *real-time* performance gap will widen further between Interpolation and Gait-Net.

[1]To avoid confusion, hereafter, *Interpolation* refers to the task of predicting intermediate gaits whereas Interpolation refers to the Linear Interpolation method.



(a) (b)

Figure 8.2: MSE across gait features: (a) *Interpolation* MSE and (b) *Extrapolation* MSE across the 120 gait features.

To further examine prediction performance we plot MSE for all the four methods at the individual features level in Fig 8.2. These individual features of the predicted gait are the 120 bézier parameters used to generate a two step walking motion, 60 each for right and left stance. At the feature level too we notice similar trends, Gait-Net is slightly worse than Interpolation for within range samples and consistently better at approximating for out-of-range samples.

As a final analysis step, we estimate the upper bounds of *Lipschitz* constants for both the Simple Network and Gait-Net. Neural networks are well-known to be sensitive to well-chosen small perturbations, often called *adversarial attacks*. This is a serious issue when they are being used to predict gaits for a human-sized bipedal robot. Any failure mode during execution could not only damage the robot but also endanger humans nearby. Lipschitz constant is an important metric to evaluate policy smoothness. Recently, in [168], it was shown that for any k-layer feed-forward network with ReLU or linear activations, the Lipschitz constant is upper bounded by,

$$\widehat{L} = \prod_{i=1}^{k} ||M_i||_2, \tag{8.1}$$

where, $\widehat{L}$ is the Lipschitz constant upper bound and M_i is the weight matrix for layer-i. We compute $\widehat{L}$ for both Simple Network and Gait-Net. We have $\widehat{L}_{GN} = 2366.5$ whereas $\widehat{L}_{SN} = 3402.1$. Note that this are just upper bounds and still highlight the robustness of Gait-Net over the Simple Network. For accurate estimation of Lipschitz constants, see [169].

***Remark* 8.2.3.** *The lower Lipschitz constant for Gait-Net is due to the* natural *regularization by the auto-encoder's reconstruction loss. In [170], the authors coined the term* supervised autoencoders *to refer to architectures similar to the Gait-Net.*

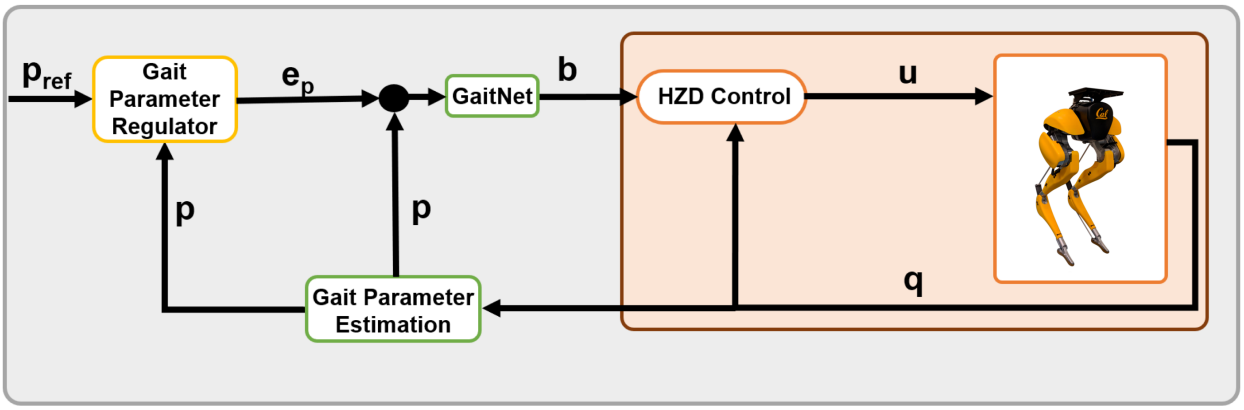


Figure 8.3: Walking Policy Overview: Gait-Net provides gaits for the HZD based controller to track in closed-loop. Here, q is the state vector defined in (3.12) and u is the input vector defined in (3.14). The outer-loop regulates the de.

8.3 Cassie Walking Simulation

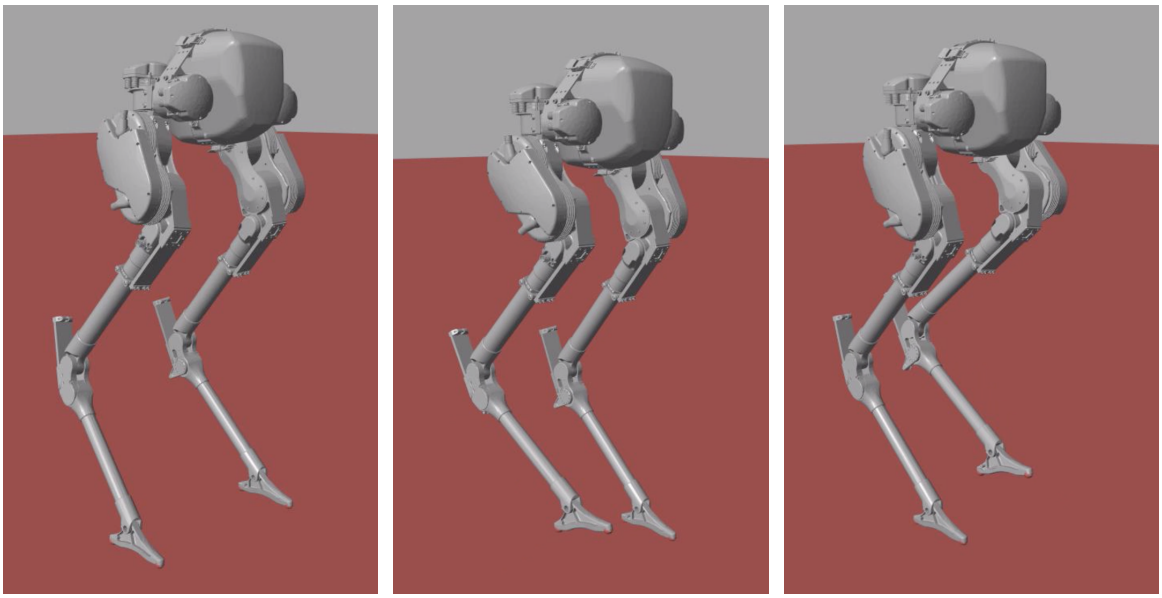
After training and evaluating the Gait-Net, we now proceed to validate its performance for robust bipedal robot locomotion. Before deploying Gait-Net onto Cassie, we first test it using a high-fidelity physical simulator of Cassie built in MATLAB Simulink. Further, we used an open-source Cassie walking controller from [161] to implement the underlying HZD controller[2]. The high bandwidth controller operates at 2kHz. The overall closed-loop walking control is outlined in Figure 8.3. Given an open-loop walking plan defined in terms of (V_x, V_y, SH), the Gait-Net describes a walking gait. The HZD-based controller then executes this closed-loop walking behavior using joint angle feedback.

The gait parameters from the walking controller were then fed into Gait-Net to online predict gait libraries. Figure 8.4 demonstrates preliminary simulation results of the Cassie bipedal walking using the Gait-Net. In the snapshots, Cassie is able to transition from a standing in place configuration to stepping in place ($V_x = 0, V_y = 0, SH = 0$) and then walking forward ($V_x = 0.5, V_y = 0, SH = 0$). A numerical comparison between Gait-Net and online linear interpolation is presented in Figure 8.5. For the reference tracking task, the Gait-Net achieves a better performance while the interpolation method has a steady state error.

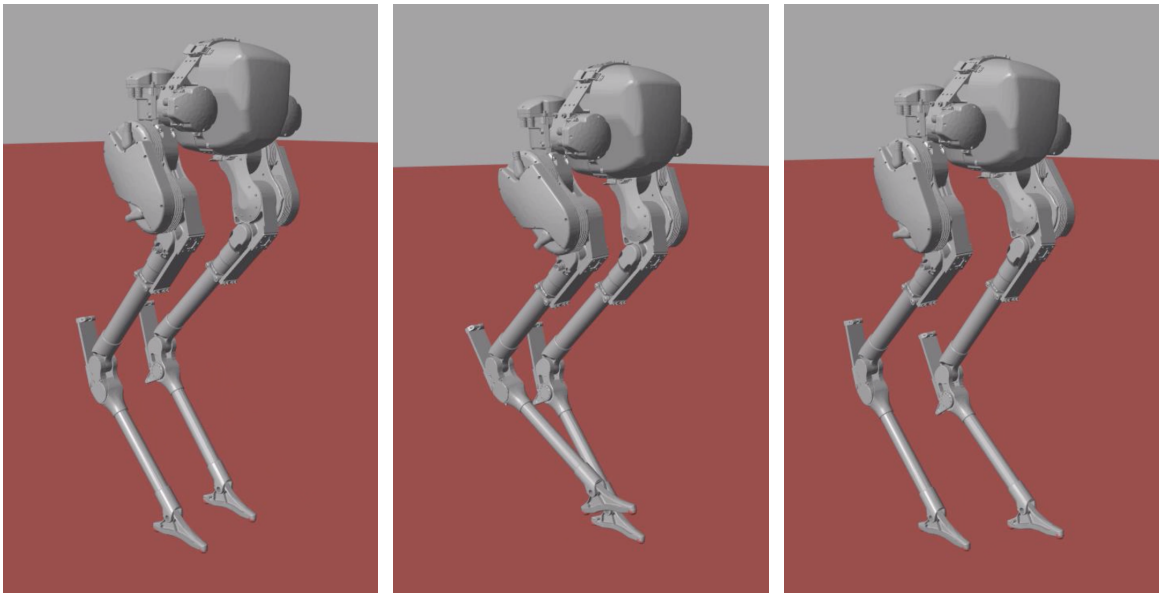
8.4 Summary

In this chapter, we proposed GaitNet a data-driven policy to generate *approximately* stable walking motions involving forward and lateral speed regulation and step height adaptation. Gait-Net is learned from a Gait Library developed using full robot dynamics to execute a range of periodic walking motions that respect strict joint and motor limits, unilateral ground contact and friction constraints. Gait-Net was successfully tested on the Cassie bipedal robot in simulation.

[2] https://github.com/UMich-BipedLab/Cassie_FlatGround_Controller



(a) (b) (c)



(d) (e) (f)

Figure 8.4: Walking Simulation Snapshots: From (a) to (c), the robot is seen stepping in place. From (d) to (f), the robot transits to a forward walking gait.

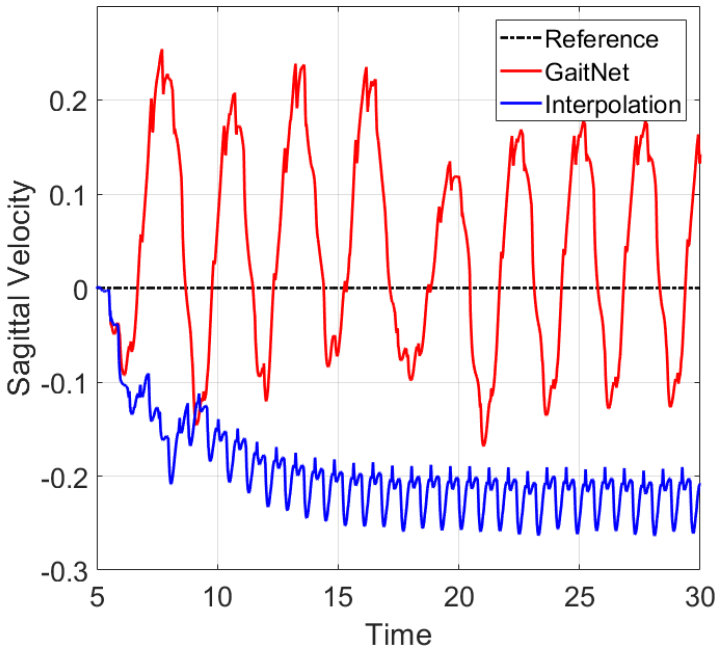


Figure 8.5: Forward velocity regulation plot. Gait-Net achieves a better performance while the interpolation method has a steady state error.

8.4.1 Limitations

We have only demonstrated preliminary results of using Gait-Net on the robot. We still need to fully evaluate Gait-Net's prediction performance, particularly validating its *extrapolation* ability on hardware. Further, we haven't yet utilized the proposed gait estimation quality metric provided by the autoencoder part of the Gait-Net. A high-level planner needs to be developed that can leverage this information. Still, the autoencoder serves as a good regularization for better policy learning. This needs to be ascertained formally or empirically. As future work, we wish to focus on addressing these limitations.

Chapter 9

Deep Perception for Autonomous Walking

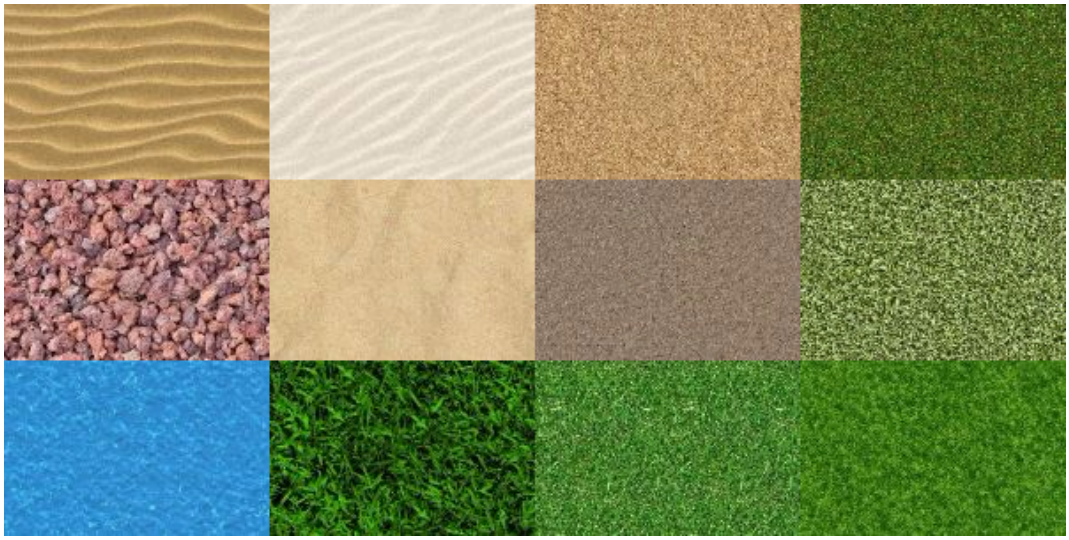
Dynamic bipedal walking on discrete terrain, such as stepping stones, with a human-sized under-actuated robot is a challenging problem needing sophisticated safety-critical controllers [13]. These controllers in-turn need a fast, accurate, and reliable sensory feedback to guarantee safe landing of the robot's swing foot on the nearest foothold. We believe that recent advances in parallel computing and deep learning could aid in achieving our goals of high speed and accuracy in visual perception for bipedal locomotion. Having presented the robot's dynamical model in 3.1 and the gait-library-based controller (refer to 3.1.1) for walking on discrete terrain, we will now present a systematic way to build and train a deep visual perception model that estimates the step length from a single camera image. The system will take an input of the upcoming terrain through a front-facing camera at the beginning of each step. This image is fed to a convolutional neural network (CNN) to estimate the step length to the next step. Estimated step length is then fed to the gait-library-based controller to enable the robot to precisely land on the next foothold.

This CNN-based deep perception model has two critical components. Firstly, we need a large corpus of step-length annotated imagery of the robot's front person view while walking. This dataset is used to train the model for accurate step length estimation. The systematic methodology used to create this synthetic dataset is described in the next subsection. Secondly, we need a suitable deep neural network architecture that can best approximate the complex non-linear mapping from image to step length estimate. The network needs to be tuned methodically to not only obtain the best test accuracy but also bound the worst-case prediction. These details are summarized after the next subsection.

9.1 Visual Simulator for Synthetic Dataset Generation

To generate the image dataset, we use a popular open-source graphics software called Blender [171] generate realistic scenes through scripting. To create a discrete terrain scene, we need four key details: 1) Camera location and intrinsic parameters 2) Stepping stone location, 3) Lighting model and location, and 4) Color and texture information of both the stepping stone and background. These parameters will be randomized in ranges larger than what the robot may encounter in order to account for error accumulation over time.

The above four parameters are randomized for image generation in the following manner:



(a) Background Textures

(b) Stone Textures

Figure 9.1: A collage of textures use for the synthetic outdoor dataset generation: (a) Background Textures, (b) Stone Textures.

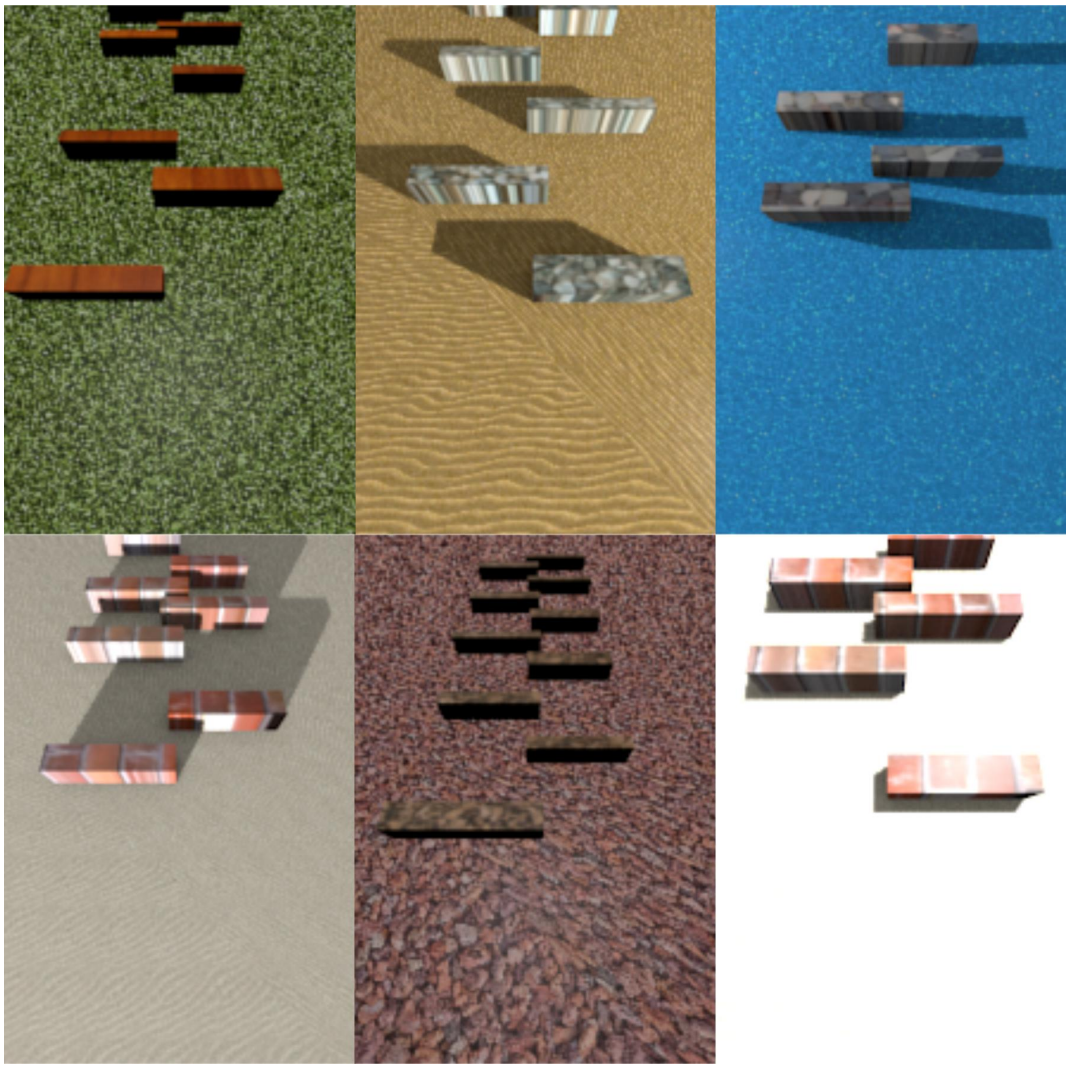


Figure 9.2: Sample images from the *Synthetic Outdoor Dataset* (SOD).

Camera Location: The camera location is measured with respect to the stance foot position. For each image, we randomly sampled from a range of $[-10:20] \times [-10:10] \times [80:120]$ (cm) to obtain the x, y, z offsets of the camera from the stance foot, respectively. The ZED Stereo Camera [172] model is used for rendering the images. However, since we only focus on the step length, we only generate monocular images for this study.

Stone Location: From [13], we note that, the robot was able to walk on a discrete terrain where the step lengths ranged from $[20:90]$ (cm). True step length is the distance from the robot's stance foot to the next stone's center. For this work we uniformly sample step lengths from $[15:95]$ (cm) range and arrange the stones in a single column accordingly. Based on the width of the robot, we spread out stones $[30:35]$ (cm) away of the robot's center-line. Moreover, they are alternately positioned on either sides of the center-line, as shown in Fig. 9.2. Finally, the stone size itself is varied randomly between $[10:20]$ (cm) in length and $[60:90]$ (cm) in width, respectively. However, the stone height is kept fixed at 15 cm.

Lighting Properties: The light source is always facing the current stepping stone. However, its position is randomly chosen from a semi-spherical dome-like space above the robot with a 6 meters constant radius to the current stone-center.

Texture: Here we select real world textures for both stones and background. We chose 12 unique textures comprising of grass, sand, water, pebbled terrain, etc. for the background and 14 unique textures for the stone including granite, cement, brick, wood, etc. Texture samples are shown in Fig. 9.1 Each scene is rendered by randomly sampling a texture pair from the available collection.

The images are rendered with a resolution of 223×223 pixels. Additionally, we crop the left and right 15% of the image to further reduce computational overhead. The final image resolution is 223×149. We generate $40,000$ images and call it the Synthetic Outdoor Dataset (SOD). Sample images are shown in Fig. 9.2. Having presented the dataset generation details, we will present the neural network architecture design next.

9.2 Custom Deep Neural Network Design

For training our object detector, we propose a custom neural network architecture called *SL-CNN*. It consists of six convolutional layers of 32 filters each, followed by two dense layers, both with 256 neurons each. Unlike traditional designs where the number of features maps increase with depth, we found that, a constant number of feature maps throughout does better and needs fewer parameters. The kernel size of each filter is (4×4). Batch Normalization and Max Pooling are applied after every two convolutional layers. Additionally, we apply Batch Normalization just before the final output layer as well. We use *relu* activations in all the layers except the last one, where we use a *linear* activation function instead. Fig. 9.3 summarizes the CNN architecture. The detector is trained using the *Mean Squared Error* loss and the *Adam* optimizer. We use a learning rate of $1e-4$ along with a suitable learning rate decay policy. We use Keras API [173] with TensorFlow backend [174] to build and train our model. We trained for 40 epochs with a batch size of 50, on a Intel i7 machine with an NVIDIA Titan X GPU. The model has roughly 2.5 million trainable parameters. Finally we split the dataset into Training, Validation and Test sets, each comprising of $28900, 5100, 6000$ images, respectively.

The step length is estimated as follows: Using the joint encoders on the robot, location of the camera is first estimated. The deep neural network only learns to predict the distance from camera to the next stone center. Therefore, the predicted step length is the sum of these two distances. Note that the camera position information is not used during training.

While deep networks have remarkable function approximation abilities, they have many hyper-parameters whose fine-tuning critically impacts network performance and generalization ability. In the next section, we systematically outline our network design and customization process while examining the impact of each hyper-parameter on reducing worst-case test error.

9.2.1 Hyper-parameter Search

Deep neural networks have a very high dimensional hyper-parameter space, where almost every single building block can be optimized. Most papers use existing architectures and leverage their transfer learning properties. Few papers explain in sufficient detail, the impact of each hyper-

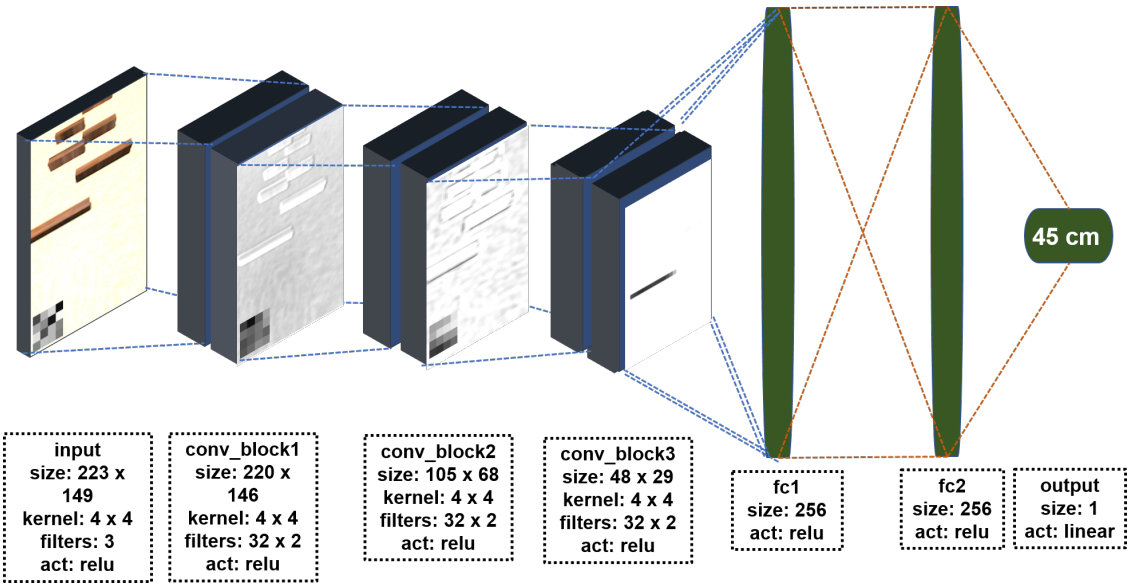


Figure 9.3: A schematic of SL-CNN. Each convolution block consists of two layers followed by a Max-Pool and Batch Normalization. Two identically sized fully connected layers are used. Finally, the output activation function is linear. Sample output activation maps are overlayed on each convolution block along on the learned filters.

parameter on the learning outcomes. Unfortunately, these choices don't generalize well to all problems and it is worthwhile to carefully tweak and specifically examine them for individual problems. Important insights drawn from this exploration for our problem are summarized below. Note that, all the results reported below are on the test set.

- Roughly 89% drop in error occurs within the first 20 epochs. Therefore, a wider hyper-parameter coarse search was carried out on models trained for 20 epochs while for the finer search, the models were trained for 40 epochs.
- As already identified in [96], Dropout with any probability or placement in the network worsens performance.
- Batch Normalization improves learning and results in around 33% drop in mean absolute error. More interestingly it leads to over 55% drop in the worst-case prediction error (or the maximum error on the test set).
- Using L2 Regularization (default is 0.01) for only the fully connected layers helps further reduce the worst-case prediction error.
- We tested the Architecture with 5 kernel sizes, $(2, 2), (4, 4), (6, 6), (3, 3), (2, 4)$. We observed that rectangular kernels had the largest worst-case error, followed by the $(3, 3)$ kernel. Surprisingly, even numbered kernels did a better job, against conventional wisdom. The best kernel was $(4, 4)$.

Test Avg. Loss (in cm)	Std. Dev. of Loss	% Above 5 cm	Max Pred. Error (in cm)
1.618	1.32	2.116	10.38

Table 9.1: Summary of prediction performance on test data.

- Adding an extra convolution block (ie., two additional layers followed by a max pool and a batch norm) increased the mean absolute error. Unlike classification tasks where depth almost always helps, in regression tasks, localization accuracy is affected by max-pooling layers beyond some depth. Therefore, for regression tasks one must find the sweet spot between depth (complexity) and accuracy.

Finally, based on the above mentioned hyper-parameter search, an optimal network architecture is designed and it is trained with the synthetic outdoor dataset. The qualitative and quantitative results obtained are presented in the next section.

9.3 Results and Discussion

In this section, we analyze the performance of SL-CNN and later integrate it with a physics-based simulator and gait-library based controller to numerically realize and analyze autonomous dynamic walking on randomly generated discrete terrain.

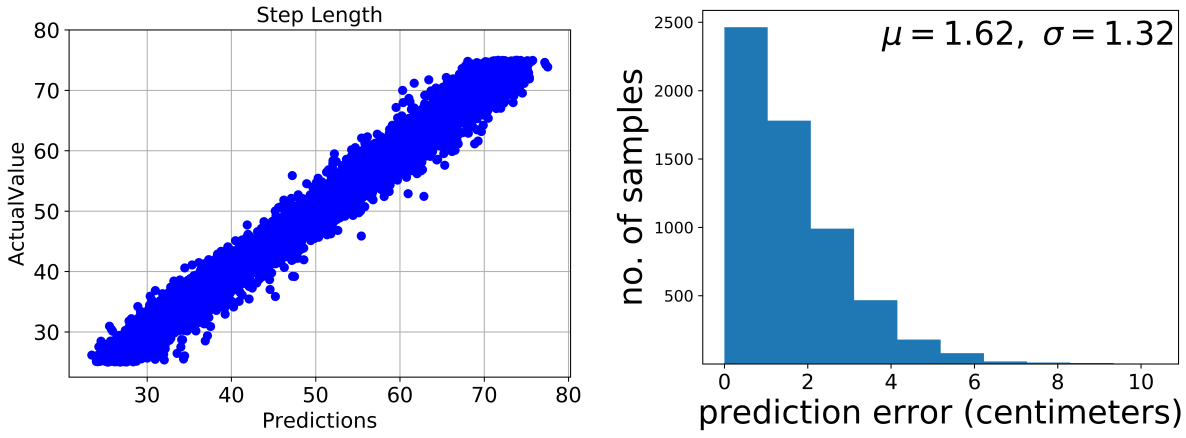
9.3.1 Step Length Prediction Performance

Once trained, we expect the step length predictor to accurately detect the next stepping stone and output it's distance in centimeters. Note that, each image will have anywhere between $1-5$ stepping stones. Therefore, even though they have the same texture and geometry, our perception framework needs to overlook other stones and actively seek out the first one. In this situation, we believe perspective distortion helps in better distinguishing the stone of interest. Further, due to the safety-critical nature of this problem, in addition to describing network performance based on mean squared error, we will also report the variance and the worst-case prediction.

Error is unavoidable in function approximation. However, in a safety-critical scenario like discrete terrain walking, the default approach of choosing the network that gives the least mean squared error could be detrimental as the worst-case estimate could still be off the safety limits. In order to avoid this issue, in this work, hyper-parameters were tuned with the objective to find the least possible worst-case estimate with the available dataset. The performance of the CNN was evaluated on the 6000 sample test data and is visualized in Fig. 9.4 and summarized in Table 9.1.

Best and Worst predictions:

In addition to studying the qualitative learning outcomes like average loss, standard deviation of loss, etc., we also visualize images of the best-8 and the worst-8 predictions in Fig. 9.5 to visually interpret which parameters the model could and couldn't generalize over. From the figure, it is clear that the model is able to generalize over the various foreground and background textures, including bright and dim lighting conditions. However, shadows contributed to some of the higher estimation errors.



(a) Predicted v/s Actual Step Length

(b) Prediction Error Histogram

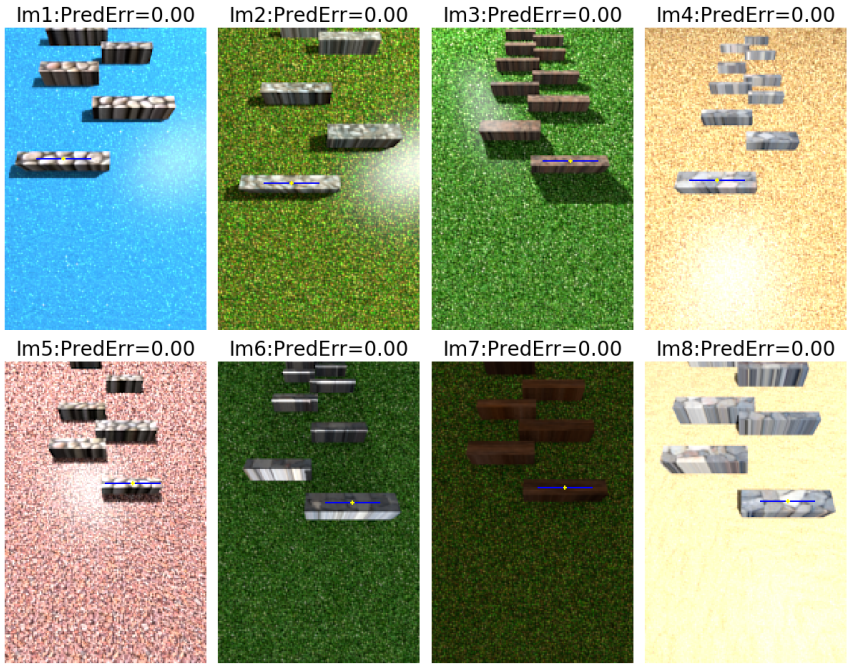
Figure 9.4: Visualizing step length prediction performance through plots of (a) Predicted versus True Step Length values and (b) Prediction Error Histogram. Both plots are for the test data obtained from the Synthetic Outdoor Dataset.

9.3.2 Simulation Results

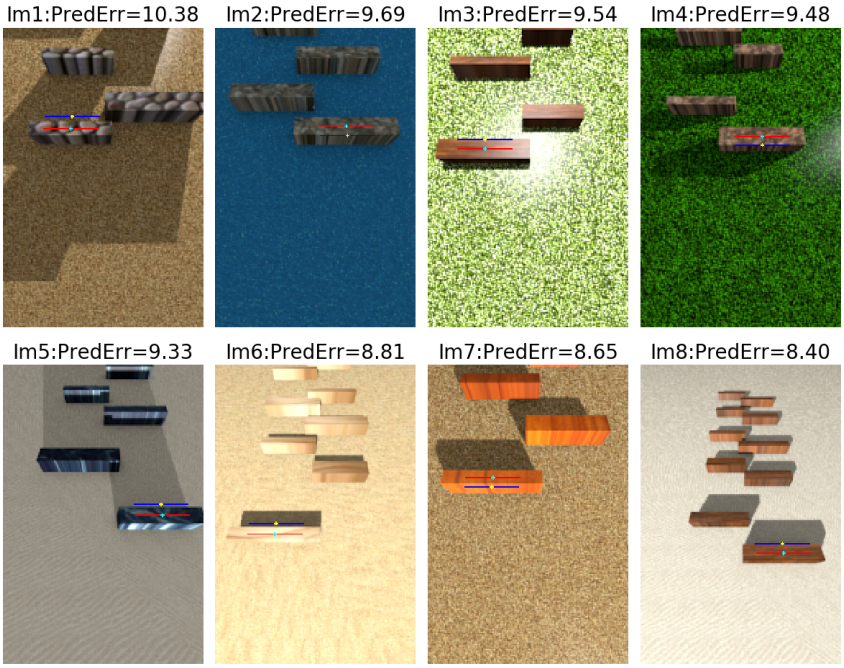
In this section, we integrate SL-CNN with a physics-based robot simulator in 3.1 and the gait-library based controller in 3.1.1 to evaluate the closed-loop autonomous operation of the robot. The robot dynamics are simulated in Matlab. At the beginning of each step, robot's current position (specifically the camera position) is supplied to Blender to render a first-person-view synthetic outdoor image using the information from the terrain generator. This image is in turn supplied to the Convolution Neural Network (implemented in Keras and TensorFlow) to predict the step length for the next step. Provided with this predicted step length, the closed-loop dynamics of the robot and controller are simlated for one step to enable the robot to take the step forward. The process is repeated for subsequent steps. A schematic of the numerical simulation pipeline is shown in Fig. 9.6.

Numerical simulations are carried out with stones randomly placed with the inter-stone distance within the $[45 : 85]$ cm range. Recall that the camera position and stone location were uniformly sampled from dissimilar ranges in Section IV. The label used for training the CNN is the distance to the camera which is difference of stone location and camera position. Therefore, the distribution of labels used for training is no longer uniform. Accounting for this fact, the step length ranges have been adjusted to only test the CNN in the range where there was enough data to guarantee a good learning outcome.

In this study, we render images with light fixed in an overhead position and focus on carefully examining sensitivity to potential failure modes like camera position or step length going outside the range used for dataset generation during continuous simulation. (We have already noted earlier that shadows are a failure mode.) Our evaluation is based on two metrics, 1) Perception Error and 2) Foot Placement Error. Prediction Error is defined as the difference between Predicted Step Length and True Step Length and it is purely an artifact of the perception module. Similarly, Foot Placement Error is defined as the difference between foot-contact-point and stone-center. This



(a) Best-8



(b) Worst-8

Figure 9.5: Snapshots of the best-8 and worst-8 predictions of the neural network alongwith the corresponding prediction error in centimeters. The desired (red line) and predicted (blue line) step lengths are marked along with pixel coordinates of the resulting foot placement location (yellow dot).

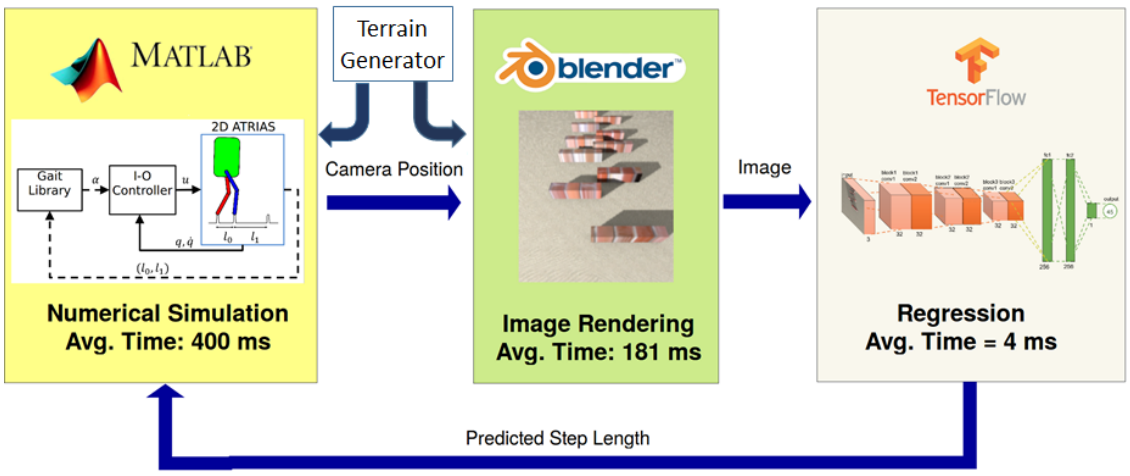
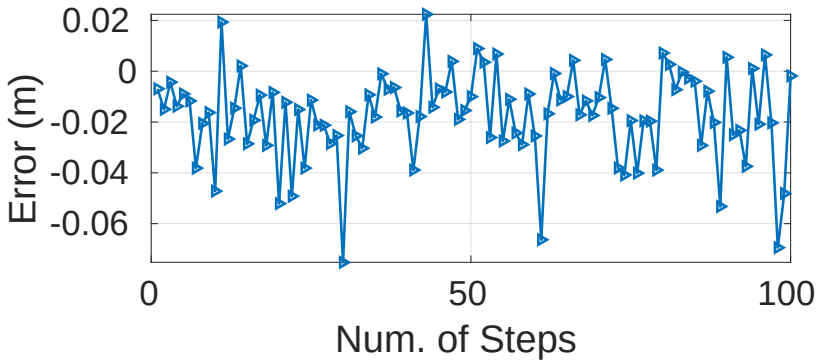
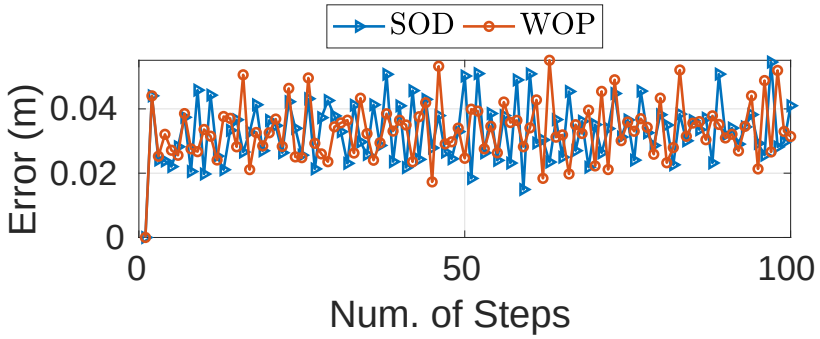


Figure 9.6: Simulation pipeline for autonomous dynamic walking on discrete terrain. The pipleline integrates gait optimization, nonlinear control, vision, and deep learning. Simulation Video: https://youtu.be/ijJAPapU7qI.



(a) Prediction Error: Predicted Step Length - Desired Step Length



(b) Foot Placement Error: Actual Step Length - Desired Step Length

Figure 9.7: (a) Prediction Error, and (b) Foot Placement Error plots for 100 step walking simulation with step lengths varying within $[45 : 75]$ cm. In (b) WOP indicates error *without perception* while SOD indicates errors when step length is estimated using the *synthetic outdoor dataset*-based image preview.

error is the cumulative error of both perception and control.

The robot was simulated for 100 steps and these two errors are plotted, as shown in Fig. 9.7. Note that the prediction error is bounded within an 8 cm range from the center for the most part and doesn't show significant accumulation of error over time. In Fig. 9.7(b), the foot placement error is also plotted to compare stepping performance without (called WOP - With Out Perception) and with perception (using SOD - synthetic outdoor dataset).

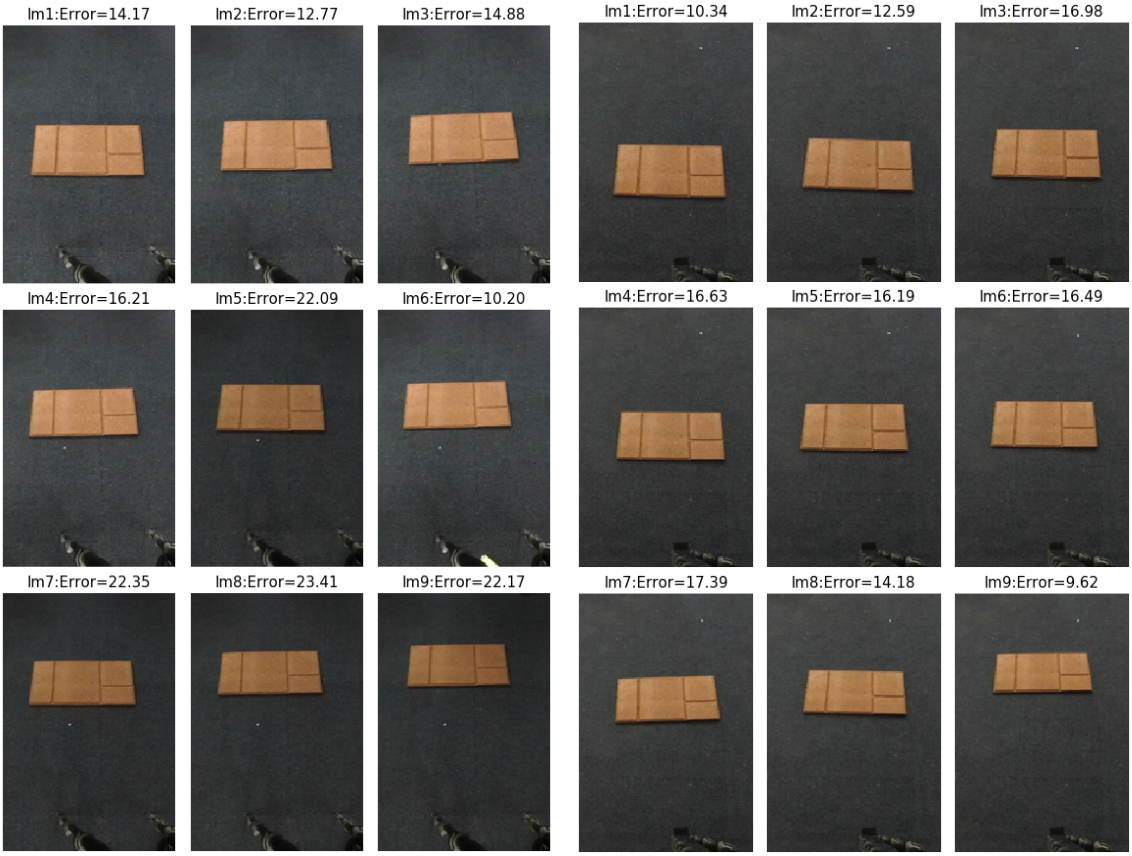
9.4 Sim2Real Performance Evaluation

The main objective of this work was to develop a synthetic dataset of discrete terrain images and train a deep perception module with it such that its prediction performance can generalize to real world data. In order to achieve this we used the Domain Randomization technique [96] for dataset generation. Before porting the deep perception module onto the hardware, we first benchmark its *sim2real* performance on a small real-world dataset generated using the ZED camera [172] attached to a tripod. Sim2Real is a common term used in the learning community while referring to generalization ability of data-driven models which are trained using synthetic data but applied on real-world data at test time. For fair comparison, we kept the camera properties consistent for both the synthetic and real datasets. Note that, the synthetic dataset was also generated using the intrinsic properties of ZED camera. Each image contains a single tile placed between 35 to 85 centimeters away from the camera (distance is measured to the center of the tile). In each dataset the tile moves further away from the camera from the first image to the last image. Individual step length distances are randomly chosen. Further in the first dataset, called *Batch-1*, the camera is oriented such that the tripod legs are visible in the image (mimicking the cases where the robot's legs are visible). In the second dataset, called *Batch-2*, the camera orientation is slightly adjusted to minimize tripod visibility. Since the synthetic dataset was trained assuming zero visibility of the legs, we wish to evaluate the sensitivity of this artifact in the prediction performance of the neural network. Collages of both *Batch-1* and *Batch-2* are shown in Fig. 9.8a and Fig. 9.8b, respectively. The prediction performance of SL-CNN on the two datasets is summarized in Table 9.2. As noted,

Metrics	Batch-1	Batch-2
Error Mean (in cm)	17.743	14.451
Error Standard Deviation (in cm)	7.017	2.937

Table 9.2: Prediction error statistics using real world data.

the deep perception module generalizes poorly than expected onto real world data. In particular, the presence of tripod legs further worsens performance (compare error statistics in between Batch-1 and Batch-2 in Table 9.2). This should be another parameter to randomize during training. We conclude that the sim2real performance transfer is still poor for the deep perception module. We shall discuss potential improvements in the Future Work section.



(a) Batch-1 (b) Batch-2

Figure 9.8: Snapshots of the real world datasets, Batch-1 and Batch-2, used for testing the deep perception module performance. The prediction error on each image is added to the image title.

9.5 Summary

In this chapter, we first built a realistic visual simulator to generate the robot's first person view while walking over stepping stones and combined it with an accurate physics simulator of the bipedal robot. The physics simulator also contains an inner-loop safety-critical controller that can generate stable and safe limit cycle walking for a desired step length. Note that, we limited our attention to only autonomous planar walking, and accordingly, only predict step length information. This simplification allowed us to keep the focus on the visual simulator development, convolutional neural network customization (in order to bound worst-case estimate) and perception-control integration. However, the pipeline can be extended to 3D walking without any loss of generality. Note that, even though we used a planar robot model in the physical simulator, the deep perception module presented here takes an image rendered from a 3D scene as input without making any geometric simplifications that stem from planar walking. Next, we trained a deep neural network to estimate the step length (distance to the next stepping location) using a single sampled image preview that is obtained at the beginning of each step. Detecting footholds and estimating distance

is a classic object localization problem similar to object grasping in robotic manipulation, however in the case of locomotion there are additional challenges due to the time-critical and safety-critical nature of the problem. Failures can lead to very adverse consequences and there is very slim scope for recovery as the robot is operating in a dynamic regime around an unstable equilibrium. The deep neural network was customized keeping these challenges in mind and with an objective to minimize its worst-case performance. Finally, we show promising results with the network predicting step length with the worst-case error capped at 11 cm. Moreover, when the perception and control pipelines are integrated and tested in simulation, the robot was able to walk at least 100 steps without failure. Despite it's promising performance in simulation, the deep perception module generalizes poorly to real world data. We conjecture that a larger dataset using a wider range of textures and colors, adding occlusions like robots legs, etc. can improve performance and we will investigate it as a part of future work.s

Chapter 10

Part III Conclusions and Future Work

In Part III, Learning for Dynamic Legged Robots, two data-driven models were presented to offer greater versatility and autonomy to legged robots. In Chapter 8, we introduced Gait-Net, a data-driven open-loop policy to generate stable walking motions involving forward and lateral speed regulation and step height adaptation. Gait-Net is learned from a Gait Library developed using full robot dynamics to execute a range of periodic walking motions that are dynamically feasible and respect strict joint and motor limits, unilateral ground contact and friction constraints. Open-loop walking policies generated by Gait-Net could enable Cassie to climb stairs, walk on uneven or sloped terrain and safely navigate crowded urban environments. Later, in Chapter 9, a CNN-based predictor, called SL-CNN, was developed to estimate step lengths for a dynamic bipedal walker operating in discrete terrain from visual input. It was empirically shown that a feed-forward gait adjustment based on intermittent visual feedback is sufficient to walk on a discrete terrain where high-speed prediction and accurate foot placement are critical. Several visual factors that impact the predictor's performance are identified for further refinement.

10.1 Future Directions

Both Gait-Net and SL-CNN were developed in a robot agnostic fashion. Therefore, they can be customized to other legged robots following the same design and development pipeline. However, there is still room for improvement in both and that directs future work as follows:

10.1.1 Test on Hardware

Gait-Net and SL-CNN need to be tested on real robots. In Section 9.4, it was noted that the SL-CNN prediction performance on real world data is still poor. There are multiple ways to improve it.

1. Generate a much larger and more diverse synthetic dataset using Domain Randomization. Particularly, add occlusions, like robot legs, increase color and texture distributions for the stones and backgrounds, vary stone shapes and sizes.

2. Provide camera height and orientation also as inputs to the network in addition to the image.

3. Leverage recent advances in object detection like [175] to first detect the stones and then localize. This way, the neural network is provided more context during training to learn a more robust predictor.

4. Since multiple steps are visible in an image, accuracy could be improved using Recurrent Neural Networks. Instead of training the network with a single image from step $N-1$, it can be trained with images from steps $N-2$ and $N-1$ to predict the step length to step N. Further, most bipedal robots used in research today are equipped with a range of other sensors like IMUs, LiDARs, etc. Using mulit-modal data is yet another way to improve predictive power.

10.1.2 Learning Visuomotor Policies for Walking

The Gait-Net and SL-CNN architectures could be combined and trained jointly to obtain an image-to-gait mapping that can generate perception-driven open-loop walking policies. This serves as an important first step towards emulating human locomotory behaviors. Here, we leverage physical models for rapidly generating gait data and synthetic image data, and learned models to unify the two via end-to-end training.

10.1.3 Gait-Net in DRL

Recently, [176] cast the HZD-based bipedal robot gait optimization as a deep reinforcement learning (DRL) problem, and later extended it to the Cassie robot in [177]. While successful at learning stable and robust policies for planar walking in simulation, it's subsequent extension to tackle the complexities of full $3D$ walking remains to be seen. Our proposed Gait-Net can aid and speed up learning in this DRL setting as well. More importantly however, we believe DRL is better suited to learn robust gait transitions and higher-level plans using gait parameters. Note that the nonlinear trajectory optimization is much faster at generating stable gaits that can strictly satisfy all the friction, unilateral contact and torque limit constraints.

Bibliography

[1] G. Hodan. Bolton abbey stepping stones. x, 4

[2] Miomir Vukobratović and Branislav Borovac. Zero-moment point—thirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157–173, 2004. 2, 11, 13

[3] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 239–246. IEEE, 2001. 2, 8, 13, 29, 63

[4] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, pages 1–27, 2015. 2, 9, 12

[5] Siyuan Feng, X Xinjilefu, Weiwei Huang, and Christopher G Atkeson. 3d walking based on online optimization. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 21–27. IEEE, 2013. 2, 9, 12, 13

[6] JW Grizzle, Jonathan Hurst, Benjamin Morris, Hae-Won Park, and Koushil Sreenath. Mabel, a new robotic bipedal walker and runner. In *American Control Conference, 2009. ACC'09.*, pages 2030–2036. IEEE, 2009. 2, 13

[7] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and Jessy W Grizzle. A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel. *The International Journal of Robotics Research (IJRR)*, 30(9):1170–1193, 2011. 2, 3, 11, 13, 78, 79

[8] Russ Tedrake, Teresa Weirui Zhang, Ming-fai Fong, and H Sebastian Seung. Actuating a simple 3d passive dynamic walker. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4656–4661. IEEE, 2004. 2, 9, 12

[9] Martijn Wisse. Three additions to passive dynamic walking: actuation, an upper body, and 3d stability. *International Journal of Humanoid Robotics*, 2(04):459–478, 2005. 2

[10] Pranav A Bhounsule, Jason Cortell, Anoop Grewal, Bram Hendriksen, JG Daniël Karssen, Chandana Paul, and Andy Ruina. Low-bandwidth reflex-based control for lower power

walking: 65 km on a single battery charge. *The International Journal of Robotics Research (IJRR)*, 33(10):1305–1321, 2014. 2, 9, 12

[11] Ruta Desai and Hartmut Geyer. Robust swing leg placement under large disturbances. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 265–270. IEEE, 2012. 2

[12] Brent Griffin and Jessy Grizzle. Walking gait optimization for accommodation of unknown terrain height variations. In *American Control Conference (ACC)*, pages 4810–4817. IEEE, 2015. 2, 13

[13] Quan Nguyen, Xingye Da, William Martin, Hartmut Geyer, Jessy W. Grizzle, and Sreenath. Dynamic walking on randomly-varying discrete terrain with one-step preview. In *Robotics: Science and Systems (RSS)*, 2017. 2, 69, 79, 81, 86, 88

[14] William C Martin, Albert Wu, and Hartmut Geyer. Experimental evaluation of deadbeat running on the atrias biped. *IEEE Robotics and Automation Letters*, 2(2):1085–1092, 2017. 2

[15] Erico Guizzo and Evan Ackerman. DARPA Robotics Challenge: A compilation of robots falling down. http://spectrum.ieee.org/automaton/robotics/humanoids/darpa-robotics-challenge-robots-falling, 2015. 2

[16] A celebration of risk (a.k.a., robots take a spill). https://www.youtube.com/watch?v=7A_QPGcjrh0, 2015. 2

[17] Siavash Rezazadeh, Christian Hubicki, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Andy Abate, and Jonathan Hurst. Spring-mass walking with atrias in 3d: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot. In *Proceedings of the ASME 2015 Dynamic Systems and Control Conference*, 2015. 3

[18] Avinash Siravuru and Koushil Sreenath. Nonlinear control using coordinate-free and euler formulations: An empirical evaluation on a 3d pendulum. Submitted to IFAC World Congress. 5

[19] Koushil Sreenath and Amit K Sanyal. The reaction mass biped: Equations of motion, hybrid model for walking and trajectory tracking control, 2015. 5, 17

[20] Shuxiao Chen, Jonathan Rogers, Bike Zhang, and Koushil Sreenath. Feedback control for autonomous riding of hovershoes by a cassie bipedal robot. *arXiv preprint arXiv:1907.11353*, 2019. 5, 27, 62, 68, 71

[21] Avinash Siravuru, Sasi P Viswanathan, Koushil Sreenath, and Amit K Sanyal. The reaction mass biped: Geometric mechanics and control. *Journal of Intelligent & Robotic Systems*, 89(1-2):155–173, 2018. 6

[22] Avinash Siravuru, Zhongyu Li, Bike Zhang, and Koushil Sreenath. Gait-net: Online gait inference for bipedal locomotion. Submitted to PMLR Learning for Dynamics and Control (L4DC). 6

[23] Avinash Siravuru, Allan Wang, Quan Nguyen, and Koushil Sreenath. Deep visual perception for dynamic walking on discrete terrain. In *International Conference on Humanoid Robotics (Humanoids)*, pages 418–424, 2017. 6

[24] Dean A Pomerleau. Alvinn, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, Computer Science Department, 1989. 7

[25] Siddhartha S Srinivasa, Dave Ferguson, Casey J Helfrich, Dmitry Berenson, Alvaro Collet, Rosen Diankov, Garratt Gallagher, Geoffrey Hollinger, James Kuffner, and Michael Vande Weghe. Herb: a home exploring robotic butler. *Autonomous Robots*, 28(1):5, 2010. 7

[26] Amit Kumar Pandey and Rodolphe Gelin. A mass-produced sociable humanoid robot: pepper: the first machine of its kind. *IEEE Robotics & Automation Magazine*, 25(3):40–48, 2018. 7

[27] Aaron M Johnson, G Clark Haynes, and Daniel E Koditschek. Disturbance detection, identification, and recovery by gait transition in legged robots. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5347–5353. IEEE, 2010. 7

[28] Ayush Agrawal, Omar Harib, Ayonga Hereid, Sylvain Finet, Matthieu Masselin, Laurent Praly, Aaron D Ames, Koushil Sreenath, and Jessy W Grizzle. First steps towards translating hzd control of bipedal robots to decentralized control of exoskeletons. *IEEE Access*, 5:9919–9934, 2017. 8

[29] Adam B Zoss, Hami Kazerooni, and Andrew Chu. Biomechanical design of the berkeley lower extremity exoskeleton (bleex). *IEEE/ASME Transactions on Mechatronics*, 11(2):128–138, 2006. 8

[30] Ambarish Goswami, Bernard Espiau, and Ahmed Keramane. Limit cycles in a passive compass gait biped and passivity-mimicking control laws. *Autonomous Robots*, 4(3):273–286, 1997. 8

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

[32] Christine Chevallereau, ABBA Gabriel, Yannick Aoustin, Franck Plestan, Eric Westervelt, Carlos Canudas De Wit, and Jessy Grizzle. Rabbit: A testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, 2003. 9, 13, 79

[33] Siavash Rezazadeh, Christian Hubicki, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Andy Abate, and Jonathan Hurst. Draft: Spring-mass walking with atrias in

3d: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot. 2015. 9, 74

[34] Brian G Buss, Amin Ramezani, Kaveh Akbari Hamed, Brent Griffin, Kevin S Galloway, Jessy W Grizzle, et al. Preliminary walking experiments with underactuated 3d bipedal robot marlo. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2529–2536. IEEE, 2014. 9

[35] Tad McGeer et al. Passive dynamic walking. *The International Journal of Robotics Research (IJRR)*, 9(2):62–82, 1990. 9, 12

[36] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Compliant leg behaviour explains basic dynamics of walking and running. *Proceedings of the Royal Society of London B: Biological Sciences*, 273(1603):2861–2867, 2006. 10

[37] Zachary Batts, Seungmoon Song, and Hartmut Geyer. Toward a virtual neuromuscular control for robust walking in bipedal robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6318–6323. IEEE, 2015. 10

[38] John Nassour, Patrick Hénaff, Fethi Benouezdou, and Gordon Cheng. Multi-layered multi-pattern cpg for adaptive locomotion of humanoid robots. *Biological cybernetics*, 108(3):291–303, 2014. 10

[39] Ludovic Righetti and Auke Jan Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1585–1590. IEEE, 2006. 10

[40] Russ Tedrake, Teresa Weirui Zhang, and H Sebastian Seung. Learning to walk in 20 minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, volume 95585, pages 1939–1412. Beijing, 2005. 10, 12

[41] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2016)*, 35(4), 2016. 10, 14

[42] Arjun Sharma and Kris M Kitani. Phase-parametric policies for reinforcement learning in cyclic environments. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 10

[43] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2817–2826. JMLR. org, 2017. 10

[44] Yoshiaki Sakagami, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo Fujimura. The intelligent asimo: System overview and integration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2478–2483. IEEE, 2002. 11, 74

[45] Linda Geppert. Qrio, the robot that could. *IEEE Spectrum*, 41(5):34–37, 2004. 11

[46] Kenji Kaneko, Kensuke Harada, Fumio Kanehiro, Go Miyamori, and Kazuhiko Akachi. Humanoid robot hrp-3. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2471–2478. IEEE, 2008. 11, 74

[47] Yoshihiro Kusuda. Toyota's violin-playing robot. *Industrial Robot: An International Journal*, 35(6):504–506, 2008. 11

[48] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1620–1626. IEEE, 2003. 11

[49] Steven H Collins, Martijn Wisse, and Andy Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research (IJRR)*, 20(7):607–615, 2001. 11, 12

[50] Jun Ho Choi. *Model-based control and analysis of anthropomorphic walking*. University of Michigan, 2005. 11

[51] Jessy W Grizzle, Christine Chevallereau, Ryan W Sinnet, and Aaron D Ames. Models, feedback control, and open problems of 3D bipedal robotic walking. *Automatica*, 50(8):1955–1988, 2014. 11, 12, 13, 74

[52] Mark W Spong and Francesco Bullo. Controlled symmetries and passive walking. *IEEE Transactions on Automatic Control*, 50(7):1025–1031, 2005. 12

[53] Tomomichi Sugihara and Yoshihiko Nakamura. Whole-body cooperative balancing of humanoid robot using cog jacobian. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2575–2580. IEEE, 2002. 12

[54] Don Joven Agravante, Giovanni Claudio, Fabien Spindler, and François Chaumette. Visual servoing in an optimization framework for the whole-body control of humanoid robots. *IEEE Robotics and Automation Letters*, 2(2):608–615, 2016. 12

[55] Jessy W Grizzle, Christine Chevallereau, Aaron D Ames, and Ryan W Sinnet. 3d bipedal robotic walking: models, feedback control, and open problems. *IFAC Proceedings Volumes*, 43(14):505–532, 2010. 13

[56] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and Jessy W Grizzle. Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on mabel. *The International Journal of Robotics Research (IJRR)*, 32(3):324–345, 2013. 13

[57] Quan Nguyen, Ayush Agrawal, Xingye Da, William C Martin, Hartmut Geyer, Jessy W Grizzle, and Koushil Sreenath. Dynamic walking on randomly-varying discrete terrain with one-step preview. In *Robotics: Science and Systems*, 2017. 13

[58] Aaron D Ames. Human-inspired control of bipedal walking robots. *IEEE Transactions on Automatic Control (TAC)*, 59(5):1115–1130, 2014. 13, 69

[59] Quan Nguyen and Koushil Sreenath. 3d dynamic walking on stepping stones with control barrier functions. In *2016 IEEE 55th Conference on Decision and Control*, Submitted, 2016. 13

[60] Shishir Nadubettu Yadukumar, Murali Pasupuleti, and Aaron D Ames. Human-inspired underactuated bipedal robotic walking with amber on flat-ground, up-slope and uneven terrain. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2478–2483. IEEE, 2012. 13

[61] Sebastian E Sovero, Cenk Oguz Saglam, and Katie Byl. Passive Frontal Plane Stabilization in 3D Walking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. 13

[62] Robert D Gregg and Mark W Spong. Reduction-based control of three-dimensional bipedal walking robots. *The International Journal of Robotics Research*, 2009. 13

[63] R. Gregg, A. Tilton, S. Candido, T. Bretl, and M. Spong. Control and planning of 3d dynamic walking with asymptotically stable gait primitives. *IEEE Transactions on Robotics*, 28(6):1415–1423, 2012. 13

[64] Brian G Buss, Kaveh Akbari Hamed, Brent A Griffin, and Jessy W Grizzle. Experimental results for 3d bipedal robot walking based on systematic optimization of virtual constraints. In *American Control Conference (ACC), 2016*, pages 4785–4792. IEEE, 2016. 13

[65] Ching-Long Shih, JW Grizzle, and Christine Chevallereau. From stable walking to steering of a 3d bipedal robot with passive point feet. *Robotica*, 30(7):1119–1130, 2012. 13

[66] Taeyoung Lee, Koushil Sreenath, and Vijay Kumar. Geometric control of cooperating multiple quadrotor uavs with a suspended payload. In *IEEE Conference on Decision and Control (CDC)*, pages 5510–5515, 2013. 13, 29

[67] Koushil Sreenath and Vijay Kumar. Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots. *Robotics: Science and Systems (R:SS)*, 2013. 13, 29

[68] A. K. Sanyal, A. Fosbury, N. A. Chaturvedi, and D. S. Bernstein. Inertia-free spacecraft attitude trajectory tracking with disturbance rejection and almost global stabilization. *AIAA Journal of Guidance, Control and Dynamics*, 32(2):1167–1178, Feb 2009. 13, 56

[69] A. K. Sanyal and N. A. Chaturvedi. Almost global robust attitude tracking control of spacecraft in gravity. In *AIAA Guidance, Navigation and Control Conference*, pages AIAA–2008–6979, Aug 2008, Honolulu. 13, 56

[70] S P Viswanathan, A K Sanyal, F Leve, and N H McClamroch. Dynamics and control of spacecraft with a generalized model of variable speed control moment gyroscopes. *Journal of Dynamic Systems, Measurement, and Control*, 137(7):071003, 2015. 13, 49

[71] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. *Global Formulations of Lagrangian and Hamiltonian Dynamics on Manifolds*. Springer, 2017. 13, 31

[72] Tse-Huai Wu. *Spacecraft Relative Attitude Formation Tracking on SO (3) Based on Line-of-Sight Measurements*. PhD thesis, The George Washington University, 2013. 13

[73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 14

[74] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4733–4742, 2016. 14

[75] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015. 14

[76] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4346–4354, 2015. 14

[77] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 14

[78] Mark Fuge, Mehmet Ersin Yumer, Gunay Orbay, and Levent Burak Kara. Conceptual design and modification of freeform surfaces using dual shape representations in augmented reality environments. *Computer-Aided Design*, 44(10):1020–1032, 2012. 14

[79] Mehmet Ersin Yumer, Paul Asente, Radomir Mech, and Levent Burak Kara. Procedural modeling using autoencoder networks. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 109–118. ACM, 2015. 14

[80] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. 14

[81] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real singel-image flight without a singel real image. 2017. 14, 15

[82] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 14

[83] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015. 14, 16

[84] Martim Brandao, Yukitoshi Minami Shiguematsu, Kenji Hashimoto, and Atsuo Takanishi. Material recognition cnns and hierarchical planning for biped robot locomotion on slippery terrain. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 81–88. IEEE, 2016. 14

[85] John M Hsu and Steven C Peters. Extending open dynamics engine for the darpa virtual robotics challenge. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 37–48. Springer, 2014. 14

[86] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033. IEEE, 2012. 14

[87] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 14

[88] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Aerial Informatics and Robotics platform. Technical report, Microsoft Research, 2017. 14

[89] Udacity. Self-driving car simulator, 2016. 14

[90] Judy Hoffman, Sergio Guadarrama, Eric S Tzeng, Ronghang Hu, Jeff Donahue, Ross Girshick, Trevor Darrell, and Kate Saenko. Lsda: Large scale detection through adaptation. In *Advances in Neural Information Processing Systems*, pages 3536–3544, 2014. 15

[91] Brian Kulis, Kate Saenko, and Trevor Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1785–1792. IEEE, 2011. 15

[92] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7167–7176, 2017. 15

[93] Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016. 15

[94] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, pages 102–118. Springer, 2016. 15

[95] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, and Ram Vasudevan. Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks? 2016. 15

[96] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017. 15, 90, 95

[97] Joel Chestnutt, Yutaka Takaoka, Keisuke Suga, Koichi Nishiwaki, James Kuffner, and Satoshi Kagami. Biped navigation in rough environments using on-board sensing. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009. 15

[98] Maurice F. Fallon, Pat Marion, Robin Deits, Thomas Whelan, Matthew Antone, John McDonald, and Russ Tedrake. Continuous humanoid locomotion over uneven terrain using stereo fusion. *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015. 15

[99] Chris Urmson, Drew Bagnell, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 2008. 16

[100] Max Bajracharya, Jeremy Ma, Matt Malchano, Alex Perkins, Alfred A Rizzi, and Larry Matthies. High fidelity day/night stereo mapping with vegetation and negative obstacle detection for vision-in-the-loop walking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013. 16

[101] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014. 16

[102] Aftab E Patla. Understanding the roles of vision in the control of human locomotion. *Gait & Posture*, 5(1):54–69, 1997. 16

[103] Jonathan Samir Matthis and Brett R Fajen. Humans exploit the biomechanics of bipedal gait during visually guided walking over complex terrain. *Proceedings of the Royal Society of London B: Biological Sciences*, 280(1762):20130700, 2013. 16

[104] Jonathan Samir Matthis, Sean L Barton, and Brett R Fajen. The critical phase for visual control of human walking over complex terrain. *Proceedings of the National Academy of Sciences*, page 201611699, 2017. 16

[105] Alireza Ramezani, Jonathan W. Hurst, Kaveh Akbari Hamed, and J. W. Grizzle. Performance Analysis and Feedback Control of ATRIAS, A Three-Dimensional Bipedal Robot. *Journal of Dynamic Systems, Measurement, and Control*, 136(2), 2014. 19

[106] E. Westervelt, J. W. Grizzle, C. Chevallereau, J. O. Choi, and B. Morris. *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, Boca Raton, FL, 2007. 19, 20

[107] Mikhail S Jones. Optimal control of an underactuated bipedal robot. Master's thesis, Oregon State University, ScholarsArchive@OSU, 2014. 20

[108] Ayonga Hereid, Christian M Hubicki, Eric A Cousineau, Jonathan W Hurst, and Aaron D Ames. Hybrid zero dynamics based multiple shooting optimization with applications to robotic walking. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5734–5740. IEEE, 2015. 20

[109] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control (TAC)*, 59(4):876–891, April 2014. 21

[110] Quan Nguyen, Xingye Da, J. W. Grizzle, and Koushil Sreenath. Dynamic walking on stepping stones with gait library and control barrier. *Workshop on Algorithimic Foundations of Robotics*, 2016. 21, 24

[111] Wikipedia. 23

[112] Ayonga Hereid, Omar Harib, Ross Hartley, Yukai Gong, and Jessy W Grizzle. Rapid bipedal gait design using c-frost with illustration on a cassie-series robot. *arXiv preprint arXiv:1807.06614*, 2018. 25, 79

[113] J W. Grizzle, C Chevallereau, A Ames, and R Sinnet. 3d bipedal robotic walking: Models, feedback control, and open problems. In *IFAC Symposium on Nonlinear Control Systems*, Bologna, Italy, September 2010. 26, 79

[114] Kenji Hashimoto et al. Realization by biped leg-wheeled robot of biped walking and wheel-driven locomotion. In *Proceedings of the 2005 IEEE Int*. IEEE International Conference on Robotics and Automation (ICRA), 2005. 26

[115] Sonia Waharte and Niki Trigoni. Supporting search and rescue operations with uavs. In *IEEE International Conference on Emerging Security Technologies*. IEEE, 2010. 27

[116] Debra Lewis, T Ratiu, JC Simo, and Jerrold E Marsden. The heavy top: a geometric treatment. *Nonlinearity*, 5(1):1, 1992. 29

[117] Nalin A Chaturvedi, Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Nonlinear dynamics of the 3d pendulum. *Journal of Nonlinear Science*, 21(1):3–32, 2011. 29, 31

[118] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Stable manifolds of saddle equilibria for pendulum dynamics on s 2 and so (3). In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 3915–3921. IEEE, 2011. 29

[119] Chung Choo Chung and John Hauser. Nonlinear control of a swinging pendulum. *Automatica*, 31(6):851–862, 1995. 29

[120] Karl Johan Åström and Katsuhisa Furuta. Swinging up a pendulum by energy control. *Automatica*, 36(2):287–295, 2000. 29

[121] AS Shiriaev, H Ludvigsen, and O Egeland. Swinging up of the spherical pendulum. *IFAC Proceedings Volumes*, 32(2):2193–2198, 1999. 29

[122] Simon Lefrançois and Clément Gosselin. Point-to-point motion control of a pendulum-like 3-dof underactuated cable-driven robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5187–5193. IEEE, 2010. 29

[123] Daniel Cunningham and H Harry Asada. The winch-bot: A cable-suspended, underactuated robot utilizing parametric self-excitation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1844–1850. IEEE, 2009. 29

[124] Damiano Zanotto, Giulio Rosati, and Sunil K Agrawal. Modeling and control of a 3-dof pendulum-like manipulator. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3964–3969. IEEE, 2011. 29

[125] Christine Chevallereau, Hamed Razavi, Damien Six, Yannick Aoustin, and Jessy Grizzle. Self-synchronization and self-stabilization of 3d bipedal walking gaits. *Robotics and Autonomous Systems*, 100:43–60, 2018. 29

[126] Ioannis Poulakakis and Jessy W Grizzle. The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. *IEEE Transactions on Automatic Control*, 54(8):1779–1793, 2009. 29

[127] Fuminori Saito, Toshio Fukuda, and Fumihito Arai. Swing and locomotion control for a two-link brachiation robot. *IEEE Control Systems Magazine (CSM)*, 14(1):5–12, 1994. 29

[128] Siavash Farzan, Ai-Ping Hu, Evan Davies, and Jonathan Rogers. Feedback motion planning and control of brachiating robots traversing flexible cables. In *American Control Conference (ACC)*, pages 1323–1329. IEEE, 2019. 29

[129] Guangyu Liu, Dragan Nešić, and Iven Mareels. Modelling and stabilisation of a spherical inverted pendulum. *IFAC Proceedings Volumes*, 38(1):1130–1135, 2005. 29

[130] Carlos Aguilar-Ibanez, F Octavio Gutierrez, and Humberto Sossa-Azuela. Lyapunov approach for the stabilization of the inverted spherical pendulum. In *IEEE Conference on Decision and Control (CDC)*, pages 6133–6137. IEEE, 2006. 29

[131] Brian Bittner and Koushil Sreenath. Symbolic computation of dynamics on smooth manifolds. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2016. 29

[132] Nalin Chaturvedi, Amit K Sanyal, N Harris McClamroch, et al. Rigid-body attitude control. *Control Systems, IEEE*, 31(3):30–51, 2011. 29

[133] Taeyoung Lee. Geometric tracking control of the attitude dynamics of a rigid body on so(3). In *American Control Conference (ACC)*, pages 1200–1205. IEEE, 2011. 29, 35, 69

[134] Taeyoung Lee. Geometric control of quadrotor uavs transporting a cable-suspended rigid body. *IEEE Transactions on Control Systems Technology*, 2017. 29

[135] Mark Wilfried Mueller. Multicopter attitude control for recovery from large disturbances. *arXiv preprint arXiv:1802.09143*, 2018. 29, 39

[136] Taeyoung Lee. Exponential stability of an attitude tracking control system on so (3) for large-angle rotational maneuvers. *Systems & Control Letters*, 61:231–237, 2012. 35, 69

[137] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *IEEE Conference on Decision and Control (CDC)*, pages 5420–5425. IEEE, 2010. 36, 69

[138] Y. Hürmüzlü and T.H. Chang. Rigid body collisions of a special class of planar kinematic chains. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):964–71, 1992. 46

[139] A Iserles, H Z Munthe-Kaas, S P Nørsett, and A Zanna. Lie-group methods. *Acta Numerica 2000*, 9:215–365, 2000. 47

[140] M Kobilarov, K Crane, and M Desbrun. Lie group integrators for animation and control of vehicles. *ACM Transactions on Graphics (TOG)*, 28(2):16, 2009. 47

[141] J E Marsden and M West. Discrete mechanics and variational integrators. *Acta Numerica 2001*, 10:357–514, 2001. 47

[142] M West. *Variational integrators*. PhD thesis, California Institute of Technology, 2004. 47

[143] T Lee, N H McClamroch, and M Leok. A lie group variational integrator for the attitude dynamics of a rigid body with applications to the 3d pendulum. In *IEEE Conference on Control Applications (CCA)*, pages 962–967. IEEE, 2005. 47, 50

[144] T Lee. *Computational geometric mechanics and control of rigid bodies*. ProQuest, 2008. 47, 50

[145] A. K. Sanyal and A. Goswami. Dynamics and balance control of the reaction mass pendulum (rmp): A 3d multibody pendulum with variable body inertia. *ASME Journal of Dynamic Systems, Measurement and Control*, 136(2):paper 021002, 2014. 53, 58

[146] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, Englewood Cliffs, NJ, 3 edition, 2002. 53, 55

[147] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch. Rigid-Body Attitude Control: Using rotation matrices for continuous, singularity-free control laws. *IEEE Control Systems Magazine*, 31(3):30–51, June 2011. 56

[148] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980. 59

[149] Lawrence F Shampine and Mark W Reichelt. The matlab ode suite. *SIAM journal on scientific computing*, 18(1):1–22, 1997. 59

[150] Yukai Gong et al. Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway. *American Control Conference (ACC)*, 2019, 2019. 62

[151] Agility robotics cassie documentation and software release repository. 69, 71

[152] Yanran Ding, Abhishek Pandala, and Hae-Won Park. Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *International Conference on Robotics and Automation (ICRA)*, pages 8484–8490. IEEE, 2019. 74

[153] Michael Watterson, Trey Smith, and Vijay Kumar. Smooth trajectory generation on se (3) for a free flying space robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5459–5466. IEEE, 2016. 74

[154] Michael Watterson, Sikang Liu, Ke Sun, Trey Smith, and Vijay Kumar. Trajectory optimization on manifolds with applications to quadrotor systems. *The International Journal of Robotics Research*, 2020. 74

[155] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. 74

[156] Riccardo Bonalli, Andrew Bylard, Abhishek Cauligi, Thomas Lew, and Marco Pavone. Trajectory optimization on manifolds: A theoretically-guaranteed embedded sequential convex programming approach. *arXiv preprint arXiv:1905.07654*, 2019. 74

[157] G Clark Haynes and Alfred A Rizzi. Gaits and gait transitions for legged robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1117–1122, 2006. 78

[158] Christine Chevallereau, Jessy W Grizzle, and Ching-Long Shih. Asymptotically stable walking of a five-link underactuated 3-d bipedal robot. *IEEE transactions on robotics*, 25(1):37–50, 2009. 78

[159] Wen-Loong Ma, Ayonga Hereid, Christian M Hubicki, and Aaron D Ames. Efficient HZD gait generation for three-dimensional underactuated humanoid running. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5819–5825, 2016. 79

[160] Xingye Da, Omar Harib, Ross Hartley, Brent Griffin, and Jessy W Grizzle. From 2d design of underactuated bipedal gaits to 3d implementation: Walking with speed tracking. *IEEE Access*, 4:3469–3478, 2016. 79

[161] 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 *American Control Conference (ACC)*, pages 4559–4566, 2019. 79, 81, 83

[162] Ross Hartley, Xingye Da, and Jessy W Grizzle. Stabilization of 3d underactuated biped robots: Using posture adjustment and gait libraries to reject velocity disturbances. In *Control Technology and Applications (CCTA), 2017 IEEE Conference on*, pages 1262–1269. IEEE, 2017. 79

[163] Quan Nguyen, Xingye Da, J. W. Grizzle, and Koushil Sreenath. Dynamic walking on stepping stones with gait library and control barrier. In *Workshop on Algorithimic Foundations of Robotics (WAFR)*, 2016. 79, 81

[164] Quan Nguyen, Ayonga Hereid, Jessy W Grizzle, Aaron D Ames, and Koushil Sreenath. 3d dynamic walking on stepping stones with control barrier functions. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 827–834. IEEE, 2016. 79, 81

[165] Ayonga Hereid and Aaron D Ames. Frost*: Fast robot optimization and simulation toolkit. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 719–726, 2017. 79

[166] Xingye Da, Ross Hartley, and Jessy W Grizzle. Supervised learning for stabilizing underactuated bipedal robot locomotion, with outdoor experiments on the wave field. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3476–3483, 2017. 79, 80, 81

[167] 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. *International Journal of Robotics Research (IJRR)*, 38(9):1063–1097, 2019. 79

[168] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, pages 3835–3844, 2018. 82

[169] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *arXiv preprint arXiv:1906.04893*, 2019. 82

[170] Lei Le, Andrew Patterson, and Martha White. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. In *Advances in Neural Information Processing Systems*, pages 107–117, 2018. 82

[171] Lance Flavell. Beginning blender: open source 3d modeling, animation, and game design. *The expert's voice in open source*, 2010. 86

[172] Stereo Labs. Zed stereo camera. 88, 95

[173] François Chollet. Keras, 2015. 89

[174] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016. 89

[175] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019. 99

[176] Guillermo A Castillo, Bowen Weng, Ayonga Hereid, Zheng Wang, and Wei Zhang. Reinforcement learning meets hybrid zero dynamics: A case study for rabbit. In *International Conference on Robotics and Automation (ICRA)*, pages 284–290, 2019. 99

[177] 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. *arXiv preprint arXiv:1910.01748*, 2019. 99