Contact-Aided State Estimation on Lie Groups for Legged Robot Mapping and Control

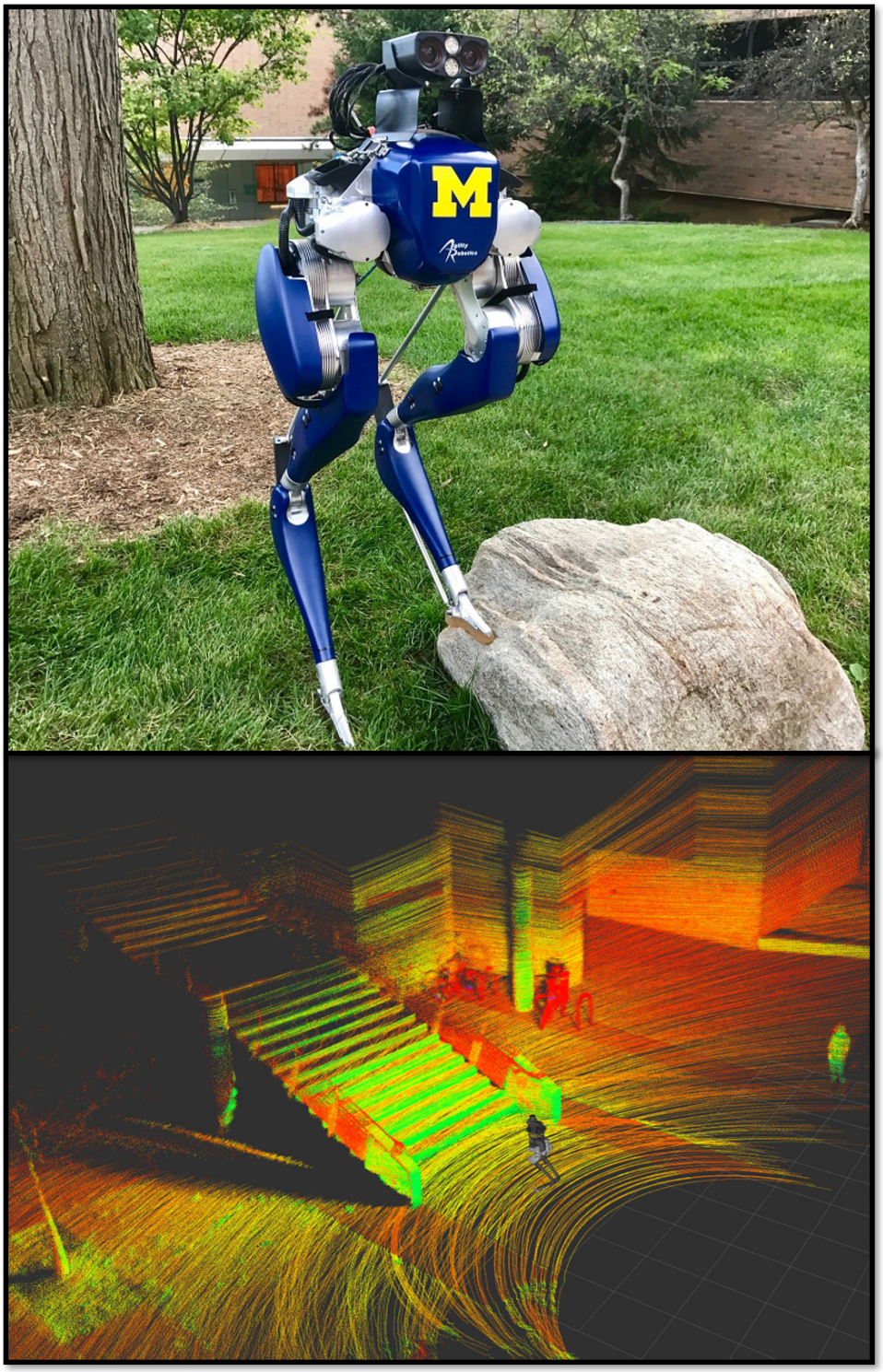
by

Matthew Ross Hartley

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in the University of Michigan
2019

Doctoral Committee:

Professor Jessy W. Grizzle, Co-Chair
Assistant Research Scientist Maani Ghaffari Jadidi, Co-Chair
Assistant Professor Dmitry Berenson
Professor Ryan M. Eustice
Assistant Professor Ram Vasudevan



Matthew Ross Hartley

rosshart@umich.edu

ORCID iD: 0000-0002-4605-4333

DEDICATION

For my wife, Kimberly, and all of my family. Without your support, none of this work would have been possible. Also to my beautiful daughter, Lena. It amazes me that that within one short year on Earth, you almost have MARLO and Cassie beat on walking.

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Jessy Grizzle, for allowing me follow my passions and do research in both controls and state estimation. I extend these thanks to Dr. Maani Ghaffari Jadidi, who guided me on the journey though Lie groups, factor graphs, and other estimation and mapping techniques. To the rest of my dissertation committee, Prof. Ryan Eustice, Prof. Dmitry Berenson and Prof. Ram Vasudevan, thank you for the helpful feedback throughout this process. I am thankful for the early members of the lab, Brian Buss, Brent Griffin, Xingye "Dennis" Da, and Omar Harib. Together we spent countless hours getting MARLO to walk, providing the motivation for most of the research presented in this thesis. I would like to acknowledge the entire Agility Robots team for both the design of Cassie and hours spent fixing her after some of our crazier experiments. I am grateful to Yukai Gong; your development of Cassie's amazing controller enabled me to focus my efforts on state estimation. None of my experimental results would have been possible without your code. To Ayonga Hereid, thank you for enduring all of my questions on trajectory optimization as well as all of the support in gait generation for both MARLO and Cassie. I would also like to acknowledge Zhenyu Gan, Eva Mungai, and Grant Gibson, for their additional support with Cassie experiments. To Cassie's perception team members, Bruce Huang, Ray Zhang, and Lu Gan, it takes a lot of software and teamwork to bring vision to legged robots. I am inspired by all of the progress we have made. Finally, I would like to give a special thanks to all of my fellow robotics program guinea pigs, Meghan Richey, Mia Stevens, Katie Skinner, and Josh Mangelson. Your friendship certainly made the long hours of course work and studying more enjoyable. Funding for this work was partially given by the Toyota Research Institute (TRI), however this thesis solely reflects the opinions and conclusions of its author and not TRI or any other Toyota entity.

Ross Hartley
May 22, 2019

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

LIST OF ABBREVIATIONS

IMU inertial measurement unit

EKF extended Kalman filter

ErEKF error-state extended Kalman filter

QEKF quaternion-based extended Kalman filter

InEKF invariant extended Kalman filter

RIEKF right invariant extended Kalman filter

INS inertial navigation systems

VIO visual-inertial odometry

VI visual-inertial

IC inertial-contact

VIC visual-inertial-contact

DOF degrees of freedom

SLAM simultaneous localization and mapping

HZD hybrid zero dynamics

PHZD partial hybrid zero dynamics

GHZD generalized hybrid zero dynamics

FROST Fast Robot Optimization and Simulation Toolkit

MAP Maximum-A-Posteriori

ICP iterative closest point

CoM center of mass

ZMP zero-moment point

RRT rapidly-exploring random tree

PRM probabilistic roadmap

LIPM linear inverted pendulum model

SLIP spring loaded inverted pendulum

UAV unmanned aerial vehicle

FPE foot placement estimator

FPI foot placement indicator

FRI foot rotation indicator

CLF control Lyapunov function

CBF control barrier function

QP quadratic program

CLF-QP control Lyapunov function based quadratic program

CBF-QP control barrier function based quadratic program

BCH Baker-Campbell-Hausdorff

PAF posture adjustment function

URDF Unified Robot Description Format

IPOPT Interior Point Optimizer

ABSTRACT

Legged robots have the potential to transform the logistics and package delivery industries, become assistants in our homes, and aide in search and rescue. Although many wheeled and flying robots have begun to hit the market, useful walking robots have yet to become a practical reality due to challenging issues in controller design, motion planning, and state estimation. These challenges arise from high degrees of freedom, underactuation, and complex dynamics along with the unstructured nature of the environment in which we want these robots to operate. In particular, state estimation is a crucial component of any mobile robot system. To maintain stability, walking robots often require knowledge of orientation, velocity, joint angles, and local terrain information. Whereas, to plan and execute walking paths, awareness of global pose and map information is needed. Estimation of these states requires consistent fusion of measurements from a variety of sensors.

This thesis focuses on contact-aided state estimation techniques for legged robots. Inertial navigation systems can be used to obtain estimates of pose and velocity by integrating measurements from an inertial measurement unit. However, due to sensor noise and bias, these estimates will quickly drift away from their true values. To reduce or eliminate this drift, additional sensors, such as magnetometers and GPS, can be used to aid these inertial measurements. For legged robots, foot contact and forward kinematic measurements will perform this correction.

First, we develop a contact-aided invariant extended Kalman filter (InEKF) using the theory of Lie groups and invariant observer design. After modeling the robot's state on a Lie group, we show that the error dynamics follows a log-linear autonomous differential equation allowing the observable state variables to be rendered convergent with a domain of attraction that is independent of the system's trajectory. Unlike the standard EKF, neither the linearized error dynamics nor the linearized observation model depend on the current state estimate, which leads to improved convergence properties and a local observability matrix that is consistent with the underlying nonlinear system. Although the robot's global position and yaw remain unobservable after fusion of inertial, kinematic, and contact data, this filter can be executed at high frequencies to provide the feedback controller with real-time orientation and velocity data. Furthermore, since the position/yaw drift is slow, the

pose estimate can be used to construct local terrain maps with the help of LiDAR sensors.

Next, we propose a method for contact-aided smoothing using factor graphs, which provide a flexible framework for fusing measurements from multiple sensors to obtain a maximum a posteriori estimate of the robot's entire trajectory. To extend this framework for legged robots, we developed two novel factors. The hybrid contact factor describes how a contact frame moves over time by preintegrating high-frequency inertial-contact data through an arbitrary number of contact switches, while the forward kinematic factor relates this contact frame to the robot's base frame using noisy encoder measurements. Taken together, these factors provide an independent leg odometry measurement that can be added into existing factor graphs to improve state estimation.

Finally, we conclude with ideas and future work on how these state estimates can be used along with gait libraries to design stabilizing feedback controllers as well as global motion planning techniques. All presented algorithms were verified through simulation and experiments results on an ATRIAS- or Cassie-series biped robot.

CHAPTER 1

Introduction

1.1 Motivation

Legged robots have the ability to traverse a wide variety of terrains that are inaccessible to typical wheeled vehicles. Being able to step over obstacles allows legged robots to walk in unstructured environments that include sand, snow, mud, rocks, or even fallen trees. In fact, it is estimated that over 50% of the earth is inaccessible using only wheels [15]. In addition, having legs allows these robots to tackle terrain commonly found in factories and home environments, such as stairs and or cluttered spaces, that are a challenge for their wheeled counterparts. These capabilities allow legged robots to potentially aid in package delivery, terrain exploration, search and rescue, and disaster relief. Advances in legged robot technology can also help humans regain the ability to walk though exoskeletons [5, 106] or powered prosthetics [93, 244]. Despite their amazing capabilities and potential, most legged robots are confined to controlled research spaces and have yet to be widely deployed. This is largely due to unsolved challenges in controller design, motion planning, and state estimation that arise from high degrees of freedom, underactuation, and complex dynamics along with the unstructured nature of the environment in which we want these robots to operate.

This research was motivated with a simple question: how do we design a system that allows a bipedal robot to move from point A to point B? Although the task appears simple, at least three difficult sub-problems need to be solved; state estimation and motion planning, and feedback control design. That is, the robot needs to know where it is, what path it should follow, and how to move its joints to get there without falling down. All of the research presented in this thesis aims towards building the fundamental algorithms that will enable legged robots to complete this task.

Chronologically, the research with control design on the bipedal robot MARLO[1]. First, a dynamic model of the robot is generated using the Euler-Lagrange equations. Then,

[1]See Sections 3.1.2 and 3.1.3 for a detailed description of the robots used in this thesis.

trajectory optimization can be used to find a dynamically feasible gait. The resulting open-loop trajectory corresponds to a single desired behavior of the robot (such as walking in place). A set of outputs can be chosen and stabilized using a variety of methods, such as input-output linearization [102] or simple PD control [91]. Unfortunately, since many walking robots (MARLO and Cassie included) are underactuated, not all of the robot's states can be stabilized with this style of control. In addition, for 3D walking, the dynamics of the uncontrolled states are often unstable. This led to Griffin and Grizzle's [96] work on nonholonomic virtual constraints, and the use of a heuristic foot placement controllers by Da et al. [59]. In particular, these foot placement strategies proved to be time-consuming to tune, with no easy generalization to other legged robots. This motivated the development of posture adjustment functions (PAFs) [107], presented in Section 6.2. The PAF computes an offset to the desired output trajectories based on some deviation from the nominal periodic orbit. The parameters of the PAF are computed through an optimization problem where the robot is pertubed while walking, and the cost minimizes the deviation from the nominal orbit.

Recent trajectory optimization methods [117, 116, 118] have allowed relatively fast offline computation of gaits for high-dimensional robots. This has enabled gait libraries to be built up that cover a range of operating conditions. Da et al. [59] solved a series of planar trajectory optimization problems to generate a library of periodic, forward and backward walking gaits for MARLO. Linear interpolation of these gaits was used to choose the desired trajectory based on the robot's current velocity. This led to marginal stability, where the robot simply adapts its gait to the current speed. These gaits were then fully stabilized on the 3D robot through the aforementioned foot placement method. As presented in Section 6.2, we then expanded these ideas by utilizing the full 3D model of the robot for trajectory optimization and generating a gait library based on a grid of velocities in both the sagittal and lateral planes. These gaits were stabilized using PAFs instead of heuristically tuned foot placement. Quan Nguyen et al. [187] later showed extending the gait library to include multiple step heights can be used to tackle uneven terrain. Da and Grizzle [58] generalized these gait library methods by using supervised maching learning to extract a control policy from the set of reference trajectories. If the library contains perturbation recovery gaits (or transitioning between velocities), then a dully stabilizing controller can be learning [60], potentially eliminating the need for foot placement for PAFs. These ideas are captured by the theory of generalized hybrid zero dynamics (GHZD) [58].

These gait library methods allowed for impressing walking feats on MARLO, such as walking through the valleys of the University of Michigan's Wave Field without the use of vision [60]. As described in Section 6.3. a similar controller was transferred onto the Cassie

biped robot allowing it to walk on a variety of different surfaces, including concrete, grass, snow, and sand [91]. However, two key issues were identified through working with these controllers. First, the learned control policy depended on having an estimate of the robot's current velocity (in the body frame). At this point, velocity estimation was done using only joint encoder measurements and the assumption that the stance foot was pinned to the ground. On our robots, this "kinematic velocity" estimate was too noisy to be directly used for control (as shown in Figure 1.1). A low-pass filter had to be used causing significant delays

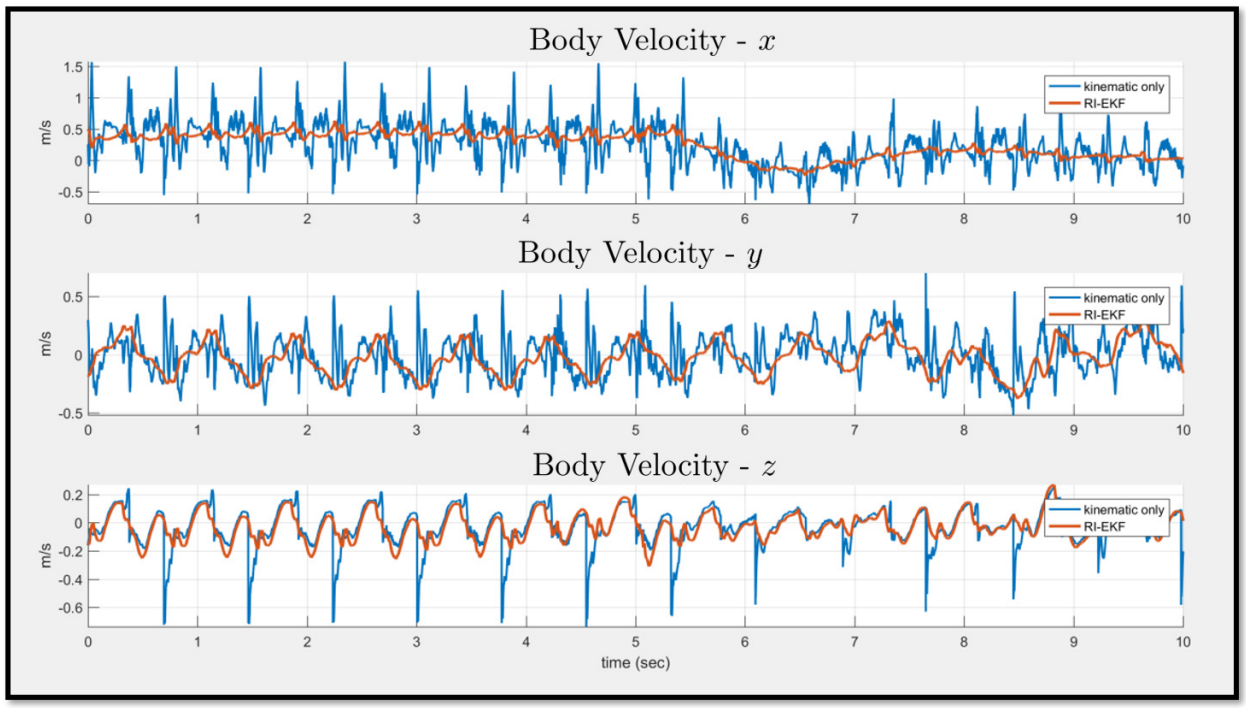


Figure 1.1: Estimate of velocity in the body frame. The using kinematics only results in a noisy signal (blue) due to sensor noise and foot slip. This motivated the development of the right invariant extended Kalman filter (RIEKF) (red) in Chapter 4.

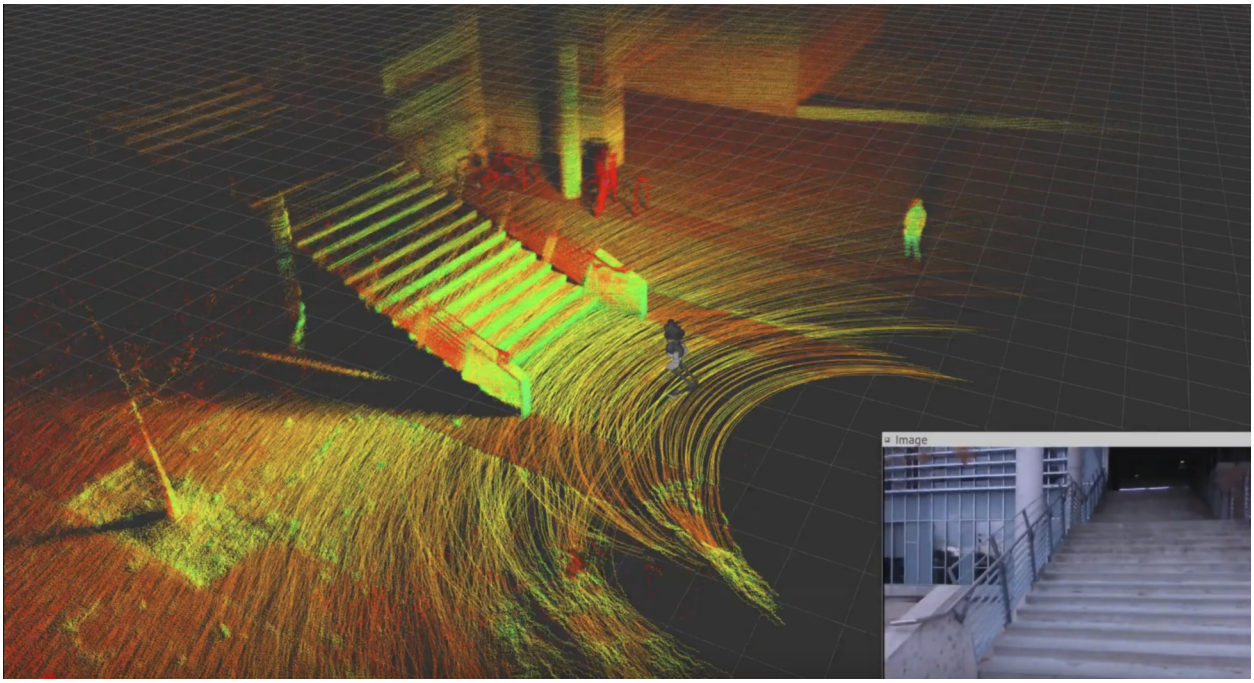
in the signal. This delay led to oscillations, further complicating the tuning of the controller parameters. Second, without a priori knowledge of the upcoming terrain, large height changes can destabilize the controller, ultimately causing the robot to fall. Walking "blind" across the large drop shown, shown in Figure 1.2, is what prevented MARLO from conquering the Wave Field. In addition, without global map information, no level of autonomy could be instilled in the robot. Solving these issues was the primary motivation behind the state estimation work presented in this thesis.

State estimation involves recovering the robot's state (position, orientation, velocity, etc.) from a collection of noisy sensor measurements. Accurate state estimates are key to both building a map of the environment and to designing a stabilizing feedback controller for walking. Bloesch et al. [34] developed an extended Kalman filter (EKF) suitable for legged

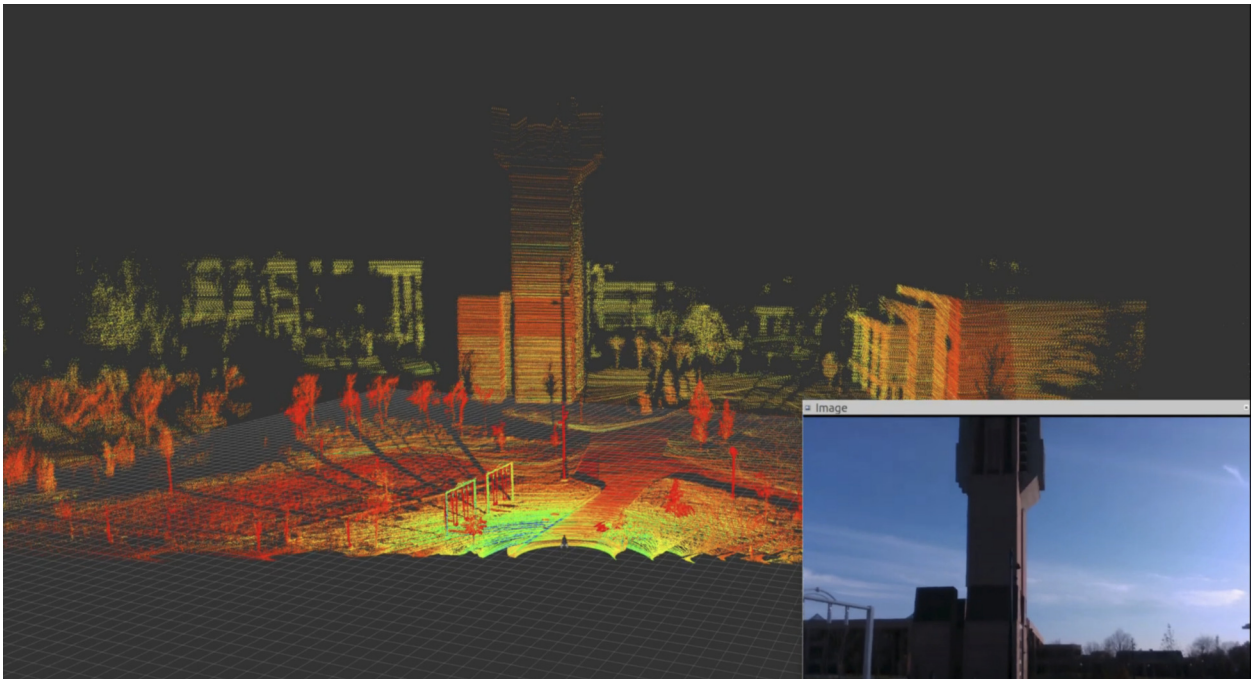
Figure 1.2: MARLO walking on the Wave Field. Without prior knowledge of the upcoming terrain, MARLO cannot adapt its gait to the large drop causing the robot to fall.

robots that combined inertial, contact, and kinematic data to estimate the robot's pose and velocity. Due to the nonlinear nature of the robot's dynamics, EKFs use the Jacobian linearization along the current best estimate of its trajectory to linearize both the dynamics and measurement models. Therefore, if the current state estimate is wrong, the linearization will poorly represent the true dynamics. This often leads to consistency issues, poor state estimates, and potentially filter divergence. In Chapter 4, we use recent advancements in invariant observer design [24] to develop an invariant extended Kalman filter (InEKF), which improves over the state-of-the-art quaternion-based extended Kalman filter (QEKF). The main advantage of our proposed filter is that the linearization no longer depends on the state estimate, leading to improved convergence properties. This filter can be used to provide the feedback controller with real-time velocity estimates and local terrain information (provided the robot is equipped with a stereo camera or LiDAR sensor). An example of this is shown in Figure 1.3

In order to obtain a global map estimate, we turned our attention to the slew of progress in visual-inertial odometry (VIO) and solving the simultaneous localization and mapping (SLAM) problem [67, 136, 137, 160, 154, 85, 75]. Although these techniques provided the underlying frameworks, there was no clear way to take advantage of the joint encoder and contact sensors that are often available on legged robots. This led the development of the contact-aided state smoothing algorithms presented in Chapter 5.



(a) Looking towards a staircase



(b) University of Michigan's North Campus with the bell tower

Figure 1.3: Once accurate pose information is obtained, LiDAR points can be used to build up a map of the environment. These maps could potentially be used to improve stability by informing the robot about upcoming terrain.

1.2 Objectives

The primary objectives of this research were to improve legged robot state estimation algorithms for use in feedback control, mapping, motion planning, and general autonomy.

Based on these objectives, the state estimation problem can be split into two goals: odometry and local terrain mapping, and global pose and map estimation. This separation stems from the following insight that holds for even human walking: only local motion and terrain estimates are needed to maintain stability whereas global pose and map information is only needed higher-level path planning. When we walk across town, we only need to look at the ground to prevent us from tripping and falling, and we only need our current location and Google Maps to plan the route.

To design a stabilizing feedback controller for a legged robot, estimates of orientation, velocity, and joint angles, local terrain are often needed. To estimate these states, we developed a *contact-aided invariant extended Kalman filter* (InEKF) using the theory of Lie groups and invariant observer design [24]. This filter combines inertial measurements from an inertial measurement unit (IMU) with contact and kinematic measurements to estimate the robot's base pose and velocity along with all current contact points. In the absence of IMU bias, we show that the error dynamics of the inertial-contact system follow a log-linear differential equation allowing the observable state variables to be rendered convergent with a domain of attraction that is independent of the system's trajectory. Unlike the standard extended Kalman filter (EKF), neither the linearized error dynamics nor the linearized observation model depend on the current state estimate, which leads to improved convergence properties and a local observability matrix that is consistent with the underlying nonlinear system [109]. We show how to augment the state with IMU biases, and show that even though some important theoretical properties are lost, the developed invariant extended Kalman filter (InEKF) still outperforms similar quaternion-based extended Kalman filter (QEKF). This state estimator can be run at high-frequencies (> 2000 Hz) to provide the feedback controller with real-time state information. Although the robot's global position and yaw remain unobservable after fusion of inertial, kinematic, and contact data, this filter can be executed at high frequencies to provide the feedback controller with real-time orientation and velocity data. Furthermore, since the position/yaw drift is slow, we show that the pose estimate from this filter can be used to construct local terrain maps with the help of stereo camera or LiDAR sensors. These ideas on contact-aided filtering are presented in Chapter 4.

In order to allow for high-level path planning, a global pose and map estimate are needed. It is possible to can extend this InEKF to include estimated landmark positions, which allows for corrections of the position and yaw states. However, just like in EKF-simultaneous local-

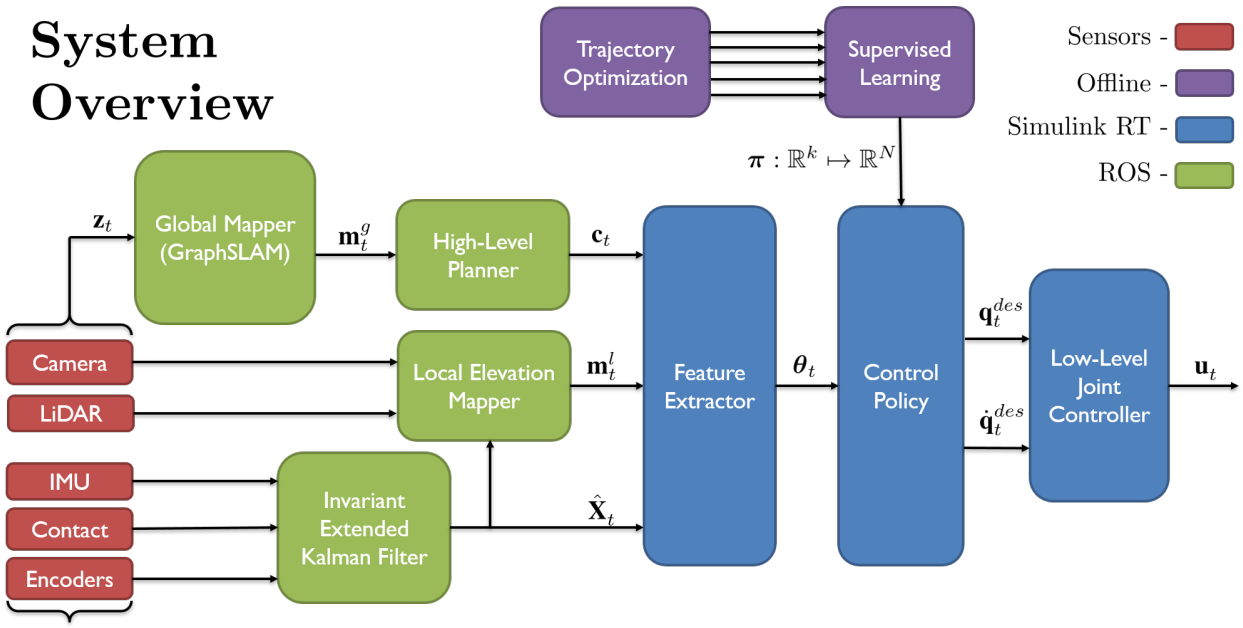


Figure 1.4: Block diagram of Cassie's proposed system architecture. The invariant extended Kalman filter (InEKF) is described in Chapter 4. The global mapper utilizes factor graph based optimization and is described in Chapter 5. Ideas on control and planning are provided in Chapters 6 and 7.

ization and mapping (SLAM), as the number of landmarks increases, the problem can become intractable to solve in real-time. In the simultaneous localization and mapping (SLAM) community, this issue typically mitigated using factor graphs and incremental smoothing techniques [66, 137]. However, the state-of-the-art offers no method for including contact and kinematic measurements within the factor graph framework. Therefore, we developed two novel factors that can be implemented alongside existing factors (such as inertial, GPS, relative pose, or visual factors) to improve Maximum-A-Posteriori (MAP) state estimation. The *forward kinematic factor* uses noisy encoder measurements to relate the robot's base frame to a contact frame located at the intersection of the robot with the ground. The *hybrid contact factors* preintegrate high-frequency contact measurements through an arbitrary number of contact switches to constrain the relative motion of the contact frame. Together, these factors combine to effectively allow "leg odometry" to be included within an existing factor graph framework. This paves the way for long-term global mapping, which is necessary for high-level path planning and general autonomy. These ideas on contact-aided smoothing are presented in Chapter 5.

In order to verify and demonstrate the capabilities of the proposed algorithms, we implement them on a the Cassie-series bipedal robot (Described in Section 3.1.3). This robot is equipped with an IMU, joint encoders, stereo cameras, LiDARs, and contact sensing capabilities. An overview of the proposed system architecture is shown in Figure 1.4, where

the colors indicate where processing was done on Cassie.

A long-term goal for legged robotics community is to achieve general autonomy. While the objectives described in this section do not directly solve this problem, they contribute towards the collection of underlying state estimation, planning, and controller design techniques that will ultimately allow legged robots to become a practical reality.

1.3 Contributions

In summary, this thesis makes the following contributions:

- Derivation of a continuous-time invariant extended Kalman filter (InEKF) for an IMU/contact process model with a forward kinematic measurement model. The filter can be run at high frequencies (> 2000 Hz) to provide estimates of a legged robot's pose and velocity along with all current contact points and inertial measurement unit (IMU) bias. We show that under certain conditions, the error dynamics of this system are actually log-linear, which leads to improved convergence properties and superior performance over standard quaternion-based extended Kalman filters (QEKFs).

- Alternative derivations of the invariant observer using a robo-centric state and the left-invariant error definitions along with analytical discretizations of the filter are provided.

- Development of an open-source C++ library for aided-inertial navigation using the InEKF, `https://github.com/RossHartley/invariant-ekf`, which was evaluated using walking experiments on a Cassie-series biped robot.

- A framework for including "leg odometry" into factor graph based smoothers for legged robot state estimation. This includes the development of two novel factors; the *forward kinematic* factor that relates the robot's base frame to a contact frame through noisy encoder measurements and several variants of *preintegrated contact factors* that constrains how this contact frame moves over time. These variants include a *rigid contact factor* which assumes that all 6 degrees of freedom (DOF) of the robot's foot is constrained while on the ground, a *point contact factor* which only assumes the foot's position is constrained, and an *invariant contact-inertial factor* which unifies this framework with the aforementioned InEKF. All developed contact factors can handle an arbitrary number of contact switches between nodes in the graph.

- A method for stabilizing walking on a 3D biped robot using velocity-based gait libraries and posture adjustment functions (PAFs). This method was tested experimentally by stabilizing a walking gait on an ATRIAS-series biped robot.

1.4 List of Publications

The work on stabilizing underactuated bipeds using gait libraries and posture adjustment functions (PAFs) was presented at the 1st IEEE Conference on Control Technology and Applications (CCTA) in 2017 [107]. The initial research on how to use "leg odometry" with factor graph based smoothing was presented at the 2018 IEEE InternationalConference on Robotics and Automation (ICRA) [110]. This work was then generalized to include hybrid preintegration of contact measurements through an arbitrary number of contact switches. This research was presented at the 2018 IEEE International Conference on Intelligent Robots and Systems (IROS) [108]. The work on contact-aided invariant filtering was presented at the 2018 Robotics Science and Systems (RSS) conference [109].

Journal Papers

1. **Ross Hartley**, Maani Ghaffari, Ryan M. Eustice, and Jessy W Grizzle. *Contact-Aided Invariant Extended Kalman Filtering for Robot State Estimation.* Under review, The International Journal of Robotics Research

Conference Papers

1. **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. ISBN 1509021825

2. **Ross Hartley**, Josh Mangelson, Lu Gan, Maani Ghaffari Jadidi, Jeffrey M Walls, Ryan M. Eustice, and Jessy W Grizzle. *Legged robot state-estimation through combined forward kinematic and preintegrated contact factors.* In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–8. IEEE, 2018

3. **Ross Hartley**, Maani Ghaffari Jadidi, Lu Gan, Jiunn-Kai Huang, Jessy W Grizzle, and Ryan M Eustice. *Hybrid contact preintegration for visual-inertial-contact state estima-tion using factor graphs.* In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3783–3790. IEEE, 2018.

4. **Ross Hartley**, Maani Ghaffari Jadidi, Jessy W Grizzle, and Ryan M. Eustice. *Contact-Aided Invariant Extended Kalman Filtering for Legged Robot State Estimation.* In Proceedings of Robotics: Science and Systems, 2018

1.5 Outline

The rest of this thesis is divided into 6 chapters that are summarized below.

Chapter 2 gives a review of the relevant literature. This includes a review of state estimation techniques for general robots, as well as techniques specific to legged robots. It also includes a review of motion planning techniques for dynamic robots and different methods for controller design.

Chapter 3 provides the background theory necessary for understanding the remainder of the thesis. This includes the mechanical description and model of the ATRIAS and Cassie-series biped robots, as well as, the theory behind the extended Kalman filter (EKF) and factor graph based smoothing. It also covers a brief background on Lie groups and Lie algebras, with an emphasis on the matrix Lie groups commonly encountered in robotics. Finally, a background on forward kinematics and the manipulator Jacobians is given.

Chapter 4 presents a novel contact-aided invariant extended Kalman filter (InEKF) which combines contact and kinematic measurements with an inertial measurement unit (IMU) based motion model to estimate a legged robot's position, orientation, velocity, and contact points. This filter is based on the theory of invariant observer design, which is briefly summarized. This contact-aided filter is developed in two forms, a world-centric form (where we are estimating the state relative the world frame), and a robot-centric form (the state is relative to the robot's body frame). The results of implementing the filter on a Cassie-series robot is given, where it is compared to the standard quaternion-based approach. LiDAR mapping results are used to provide a use case for the filter.

Chapter 5 presents a novel method for incorporating "leg odometry" into a factor graph framework. This allows information from both joint encoders and contact measurements to be combined with any existing factor to improve Maximum-A-Posteriori (MAP) estimation. It is accomplished by introducing two separate factors. First, the forward kinematics factor relates the robot's base and contact frames at any particular time step. Second, the hybrid contact factor preintegrates high-frequency contact measurements through an arbitrary number of contact switches to constrain the motion of the contact frame over consecutive time steps. The experimental results of combining these factors with inertial and relative pose factors are also given in this chapter.

Chapter 6 presents the research on control design that was conducted. It includes the

developement of posture adjustment function (PAF) to stabilize 3D walking, as well as the generation and use of gait libraries on both MARLO and Cassie.

Chapter 7 provides a discussion on future work. We present ideas on how to use terrain information in the feedback controller as well as the utilization of gait libraries for path planning. Some of these ideas have been tested in simulation, but further experimental validation is required. Thoughts on future state estimation work is also provided.

CHAPTER 2

Literature Review

2.1 Mobile Robot State Estimation

A common challenge for all mobile robots, including legged robots, is state estimation, which refers to the process of estimating a robot's state (position, orientation, velocity, joint angles, etc.) from a set of noisy sensor measurements. An accurate, real-time state estimate is a critical foundation for the design of stabilizing feedback controllers and is often assumed for many mapping and planning algorithms. If a prior map is given, the state estimation problem is often referred to as *localization* [219]. In contrast, *mapping* refers to the process of using the robot's estimated state and exteroceptive sensor measurements to build up maps of the environment. These maps can range from simple sets of landmarks to dense point cloud representations [77, 127, 82, 115]. For mobile robots, we are often interested in solving the *simultaneous localization and mapping* (SLAM) problem, where the robot needs to localize itself in the map that it is simultaneously building [73, 46]. In the full simultaneous localization and mapping (SLAM) problem, the complete robot trajectory is also recovered [66]. In practice, the robot's state can often be augmented to include these map parameters. Therefore, in this dissertation, we broadly use the term state estimation to refer to the collection of all of these problems. Throughout the history of robotics, numerous state estimation algorithms have been proposed, but most can be grouped into two broad categories, filtering and smoothing [14].

2.1.1 Filtering Methods

Filtering methods involve estimating the robot's current state (and potentially landmarks) using the set of all measurements up to the current time [14, 94, 204]. When the process model and measurements are linear and the noise is white and Gaussian, Kalman filtering [143] provides an optimal method (minimum mean squared error) for state estimation. The general process for Kalman filtering involves two phases, propagation and correction. The

state is typically represented using a Gaussian random vector, which is parameterized by a mean and a covariance. During the propagation phase, the previous state and covariance estimate are propagated forward in time using the system dynamics (alternatively known as the process model). When a measurement is obtained, the state and covariance estimate are corrected using the measurement model along with an associated measurement noise covariance.

Although the Kalman filter provides a method for optimal linear filtering, most practical mobile robots have nonlinear system dynamics, and many useful sensor models are also nonlinear. For these cases, an extended Kalman filter (EKF) can be designed, which utilizes Taylor series expansions to linearize the process and measurement models around the current state estimate [219]. Due to its low computational complexity and accurate performance, the EKF quickly became the de facto standard of nonlinear filtering for many mobile robot applications, including wheeled vehicles, drones, and legged robots [44, 199, 34]. EKFs have also been proposed to solve the SLAM problem [205]. However, since the nonlinear system is linearized about the current state estimate, the EKF is, at best, only a locally stable observer [208, 146]. The local convergence proofs are based on Lipschitz bounds of the nonlinear terms in a model, and hence "the more nonlinear a system is, the worse an EKF may perform". Importantly, if the state estimate is initialized poorly, it is possible for the filter to diverge. In addition, because an EKFs uses a system's linearization about the current estimate, states that are unobservable can spuriously be treated as observable by the filter. While this can be mitigated through the use of an observability-constrained EKF developed by Huang et al. [130], it cannot be altogether avoided.

For many systems, the *error-state* (or *indirect*) EKF offers superior performance to the standard (total state or direct) form. As the name implies, the error-state extended Kalman filter (ErEKF) is formulated using the errors, such as pose and velocity errors, as the filter variables, while the standard EKF tracks the states themselves (pose and velocity directly) [162, 197, 222, 207]. Under small noise assumptions, this leads to linear error dynamics which are then used for covariance propagation in the error-state filter. The measurement model is also rewritten with respect to these errors. Although the error dynamics are linear in the error variables, they may still depend on the current state estimate, which if initialized poorly, will degrade the performance of the filter. However, the approximate linear nature of the error dynamics may respect the linear assumptions of the original Kalman filter better than the underlying system dynamics, which can lead to improved performance [162].

Perhaps the most important feature of the ErEKF is the ability to circumvent dynamic modeling [197]. This is done by replacing the potentially complicated process model with a relatively simple inertial measurement unit (IMU) integration model (also known as *strap-*

down modeling) [220, 167, 236]. Essentially, the IMU's angular velocity and linear acceleration measurements are integrated to propagate the state estimate, while the covariance is propagated using the error dynamics. Additional (independent) sensor measurements will correct the estimated error, which can then be used to update the state estimate. Using this method, there is no longer a need to formulate complicated, platform-specific dynamics models, which may require a large number of state variables and is likely to be exceedingly nonlinear. The "strapdown" ErEKFs has proven to yield highly accurate results (even with a low-cost IMUs) and continues to form the basis of many inertial navigation systemss (INSs) [26, 153, 222, 207, 35].

In the standard formulation of Kalman filtering theory, the system evolves on Euclidean spaces. However, in many cases, the state variables we are interested in lie on a *manifold*. For example, the orientation of a 3D rigid body is represented by an element of the *special orthogonal group*, SO(3). This matrix Lie group is defined by the set of orthogonal 3×3 matrices with a determinant of one. Although the matrix contains nine variables, the dimension of the manifold is only three. One common approach is to parameterize SO(3) using local coordinates such as three Euler or Tait-Bryan angles [90, 200]. This allows the standard ErEKF equations to be applied; however, these local parameterizations are often arbitrary (and therefore confusing) and contain singularities (the well-known Gimbal lock problem). Alternatively, it is possible to represent 3D orientation using quaternions, which are a four-dimensional double cover (a two to one diffeomorphism) of SO(3). Using quaternions eliminates the singularities; however, modifications to the standard ErEKF equations have to be made [207, 222]. In brief, while the orientation is represented by a four-dimensional quaternion, the orientation error has to be defined by a $3-$vector (in the Lie algebra of SO(3)) and the associated covariance by a 3×3 matrix in order to prevent degeneracy. Also, the orientation corrections are done through quaternion multiplication instead of vector addition. This quaternion-based extended Kalman filter (QEKF) [1] has been well studied and implemented on a number of platforms, ranging from spacecraft [172, 153] to legged robots [35, 196, 81].

It turns out, many useful robot states can be characterized using matrix Lie groups. Examples include 2D orientation, SO(2), 3D orientation, SO(3), and 3D pose (orientation and position), SE(3). If the state to be estimated is a matrix Lie group, it is possible to further improve the EKF filtering approach. Bourmaud et al. [40, 41] developed versions of both discrete and continuous-time EKFs for systems where the state dynamics and measurements evolve on matrix Lie groups. In these formulations, noise is represented as a *concentrated*

[1] The quaternion-based formulation of EKFs is also sometimes called "multiplicative filtering" (MEKF) due to the orientation correction being done through quaternion multiplication [164].

Gaussian on Lie groups [229, 230], which is a generalization of the multivariate Gaussian distribution. In essence, noise is represented as a Gaussian in the tangent space about a point on the manifold. This noise is then mapped to the Lie group through the use of the group's exponential map, resulting in a decidedly non-Gaussian distribution on the manifold. An improved state estimate is obtained due to the filter taking into account the geometry and structure of the problem [41].

Most recently, a new type of EKF has been developed that is rooted in the theory of invariant observer design, in which the estimation error is invariant under the action of a Lie group [3, 38]. This invariance is referred to as the *symmetries* of the system [19]. This work led to the development of the invariant extended Kalman filter (InEKF) [37, 23, 24, 25], with successful applications and promising results in SLAM [23, 243] and aided INSs [39, 18, 19, 23, 237]. Similar to the above mentioned EKFs on matrix Lie groups, the state is again represented as a matrix Lie group and the noise as a concentrated Gaussian on the group.

However, the InEKF exploits available system symmetries to further improve filtering results. The culminating result of the InEKF states that if a system satisfies a "group-affine" property, the estimation error satisfies a "log-linear" autonomous differential equation on the Lie algebra of the corresponding Lie group [25, 23]. In other words, the system linearization does not depend upon the estimated states. Therefore, one can design a nonlinear state estimator with strong convergence properties. Surprisingly, many mobile robot state estimation problems can be solved within the InEKF framework. This includes attitude estimation [37], inertial odometry [25, 23], velocity-aided inertial navigation [39], landmark-aided navigation [23], GPS and magnetometer-aided navigation [18], and even EKF-based SLAM. In this thesis, we extend this class of solutions to contact-aided inertial navigation [109], where forward kinematics is used to correct inertial and contact-based prediction models. This successfully allows an InEKF to be used for legged robot state estimation. More details can be found in Chapter 4. This approach successfully allows an InEKF to be used for legged robot state estimation.

Countless other filtering techniques and modifications have been proposed throughout the literature to solve the mobile robot state estimation and SLAM problems. The *information filter*, often considered to be dual to the Kalman filter, is a Gaussian filter where the states are represented by an information vector and an associated information matrix as opposed to a mean and covariance [219]. The natural sparsity that arises in the information matrix can be exploited, which has allowed the extended information filter to be used to improve the efficiency of SLAM techniques [218, 79, 227, 80]. The *unscented Kalman filter* provides an alternative method for linearizing the process and measurement models

based a deterministic sampling approach [135]. When propagated through the true nonlinear systems, these carefully chosen samples can be used to recompute the posterior mean and covariance accurately up to the third order [228]. This filter has also been implemented on a range of platforms (including legged robots) to solve state estimation and SLAM problems [35, 242, 126, 57]. All of the before mentioned filters assume Gaussian distributions; however, a number of nonparametric filters have also been developed. In particular, *particle filtering* approximates the posterior distribution using a finite number of samples (or particles) [219, 71]. The nonparametric nature allows arbitrary probability distributions (even multi-model ones) to be captured. By consequence, particle filters are well adapted to solve the localization problem for many mobile robots [193], including humanoids [81].

Despite all the breakthroughs in filtering, over time, the accumulation of numerous landmarks in the state renders many filtering methods computationally intractable for long-term SLAM [69]. In addition, since only the current state is being estimated, all previous states are marginalized out. As a consequence, for nonlinear systems, the chosen linearization points are fixed which may lead to an accumulation of linearization errors. For mobile robotics, smoothing methods are popular alternatives that can potentially alleviate these issues.

2.1.2 Smoothing Methods

Smoothing methods involve estimating the robot's state at a particular time given all previous measurements and a set of future measurements [14, 43]. Perhaps the most simple example is the fixed-lag smoother [168] where a delayed state is being estimated, which allows future measurements to be incorporated. Other formulations of the fixed-lag smoother involve tracking all states within a sliding time window [70, 202], which can be solved using nonlinear optimization to obtain maximum likelihood estimates. All states outside this window are marginalized out, leading to many of the same issues that filtering approaches are plagued with (inconsistency, and accumulation of linearization errors). In general, however, smoothing methods typically outperform comparable filtering approaches since past measurements can be re-linearized as future information is obtained [165].

Full smoothers, which estimate the complete trajectory of states (and landmarks), are often employed to solve the full SLAM problem. Since old states are never marginalized out, there are no commitments to previous (and potentially bad) linearizations, as the motion and measurement models can always be re-linearized around the current best estimate. This leads to highly accurate performance; however, the unbounded growth of states makes real-time implementation a challenge. These smoothing problems are often represented using various graphical modeling schemes, including the widely used factor graph framework [67, 47, 65].

A *factor graph* is a bipartite graph that provides a convenient method for modeling the relationships between measurements and state variables. Once a factor graph is defined, inference is often performed by computing the *Maximum-A-Posteriori* (MAP) estimate, which maximizes the posteriori probability of the states (and landmarks) given the complete set of measurements [67, 99]. This can be reduced down to a weighted, nonlinear least-squares problem, which can be iteratively solved using a variety of methods, such as the gradient descent, Gauss-Newton, Levenberg-Marquardt, or Powell's Dogleg algorithms [68]. As the number of states/landmarks grows, this optimization problem becomes intractable to solve in real-time, with the computational bottlenecks being re-linearization and the factorization of the information matrix (which is necessary for solving the approximated linear system at each time step) [66]. This lead to the development of incremental solvers, such as iSAM [136] and iSAM2 [137], which allow this factorization to be updated incrementally. When combined with periodic variable re-ordering, carefully timed re-linearizations, and batch solves, factor graph based smoothing methods can yield highly accurate results that can be obtained in real-time [67, 65].

Many mobile robots contain both visual (camera, LiDAR, etc.) and inertial sensors (gyroscopes and accelerometers). The process of fusing these sensor measurements to estimate camera pose is referred to as *visual-inertial odometry* (VIO). Although there have been many proposed solutions, state-of-the-art visual-inertial odometry (VIO) systems typically use factor graph based smoothers to estimate relative motion [85, 134, 223, 186]. The flexibility of the factor graph framework also allows for pose-based loop closures or with the estimation of numerous landmarks, giving rise to full SLAM solutions [170]. Most of these implementations rely on carefully chosen *keyframes* to limit the growth of optimization variables and to enable real-time performance [154]. Inertial measurements are typically obtained at high-frequencies, and therefore, it would be computationally impractical to add new states (and optimization variables) at every time step. To limit variable growth at the rate of keyframe addition, Lupton and Sukkarieh [160] proposed a method for preintegrating IMU measurements between two consecutive frames to form relative motion constraints.

Similar to manifold-based filtering, smoothing methods have also been adapted for cases where the optimized states lie on manifolds. Since many smoothing methods can be reduced down to nonlinear least-squares problems, manifold-based optimization techniques [2, 1, 206] (such as the Gauss–Newton method on manifolds) can be employed to solve for these states. Forster et al. [85] combined the ideas behind IMU preintegration and manifold-based smoothing to create a real-time VIO algorithm that respects the geometry in the 3D rotation group. This idea was improved by Eckenhoff et al. [74] through the use of closed-form integration methods. Recently, Chauchat et al. [48] developed a invariant smoothing

technique that parallels the invariant filtering approach. For certain systems, where the state is a matrix Lie group and the dynamics and measurements are "group-affine", this invariant smoother can used to improve convergence and accuracy while potentially eliminating the need for re-linearization.

Despite these breakthroughs in factor graph based odometry and SLAM, most methods heavily rely on visual information and are prone to failure when visual tracking is lost, often due to lighting or scarcity of features. In addition, the state-of-the-art offers no method for incorporating some of the sensors that legged robot are typically equipped with, namely joint encoders and contact sensors. In these scenarios, *kinematic odometry*[2], offers a method for reducing the accumulated drift, which can allow the robot to more easily recover from loss of visual information. In this dissertation, we build off of our work in [110, 108] to develop a method for incorporating kinematic odometry into the factor graph framework. In doing so, we developed two novel factors, which we call forward kinematicand preintegrated contact factors. The forward kinematic factor relates the robot's base frame to a contact frame (located at the intersection of the robot and the environment) through noisy joint encoder measurements. The preintegrated contact factors adopt the concepts behind preintegration theory [160, 85] to allow integration of high-frequency contact measurements through an arbitrary number of contact switches. This binary factor relates the contact frame poses at consecutive keyframes. Together, these factors constrain the robot's base motion leading to improved Maximum-A-Posteriori (MAP) state estimation. In addition, we derive these factors using Lie group theory, and show that this visual-inertial-contact (VIC) odometry system satisfies the "group-affine" property. Therefore, the techniques developed for invariant smoothing can be used to accurately solve this problem. More details can be found in Chapter 5.

2.1.3 Legged Robot and Humanoid State Estimation

Legged robots are a subclass of mobile robots that locomote through direct and switching contact with the environment. These robots typically contain proprioceptive sensors, such as IMU, joint encoders, and contact sensors. In addition, some legged robots, especially humanoids, also have access to exteroceptive sensors, namely cameras and LiDARs. As with all mobile robots, state estimation for legged robots is critical for mapping, planning, designing feedback controllers, and developing general autonomy. In this section, an overview of notable techniques for legged robot state estimation is given.

[2]Kinematic odometry refers to the process of estimating the robot's relative pose transformation (and potentially velocity) from forward kinematic measurements along with contact assumptions. For legged robots, this has also been referred to as "leg odometry".

The simplest approach for estimation of a legged robot's spatial location and velocity is kinematic dead-reckoning, otherwise known as kinematic odometry. This involves estimating relative transformations using only kinematic and contact measurements. In particular, encoder measurements and the kinematics model are used to track the position, orientation, and velocity of the robot's base frame based on the assumption that a stance foot remains fixed to the ground. Although this method can be easily implemented, the state estimate is typically noisy due to kinematic modeling errors, encoder noise, and foot slip [195]. When only one foot is in contact with the ground, this "static contact assumption" may be violated. For example, if the robot has point feet, the stance foot position may remain fixed, but the foot orientation is free to rotate (without changing the joint angles). Therefore, a gyroscope is often used to provide angular velocity measurements which allows the robot's body velocity to be recovered. Alternatively, if the terrain is known a priori and at least 3 noncollinear point feet are on the ground, Lin et at. [156] showed that the robot's instantaneous base pose can be computed through kinematics. These kinematic-based methods have been implemented on a number of legged robots including a planar one-legged hopper [125], the CMU Ambler hexapod [195], the RHex hexapod [156], and the biped robot MARLO [59, 60]. However, due to the high amounts of noise coming from encoders and foot slip, the velocity estimate typically needs to be heavily filtered before becoming usable in the feedback controller [107]. In addition, this noise causes the position and orientation estimates to drift substantially rendering the estimator useless for mapping and autonomy tasks.

Fortunately, legged robots are often equipped with additional sensors such as IMUs, GPS, cameras, or LiDARs which provide independent, noisy odometry measurements. Much of the literature on legged robot state estimation focuses on fusing these measurements (potentially with kinematic odometry) using filtering and smoothing methods. Singh et al. [203] combined inertial measurements with optical flow measurements in a four-phase hybrid EKF. This required explicit dynamic modeling of the robot in flight, landing, stance, and thrust phases. Lin et al. [157] took a similar model-based approach and used an EKF to fuse kinematic information with IMU measurements to estimate the state of a hexapod. Cobano et al. [56] developed an EKF that fuses kinematic odometry and magnetometer readings with position measurements from a GPS to localize a SILO4 quadruped outdoors. This implementation fixes the issues with unbounded drift, but cannot operate in GPS-denied environments. If a prior terrain map is known, Chitta et al. [54] showed that it is possible to solve the localization problem for legged robots using only proprioceptive sensors and a particle filter. The key idea was that if the robot "senses" that a terrain change through kinematics, then this limits the potential locations the robot can be in a known map. The method was demonstrated on the LittleDog quadruped.

A breakthrough came in 2012 when Bloesch et al. [34] combined inertial and kinematic measurements in an observability-constrained ErEKF using the strapdown IMU modeling approach. In this work, no *a priori* knowledge of the terrain is assumed, and the IMU integration model completely eliminates the need for dynamic modeling of the robot. Therefore, the derived filter equations are general enough to be used on any legged robot. The key idea was to augment the state vector with the set of all foot positions currently in contact with the environment. During the prediction phase, the foot contact dynamics are assumed to be Brownian motion, which can account for some foot slippage. In the correction phase, forward-kinematic position measurements are used to correct the estimated state. This work was conducted on the StarlETH quadruped robot. If the stance feet orientations also remain constant, as is the case for many humanoids, Rotella et al. [196] showed that this ErEKF can be extended to allow forward-kinematic orientation measurements. The same group also formulated a similar unscented Kalman filter that uses forward-kinematic velocity measurements to correct inertial predictions and to accurately detect foot slip [35]. A detailed analysis of these filtering techniques combined with methods for incorporating computer vision can be found in Bloesch [33]. Due to the complexity involved in accurately formulating dynamic models, many groups have since adopted this IMU motion model approach to legged robot state estimation [238, 81, 109].

This combined inertial and kinematic filtering approach yields an estimate of the robot's base pose and velocity. However, some legged robots require additional states to be estimated. Hwangbo et al. [133] formulated a probabilistic contact estimator for cases when contact sensors are unavailable. Xinjilefu et al. [238] developed a decoupled EKF that is able to estimate the full state of the humanoid robot ATLAS, including base states, joint angles, and joint velocities. Using proprioceptive sensing only, Bloesch et al. [34] proved that the absolute positions and yaw angles are unobservable. Thus, over time, estimates of these quantities will drift unboundedly. This is unacceptable for global mapping and planning algorithms; however, local elevation maps can still be obtained [82]. Fallon et al. [81] proposed a method for drift-free state estimation for the humanoid ATLAS. In their implementation, inertial and kinematic measurements were fused to yield accurate odometry. Point cloud data from a LiDAR sensor was used with a particle filter to localize the robot into a pre-built map. This approach provided corrections of position and yaw to obtain a drift-free estimate of the state. Nobili et al. [176] took a similar approach but used the iterative closest point (ICP) algorithm to perform LiDAR-based point cloud matching. The algorithm was tested on the HyQ quadruped robot.

In this thesis, two novel legged-robot state estimation techniques are developed. First, in Chapter 4, an invariant extended Kalman filter (InEKF) is designed that can accurately

fuse inertial, kinematic, and contact measurements to obtain fast converging estimates of a robot's base pose, velocity, and contact positions. The approach we take is similar to Bloesch et al. [34], however we model the entire state as a single matrix Lie group as opposed to a decoupled state approach. This allows us to use Barrau and Bonnabel's [24] theory of invariant observer design to take advantage of the geometry and symmetry of the estimation problem to formulate autonomous error dynamics. In addition with our formulation, the linearizations are independent of the state estimate resulting in improved convergence properties, especially for poor state initializations. We formulate both world-centric and robo-centric state estimators highlighting the relation between the left- and right-invariant error dynamics. In addition, we provide exact analytical time-discretizations of both filters. The implemented filters can be run at high speeds ($> 2000\,\text{Hz}$) and can be directly used for accurate local odometry. We demonstrate this idea through a LiDAR terrain mapping application on a Cassie-series biped robot. Using only proprioceptive sensing, the absolute position and yaw will drift over time. To remedy this, we show how incorporate landmark estimation into the same filter to provide drift-free estimation. Next, in Chapter 5, we develop a method for incorporating kinematic odometry into a factor graph based smoothing framework. These novel factors can be combined with existing state-of-the-art visual-inertial and loop-closure algorithms to provide long-term SLAM solutions for legged robots. To the best of the authors knowledge, this is the first time kinematic odometry has been implemented in a full smoothing algorithm. Some of this work was previously published in [110, 108].

2.2 Motion Planning for Legged Robots

The ultimate goal is to develop a general kinodynamic motion planning algorithm that can compute dynamically-stable, collision-free trajectories for legged robots in real-time. Despite the extensive literature [152], many existing planners cannot be directly implemented on legged robots due to switching contact points, high-dimensionality, complex dynamics, and stability requirements. Although a general technique has yet to be developed, a number of researchers have created or adapted specialized algorithms that can solve subclasses of this motion planning problem. Most of these algorithms assume that an accurate state estimate and environment map have already been obtained.

In general, motion planning algorithms for legged robots have been broadly separated into two distinct areas of research: local gait generation, and global path planning. Broadly speaking, most research on global planning provides methods focus on computing collision-free paths through a known map that can bring the robot from an initial state to a goal state. Many of planners tend to follow a "contact-before-motion" formula, where the footstep

locations are computed before the body and joint movements are generated [147, 113, 63]. This is useful for navigation on rough or discontinuous terrain, where the footstep locations are critical. However, the allowable motions are often restricted to satisfy a "quasi-static" criteria. In contrast, dynamic trajectory optimization can be used to generate local gaits that span a much larger range of motions. However, these "high-quality" gaits typically ignore obstacles and are mostly computed offline due to computational complexity [116]. For some scenarios, such as walking on flat ground, the footstep locations do not matter. Therefore, it is possible to plan collision-free routes using standard 2D planning algorithms [148]. The robot can then follow these routes using a sequenced set of precomputed gaits.

2.2.1 Stability Criteria and Gait Classifications

A *gait* is simply a specific style of legged locomotion. In general, gaits can be either periodic and aperiodic. When planning over rough or discontinuous terrain, the robot often has to make specific choices about footfall locations and body movement. In these scenarios, the resulting movements are sometimes called *non-gaited* because the motions are not precomputed to follow a specific style of walking [111]. However, in this dissertation, the term "gait" refers to any legged robot movement, periodic or aperiodic. Unlike wheeled robots and fully-actuated manipulators, stability is an important requirement for legged robot motion planning. Differing definitions of this stability criteria group gaits into three classes: static walking, dynamic walking, limit cycle walking [64].

If the projection of the center of mass (CoM) on the ground is always contained within the robot's support polygon[3], then the gait satisfies the *static stability* requirement. In other words, when the robot is statically stable, all movement can be stopped and the robot will still maintain balance. It is important to note that this is a stability requirement that relies solely on kinematics. This simplifies planning techniques, but heavily restricts allowable configurations. As the size or number of stance feet increases, the relative area of the support polygon will also increase. This, in turn, enlarges the set of valid configurations. This definition of stability has been used on robots such as Ambler [20] and ATHLETE [114].

If the zero-moment point (ZMP) is always contained within the robot's support polygon, then the gait satisfies the *dynamic stability* requirement. The ZMP is defined as the point on the ground where all forces and moments acting on the foot can be replaced by a single force and vertical moment [225, 226]. If the ZMP is computed to be outside the support polygon, then it is labeled a "fictitious" ZMP and the robot will roll about its feet. Therefore algorithms that plan dynamic gaits will attempt to compute motions that keep the ZMP

[3]The support polygon is defined as the convex hull of all supporting feet.

within the robot's support polygon at all times. Every motion that is statically stable is also dynamically stable. While the ZMP criteria extends the set of allowable movements, the inability to roll about the foot and the need for a support polygon are still restrictive. For example, when a robot runs or jumps, there is a period of time when the support polygon does not exist. ZMP-based stability has been widely used on humanoid robots such as H7 [139], P2 [119], ASIMO [198], and HRP-2 [241].

The last style of gait has been called *limit cycle walking* [123, 233]. These gaits are characterized by periodic motions are are turned into stable limit cycles using passive mechanisms or feedback control. Stability can be checked using the method of Poincaré sections [102]. This type of movement does not rely on support polygons, and can therefore be applied to all types of periodic gaits including walking, running, and jumping. It can even be applied to robots with a single underactuated contact, such as a point or line foot. Even more generally, since legged locomotion can be described using a hybrid nonlinear system (as explained in Section 3.1), stability can simply defined using the standard nonlinear systems theory definitions. This allows both periodic and aperiodic motions to be characterized as exponentially stable, asymptotically stable, or stable in the sense of Lyapunov [145]. These notions of stability have been utilized when studying McGeer's [166] passive walkers, Raibert and Tello's [189] hoppers, and a number of other underactuated bipeds: RABBIT [51], MABEL [101], ATRIAS [190], DURUS [117], and Cassie [91].

2.2.2 Local Gait Generation

Gait generation involves computing feasible, CoM or joint trajectories that allow a legged robot to move. Typically, these methods do not account for obstacles and only find a local motion plan that moves the robot a few steps (or even just a single one). These techniques have also been called *walking pattern generation* [141, 239] or *trajectory optimization* [233, 117]. Naturally, these local gaits do not solve the global motion planning problem, but they can be sequenced together to achieve more complex tasks. The methods used for gait generation depend on how the robot is modeled, if footsteps are pre-planned, and which criteria of stability is used.

While some robots rely on static stability for gait generation [246], many legged robots and humanoids use simplified models and the ZMP criteria [225] to generate gaits. Shih et al. [201] used inverse kinematics with a fixed CoM trajectory (constant forward speed, torso height, and sinusoidal lateral movement) to analytically solve for a biped's joint movements. When this computed trajectory violated the ZMP criteria, the motion was heuristically adjusted until dynamic stability was regained. Dasgupta and Nakamura [62] devised a method

for generating ZMP trajectories from human motion capture data. Sugihara et al. [216] used control methods for an inverted pendulum to compute the reference ZMP. This method allowed for real-time gait generation that was verified through humanoid simulations. Compelling experimental examples of ZMP-based gait generation is given by Honda's humanoid robots [120], P2 [119] and ASIMO [198].

Notably, in 2003, Kajita et al. [142] developed a walking pattern generator based on preview control. The robot dynamics were approximated using an 3D linear inverted pendulum model (LIPM) [141] to provide an simple relationship between the CoM dynamics and the ZMP. Assuming a reference ZMP trajectory is given, a ZMP tracking problem is then solved to obtain the desired CoM dynamics. Future information (a preview) of the ZMP trajectory is provided to allow the CoM to change before the ZMP does. The reference ZMP trajectory is assumed to be given by a higher-level footstep planner. Once the CoM trajectory is obtained, inverse kinematics are used to compute the necessary joint angles to achieve the desired motion. If the robot is fully actuated, these joint trajectories can be easily tracked to obtain a dynamic walking motion. Using this method, gaits can be generated that achieve specific foot placement, which is often necessary for navigating rough terrain.

Over time, many other methods for gait generation and control using the ZMP have been developed [132, 239]. Unfortunately, this ZMP restriction limits the allowable movements of the robot. For example, human gaits often involve significant "toe roll", which cannot happen when the ZMP is within the support polygon. In addition, gaits without a support polygon (jumping, running, point-foot walking) are also ignored. *Trajectory optimization* provides an alternate method that can used to generate all of these gaits (including static and ZMP walking). In general, trajectory optimization aims to find an dynamically feasible motion through the minimization of a user-defined cost function (such as energy per step). Since legged locomotion is typically modeled as a hybrid nonlinear system, the resulting problem is called hybrid trajectory optimization. Although this optimization problem only has a single cost function, it can contain numerous equality and inequality constraints. This allows torque limits, friction cones, periodicity, ZMP, and other constraints to be included in the problem formulation. The solution of this optimization is a locally optimal trajectory of the hybrid nonlinear system that satisfies all imposed constraints. Once formulated, this optimization problem is transcribed into a constrained nonlinear program, which is then solved using a nonlinear optimization solver [30]. In some instances, a feedback controller is optimized alongside a trajectory. This is the case for the hybrid zero dynamics (HZD) framework [233], where a set of parameters and *virtual constraints* define both the controller and the trajectory. More information about HZD-based gait design can be found in Section 3.1.4.

In recent years, trajectory optimization has become a widely used method for generating

dynamically-feasible gaits on legged robots. Chevallereau et al. [51] used trajectory optimization to design energetically efficient walking gaits for the planar biped robot RABBIT. Similar methods were used to generate dynamically-feasible gaits (that also satisfy key physical constraints) for the biped robots MABEL [213], and ATRIAS [190, 60]. Unfortunately, as the complexity of the dynamic model increases, so does the gait computation time. Therefore, most legged robot trajectory optimization problems are solved offline. Recently, Hereid et al. [117] developed a direct collocation framework for optimizing the HZD of a robot. This direct method offers a considerable speedup over similar "shooting-based" solvers, however, real-time gait computation is still intractable for high-dimensional robots. An open-source implementation of this optimizer, Fast Robot Optimization and Simulation Toolkit (FROST) [116], is available. FROST has been used to generate gaits for the biped robots MARLO [107], DURUS [117], Cassie (this dissertation), and even the exoskeleton ATALANTE [106]. Most hybrid optimization methods, including FROST, require contact mode scheduling beforehand. Typically, the sequence of contact modes is determined heuristically, however, this presents a combinatorial problem for robots with many contact modes. Posa et al. [181] developed a "through contact" optimization method that avoids this scheduling pitfall. This method was used to generate gaits for the FastRunner biped.

2.2.3 Global Planning Methods

Unlike local gait planning, global planning aims to find a dynamically-feasible, collision-free path through an environment that brings the robot from an initial configuration to a goal configuration. Most global planning methods for legged robots limit the search space to robot configurations that satisfy either the static or dynamic equilibrium condition. This restricts the allowable motions of the robot, but simplifies the global planning problem. These methods fall under the category of *quasi-static* planners. Early work primarily focused on static walking for robots with many legs. Beginning in 1986, Hirose et al. [122], Hirose and Kunieda [121] outlined a method for motion planning over rough terrain with a quadruped robot. This algorithm used geometric techniques to select an appropriate location for the swing foot that kept the CoM within the support polygon. Similar static planners were devised for the hexapod Ambler [20] and general spider robots [36].

In 2002, Kuffner et al. [150] developed a dynamically-stable motion planning algorithm for humanoid robots. This algorithm assumes that the initial and final posture of the robot is statically stable, and the contact points do not change. A modified rapidly-exploring random tree (RRT) [149] is then employed to construct a trajectory between the initial and goal posture. This RRT only samples from statically stable, collision-free configurations that

were computed a priori. This trajectory was then sent through a dynamics filter function "AutoBalancer" [138] to ensure that the final trajectory satisfies the ZMP criteria. The end result is a dynamically-stable, collision-free trajectory. Following the "contact-before-motion" formula, Kuffner et al. [147] then extended this algorithm to include switching contact points which enables the humanoid to move though the environment while avoiding obstacles. First, a library of precomputed, reachable footstep positions is used to develop a tree-based footstep planner. Once a feasible sequence of footsteps is computed, the before mentioned dynamically-stable motion planning algorithm [150] is used to plan the robot's motions between footsteps. These planners were tested on the H6 and H7 humanoid robot platforms [139]. Chestnutt et al. [50] developed a similar footstep planner to generate collision-free paths for the ASIMO robot. The algorithm used a finite set of state-dependant actions to generate footstep locations using a A-star search algorithm. More recently, Deits and Tedrake [63] developed a footstep planner based on mixed-integer convex optimization. After a linear approximation of sinusoids, their planner can be solved to a global optimum and can handle kinematic reachability and obstacle avoidance. Simulations with an Atlas biped robot were used to verify the algorithm.

Eldershaw and Yim [76] proposed a solution to the legged robot motion planning problem by combining a probabilistic roadmap (PRM) [144] with a heuristic footstep planner. The search space is discretized into cells, and a PRM is used to compute a collision-free trajectory for the CoM. An A-star search algorithm is used to connect the random samples while checking if a statically stable configuration exists. In the next phase, the footstep planner attempts to find foot positions and body motions that move the CoM between cells. If there is no feasible path, the PRM replans. Li et al. [155] deconstructed the humanoid motion planning problem into a global and local planner. The global planner computes a collision-free path for the upper-half of the robot using a variation on the best-first search algorithm. The local planner takes this reference trajectory and kinematically plans only the next several steps for the lower half of the robot. Bézier curves are used to reshape the swing foot trajectory to avoid collisions with the ground. Overall, this planner allowed their simulated robot to navigate around a layered or two and half dimensional environment.

Yoshida et al. [241] took a two stage approach to generating dynamically-feasible motion plans. First, a geometric and kinematic planner computes a collision-free path in 2D for the body along with joint angles through inverse kinematics. In the second phase, this information is fed into Kajita et al. [142] walking pattern generator that computes a dynamically stable motion plan based on the ZMP criteria. If the resulting dynamic plan collides with any obstacles, it is reshaped and the dynamic plan is recomputed. This allowed a simulated HRP-2 humanoid to navigate through narrow spaces while carrying objects. Hauser

et al. [111] developed a non-gaited humanoid motion planner following the "contact-before-motion" idea. A PRM was used to plan the next step using iterative constraint enforcement to quickly sample from feasible statically stable configurations. This method allows a robot to use any pre-defined contact points for motion planning, including hands and knees. Their planner was tested on a simulated HRP-2 humanoid. The same group extended this work by developing stance and transition graphs define the connectivity of the robot's configurations [114, 113]. Combining these graph ideas with a PRM variant allowed static planning over varied terrain for both the HRP-2 and ATHLETE robots. Hauser et al. [112] combined the strengths of precomputed motion primitives with these stance and transition graphs. The PRM planner then used these motion primitives to guide the growth of the search tree allowing higher-quality motions to be obtained.

Unfortunately, most of these global planning algorithms only work for robots and gaits that satisfy the static or ZMP stability requirements. Therefore, gaits without a support polygon (running and jumping) are not supported. In addition, if a biped has point or line feet, there will not even be a support polygon during the single-support phase of simple walking gaits. This is the case for both the ATRIAS (point-feet) and Cassie (line-feet) robots used throughout this dissertation. For example, during the double-support phase, Cassie can maintain either static or ZMP-based stability. However, as soon as one foot leaves the ground, the support polygon vanishes and there are no configurations that put Cassie in static or ZMP-based stability. Therefore, none of the before-mentioned "quasi-static" planners can be used for underactuated robots. Some notable research on dynamic planning without support polygons has been done by Chestnutt [49] and Zhao et al. [245]. Recent advances in gait optimization have allowed for relatively fast, off-line computation of "high-quality" motions [4] [117, 116]. Therefore, it is possible to build up libraries of these "motion primitives" that can be used for planning. Other researchers have explored the use of these trajectory libraries for humanoid balance [158], unmanned aerial vehicles (UAVs) [87, 163], grasping [28], and even controller design for underactuated legged robots [58, 60, 59, 107, 187, 175]. However, the utility of trajectory libraries for global motion planning of underactuated legged robots has yet to be explored.

2.3 Controller Design for 3D Bipedal Robots

Even if we obtain perfect state estimation and optimal reference trajectories, the robot will never realize this motion without a feedback controller. Generally, these controllers takes in

[4] "High-quality" is used here to describe trajectories which are solutions to the full-order dynamic model of the legged robot that also satisfy key physical constrains such as torque limits, friction cones, etc.

some reference command or gait and calculate the appropriate torques to send to the robot's actuators. In an ideal case, the feedback controller will enforce stability of the gait while rejecting bounded disturbances and model errors. This section provides a brief overview of feedback controller design methods for bipedal robots.

2.3.1 ZMP-Based Walking

The majority of 3D bipedal gaits are stabilized through control of the zero-moment point (ZMP) [131, 198, 150, 241]. After planning footsteps, the reference ZMP trajectory is typically converted into a center of mass (CoM) trajectory and eventually a joint trajectory through inverse kinematics [142]. Once a joint trajectory is obtained, a number of low-level tracking controllers can be implemented. As mentioned before, this method produces "quasi-static" gaits that can only be implemented on "fully actuated" robots. Although this method can produce stable walking, the gaits are not robust to the loss of actuation that occurs during foot roll. If a perturbation causes the ZMP to move to the edge of the support polygon [225], stability will be lost without an auxiliary controller. Although a powerful technique for bipedal control design, since ZMP walking requires a non-zero support polygon, it cannot be applied to either robot used in this dissertation; MARLO has point feet, and Cassie has line feet. Therefore, no further topics on ZMP-based control will be reviewed.

2.3.2 Simplified Models and Foot Placement Techniques

If a robot is underactuated, simply tracking a joint trajectory may not lead to a stable walking motion. Therefore, the feedback controller may need to modify the original reference trajectory to regain stability. The high-dimensionality and complex dynamics present in many biped robots limits the methods for online planning, thus simplified models are often used to develop and understand controllers. Some of these simplified models include the inverted pendulum [216], the linear inverted pendulum model (LIPM) [141], and the spring loaded inverted pendulum (SLIP) model [98, 231, 179]. These simplified models are commonly used to develop heuristics for gait stabilization. Many of these heuristic techniques can be broadly grouped into so-called "foot placement" methods, which involve placing the swing foot appropriately to actively maintain stability [221]. Unlike footstep planning, which plans multiple steps to avoid obstacles, foot placement methods typically compute only a single swing foot location while ignoring obstacles. In other words, footstep planning is used for global planning, while foot placement is used to achieve stability. Despite the considerable success of foot placement techniques on a number of robots [189, 179, 194, 59], there is no agreed upon model or method for determining where to place the swing foot.

Hodgins and Raibert [124, 189] implemented foot placement through the notion of a "neutral point", the swing foot position that will leave the velocity unchanged. If the swing foot is placed ahead of the neutral point, the robot will slow down. Placement behind the neutral point will cause the robot to accelerate. To stabilize a gait at a fixed velocity, the desired displacement of the swing foot position from the neutral point was a linear function of velocity error. Dunn and Howe [72] used a similar gait asymmetry to develop a method of foot placement to regulate velocity on a planar biped. The foot placement estimator (FPE), developed by Wight et al. [235], has also been used to develop control strategies. The FPE is the point on the ground where, when the CoM moves over it, all kinetic energy has been turned to potential energy, which would leave the robot balanced above the point. Van Zutven et al. [224] generalized the FPE to robots with an arbitrary number of non-massless links. This generalization was called the foot placement indicator (FPI). Desired swing foot placement has also been estimated though other stability measures, such as the foot rotation indicator (FRI) [92] and the notion of capture regions [185]. Recently, Da et al. [59] used a linear foot placement technique in conjunction with a library of gaits to obtain velocity-regulated, 3D walking on MARLO. Although foot placement techniques typically involve only changing the swing foot positions, there has been some research into heuristically modifying other postural aspects of the gait to achieve stability [182, 107].

2.3.3 Hybrid Zero Dynamics

The theory of *hybrid zero dynamics* (HZD), developed by Grizzle, Westervelt et al. [233], has emerged as a powerful framework for designing gaits along with stabilizing feedback controllers for underactuated biped robots. At its core, the hybrid zero dynamics (HZD) framework is built around the idea of *virtual constraints* that are enforced though feedback control. Typically, these virtual constraints are a function of the robot's configuration variables (they are holonomic) and encode a desired walking motion for a set of chosen coordinates. Unlike physical holonomic constraints, which are enforced though mechanical connections, virtual holonomic constraints are enforced through feedback control. Specifically, output functions are defined using these virtual constraints, and when driven asymptotically to zero, force the chosen coordinates to track the desired motion. This drives the full, high-dimensional nonlinear system to a low-dimensional, invariant *hybrid zero dynamics* surface. Notions of stability on this low-dimensional surface can be used to provide stability measures for the full-order dynamical system. A detailed overview of the HZD framework can be found in [233]. An important consequence of this theory is that exponentially stable orbits of the hybrid zero dynamics surface equates to exponentially stabilizable orbits for the full-order

system [232]. In other words, if the zero dynamics orbit is stable, a feedback controller can be designed to render the robot's walking gait stable. The stability of the zero dynamics orbit can be checked using the method of Poincaré sections [102, 169].

Chevallereau et al. [51] utilized the HZD framework to design stabilizing walking controllers for the planar biped RABBIT. Sreenath et al. [212] were able to achieve compliant walking and running [214] on the planar biped MABEL within the same HZD framework. A similar style of running was also obtained by Ma et al. [161] on the planar DURAS-2D biped. For 2D walking, the exact conditions required for asymptotic stability of 2D walking can be analytically computed and are independent of the virtual constraint choice [233, 100]. However, this does not extend to 3D walking, where many choices of virtual constraints will lead to an unstable zero dynamics orbit [100]. Despite this limitation, a number of methods, both heuristic and theoretical, have been used to overcome this problem. Full 3D, HZD based walking has been implemented on the bipeds DURUS [117, 192], MARLO [45, 96, 59, 107, 60], and Cassie [91]. Similar controllers have been created for powered prosthetics [93] and exoskeletons [5, 106].

After defining the virtual constraint outputs, the system can be described using two sets of coordinates: the zero dynamics and transverse dynamics coordinates. When the virtual constraint outputs are identically zero, the nonlinear system is constrained to a lower-dimensional surface; the resulting submanifold is denoted the *zero dynamics surface*. The coordinates describing the dynamics on this surface are called the *zero dynamics coordinates*. The coordinates describing the dynamics transversal to the zero dynamics surface are called the *transverse dynamics coordinates*. Under the same HZD framework, many researchers have developed differing methods for ensuring stable orbits for both the transverse dynamics and zero dynamics.

2.3.3.1 Transverse Dynamics

The choice of virtual constraints defines the coordinates that are are actuated. The dynamics of these transverse coordinates are governed by the implemented feedback controller. Over the years, a number of methods for stabilizing the transverse dynamics have been used. In [102, 232] a finite-time controller [31] is used to drive the system to the zero dynamics surface in finite time, which is useful because the impact maps associated with biped walking are typically expansive (pushing the state even further away from the surface) [11]. In many HZD-based controllers, input-output feedback linearization [145] is used to render the transverse dynamics exponentially stable [233]. This feedback linearization technique has been used to realize stable walking on a number of bipeds [52, 169, 212]; though in practice, the feedforward term is often dropped to yield a simple PD controller [51, 60, 107].

Ames et al. [11, 10] used the theory of control Lyapunov functions (CLFs) [209, 16] to extend the class of controllers that can be used to stabilize the transverse dynamics. Using this theory, an closed-form *point-wise minimum norm* control law [88] can be formulated that uses the minimum torques needed to keep the derivative of the Lyapunov function negative definite. This controller was tested on the biped robot MABEL [11]. Galloway et al. [89] re-formulated this "min-norm" controller as the solution to a quadratic program (QP). This QP formulation allows additional constraints, such as torque limits, to be included. A relaxation term was added to the CLF constraints ensure solvability of the QP, as it may not always be possible to satisfy the CLF criteria when then torques are bounded. A thorough description of these control Lyapunov function based quadratic programs (CLF-QPs) along with a connection between legged locomotion and manipulation is provided in [9]. Nguyen and Sreenath [174] developed a robust CLF-QP that can handle a greater degree of model uncertainty.

2.3.3.2 Zero Dynamics

To ensure stability of the full dynamical system, both the transverse dynamics and the zero dynamics have to be stable [232]. Although it is relatively easy to stabilize the transverse dynamics (detailed in the above section), the stabilization of the zero dynamics presents a greater challenge. By definition, the zero dynamics of a nonlinear system are uncontrollable if the number of virtual constraints is equal to the number of actuators. Therefore, if the zero dynamics are unstable for a fixed set of virtual constraints, the full system will also be unstable (even if the transverse dynamics are stabilized). Unfortunately, this is often the case for underactuated 3D walking. It is possible, however, to recover stability by either redesigning the virtual constraint trajectories or modifying them online.

One potential method for ensuring stability is to enforce the existence of a stable zero dynamics orbit during the gait generation phase. Chevallereau et al. [52] did this for a five-link walker by constraining the eigenvalues of the linearized Poincaré map to be strictly within the unit circle. [52]. This has also be done as a post-processing step where a precomputed, unstable gait is stabilized by redesigning the virtual constraints. Akbari Hamed et al. [6] developed a method for the redesign step using Bilinear Matrix Inequalities (BMIs) built from performing a sensitivity analysis of the Poincaré map. Another method, developed by Griffin and Grizzle [96], utilized *nonholonomic* virtual constraints to allow velocity-based posture regulation that can lead to stable zero dynamics.

Stabilization of the zero dynamics can also be achieved through online modification of the virtual constraint trajectories (and therefore the transverse dynamics). For example, imagine a fixed set of virtual constraints designed for walking in place. The computed

virtual constraint trajectories may keep the leg angles constant and drive the swing knee angles to bend mid-step. For an underactuated robot, this may be an unstable strategy when the robot is pushed forwards. However, if the desired swing leg angle trajectory was modified to swing forward, stability of the gait could be recovered. This gait modification is precisely the idea behind swing foot placement techniques. Da et al. [59] used this method to stabilize 3D walking on MARLO using velocity errors to modify the desired swing leg trajectory. Reher et al. [191] used a similar method to stabilize the lateral motion on the DURUS robot. In [107], optimization-based perturbation recovery was used to create posture adjustment functions (PAFs) that can modify any joint trajectory on the robot to recover stability.

Although most virtual constraints are configuration-based (and therefore relative degree two), velocity or other relative degree one outputs can be chosen. The surface defined when all relative degree two outputs are identically zero is denoted the *partial hybrid zero dynamics* (PHZD) surface [7, 8]. For fully actuated robots, the partial hybrid zero dynamics (PHZD) can become controllable allowing for stabilization techniques and speed regulation [183].

2.3.3.3 Safety Critical Constraints

In addition to stabilization, we may wish to enforce safety critical constraints using a feedback controller. During biped walking, for example, the controller should keep the swing foot from scuffing the floor, and when encountering rough terrain, precise foot placement may be required. Implementation of control barrier functions (CBFs) provide one method for achieving these safety critical constraints. These CBFs were first used in numerical optimization [42], but have also been used in nonlinear control due to their "dual" relationship to Lyapunov functions [234]. In 2014, Ames et al. [12] developed a control barrier function based quadratic program (CBF-QP) that can be used for safety critial systems. The use of a QP allows for the unification of safety constraints with stability constraints, coming from a CLF-QP. The resulting CBF-CLF-QP was evaluated on automatic cruise control and lane keeping problems [13]. Hsu et al. [128] brought CBFs to biped locomotion by designing a CBF-CLF-QP that allowed a planar model of AMBER2 to stably walk while enforcing physical constraints, such as swing foot clearance and knee angle constraints. Nguyen and Sreenath [173] used a similar formulation to enforce precise foot placement on a model of the planar RABBIT robot. Quan Nguyen et al. [187] later extended this method to develop a CBF-CLF-QP that achieved precise foot placement for 3D walking on stepping stones using a model of the DURUS biped.

2.3.3.4 Gait Libraries

In order for a legged robot to perform useful tasks, a wide range of behaviors may be needed (walking at different speeds, turning, running, climbing stairs, etc.). This motivates the use of trajectory or gait libraries. In 2016, Da et al. [59] used a library of eight gaits that were each optimized for a single sagittal walking speed. These gaits were then interpolated based on the current velocity measurement to create a continuously defined controller that was marginally stable. Asymptotic stability was recovered by augmenting the controller with a "foot placement" strategy to regulate speed. Da et al. [60] later included gaits that transition between periodic orbits into the gait library. The combination of periodic gaits, transition gaits, and supervised machine learning allowed control policies to be built that stabilized 3D walking on MARLO. A terrain adaptation policy was generated using a 2D grid of periodic gaits with varying sagittal velocities and step heights. This allowed MARLO to stably walk though the valleys of the University of Michigan's Wave Field. Future terrain height was estimated based on prior kinematic measurements. In [107], we used a 2D grid of periodic gaits to reject velocity disturbances and to extend the region of attraction of a 3D walking controller. This 2D grid of gaits included gaits for walking forwards, backwards, sideways, and diagonally. This effort is described in Section 6.2. Nguyen et al. [175] used a gait library of differing step lengths and step height to achieve planar dynamic walking across randomly varying, discrete terrain. This was verified experimentally on an planar ATRIAS-series biped by giving the robot a one-step preview of the terrain. Quan Nguyen et al. [187] also combined gait libraries with CBFs to achieve precise foot placement and dynamic walking on stepping stones. Harib et al. [106] used gait libraries and supervised learning to generate a stable walking controller for the ATALANTE exoskeleton. This gait design process, called *generalized hybrid zero dynamics* (GHZD), was fully developed by Da and Grizzle [58]. In this thesis, Chapter 6 gives an description of some of the gait library techniques that have been implemented on MARLO and Cassie.

CHAPTER 3

Background

This chapter presents an overview of the background material useful for understanding the remainder of the dissertation. Section 3.1 presents the mathematical model used for legged locomotion as well as a description of the two robots (ATRIAS and Cassie) used for experiments. Section 3.2 gives a short introduction to the extended Kalman filter (EKF). Section 3.3 gives an overview of smoothing using factor graphs and optimization on manifolds. Section 3.4 provides introduction to Lie group theory along with useful expressions for all groups used in this dissertation. Finally, Section 3.5 provides a overview of forward kinematics and the various forms of its Jacobians.

3.1 Legged Locomotion

Throughout this dissertation, simulations and experiments were performed on either an ATRIAS-series or a Cassie-series biped robot. This section provides a description of these robots, along with necessary background information on the hybrid system that is used for modeling legged locomotion. These mathematical models of the robots are later used to optimize dynamically-feasible gaits and for feedback control design.

3.1.1 Dynamic Model of Legged Locomotion

Legged locomotion can be modeled using a hybrid system consisting of a set of continuous phases along with discrete transitions between them [233]. Together, the continuous dynamics and the discrete map form a *domain*. Steady-state walking can be modeled as a cyclic directed graph composed from an ordered set of these domains [117]. The set of domains and the corresponding dynamics are robot and application specific, but most can be described through the same general equations.

The continuous dynamics can be modeled by the constrained Euler-Lagrange equation,

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{B}\mathbf{u} + \mathbf{J}^{\mathsf{T}}(\mathbf{q})\mathbf{F}, \tag{3.1}$$

where $\mathbf{q}$ is vector of generalized coordinates, $\mathbf{D}(\mathbf{q})$ is a positive-definite mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the coriolis matrix, $\mathbf{G}(\mathbf{q})$ is a vector that describes the contribution of gravity to the dynamics, and $\mathbf{B}$ is a matrix that maps the inputs to the state space. The last term represents a vector of external forces $\mathbf{F}$ that enforce any holonomic constraints (such as pinning the stance foot to the ground), which are mapped to the state space through the constraint's Jacobian, $\mathbf{J}(\mathbf{q})$. These holonomic constraints take the form $\mathbf{h}(\mathbf{q}) = \mathbf{0}$. To enforce this relation over time, the constraint's acceleration needs to be constrained to zero using

$$\mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{0}. \tag{3.2}$$

A guard S is formed from a subset of the boundary of the domain [117]. This represents the condition that causes the continuous dynamics phase to end. Here S is defined as the set of conditions which place the swing foot in contact with the ground. This can be represented by the set

$$\mathcal{S} = \{(\mathbf{q}, \dot{\mathbf{q}}) \in \mathcal{TQ} \mid p^v_{\text{sw}}(\mathbf{q}) = c_g,\ \dot{p}^v_{\text{sw}}(\mathbf{q}, \dot{\mathbf{q}}) < 0\}, \tag{3.3}$$

where p^v_{sw} denotes the switch foot's vertical position and c_g represents the ground height.

Domain transitions are modeled as an instantaneous, discrete map. For impacts (swing foot hitting the ground), this results in a discontinuity in velocity [233]. The post-impact velocities need to satisfy:

$$\begin{bmatrix} \mathbf{D}(\mathbf{q}^-) & -\mathbf{J}^{\mathsf{T}}_{D^+}(\mathbf{q}^-) \\ \mathbf{J}_{D^+}(\mathbf{q}^-) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}^+ \\ \delta\mathbf{F} \end{bmatrix} = \begin{bmatrix} \mathbf{D}(\mathbf{q}^-)\dot{\mathbf{q}}^- \\ \mathbf{0} \end{bmatrix} \tag{3.4}$$

where $(\mathbf{q}^-, \dot{\mathbf{q}}^-)$ is the pre-impact state, $(\dot{\mathbf{q}}^+)$ is the post-impact velocity, $\delta\mathbf{F}$ is a vector of impulse forces, and $\mathbf{J}_{D^+}(\mathbf{q})$ is the Jacobian matrix of the subsequent domain's holonomic constraints. The configuration of the system remains constant throughout impact ($\mathbf{q}^- = \mathbf{q}^+$).

Remark 1. When constraints are only removed between domains, the vector of impulse forces will be equal to zero and the impact map can be simplified. This can happen when the robot switches from double support to single support, or single support to flight.

Figure 3.1: MARLO, the Michigan copy of an ATRIAS-series robot designed by the Dynamic Robotics Laboratory at Oregon State University

3.1.2 ATRIAS-series Biped Robot Description

MARLO, shown in Figure 3.1, is an underactuated ATRIAS-series biped robot. This robot has 4 motors capable of controlling leg motion in the sagittal plane and 2 motors capable of controlling the legs in the lateral plane; i.e. $\mathbf{u} \in \mathbb{R}^6$. The springs are assumed to be sufficiently stiff enough to ignore for the purpose of modeling and gait design. The configuration variables for the floating base model can be defined as $\mathbf{q} := (x, y, z, q_z, q_y, q_x, q_{1R}, q_{2R}, q_{3R}, q_{1L}, q_{2L}, q_{3L}) \in \mathbb{R}^{12}$. The variables (x, y, z, q_z, q_y, q_x) correspond to the absolute world frame position and orientation (ZYX Euler angle convention) of the robot's base. The variables (q_{1R}, q_{2R}, q_{3R}) and (q_{1L}, q_{2L}, q_{3L}) refer to the local coordinates that describe the joint angles of the right and left side of the robot respectively. Figure 3.2 shows the coordinate system used. When designing walking gaits, the knee angle, $q_{KA} \triangleq q_2 - q_1$, and the leg angle, $q_{LA} \triangleq \frac{1}{2}(q_1 + q_2)$, provide more intuitive coordinates for the robot. Note that a four-bar mechanism constrains the remaining joints on the robot. For more information about the mechanics of the robot, refer to [190].

MARLO has point feet, allowing us to model contact using a point contact model where

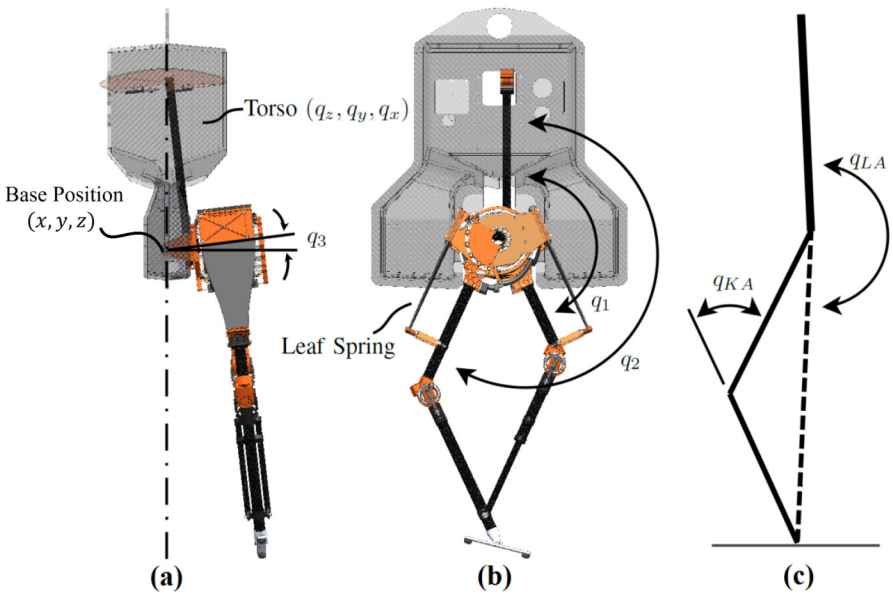


Figure 3.2: MARLO coordinate system

the foot's three position degrees of freedom (DOF) are constrained. Therefore, when one of MARLO's feet is on the ground, the position of the foot is assumed to remain fixed in the world frame. This can be represented using a holonomic constraint

$$\mathbf{h}_{\text{stance foot}}(\mathbf{q}) \triangleq \mathbf{p}_{\text{st}}(\mathbf{q}) - \mathbf{c}_{\text{st}} = 0, \tag{3.5}$$

where $\mathbf{p}_{\text{st}}(\mathbf{q})$ is the position of the stance foot computed using the robot's base pose and configuration, while $\mathbf{c}_{\text{st}}$ is a constant vector denoting the position that the stance foot will be constrained to. This holonomic constraint, along with the continuous dynamics (3.1) and impact map equations (3.4), are used to model point-foot walking.

3.1.3 Cassie-series Biped Robot Description

Cassie (Blue), show in Figure 3.3, is a biped robot designed by Agility Robotics that is capable of 3D walking. The robot has 10 actuators (5 on each side) allowing hip abduction, hip rotation, hip flexion, knee rotation, and toe pitch. There are also 4 springs (2 on each side); one spring is in series connecting the knee motor and the shin, the other spring connects the tarsus to the thigh to form a four-bar linkage. When relaxed, this "tarsus" spring keeps the thigh and tarsus 13 degrees off from parallel. The floating base model of Cassie has 20 DOF with 10 coming from the actuated joints, 4 from the springs, and 6 from the pose of the base. The links and joints of the robot are shown in Figure 3.4, and details about the

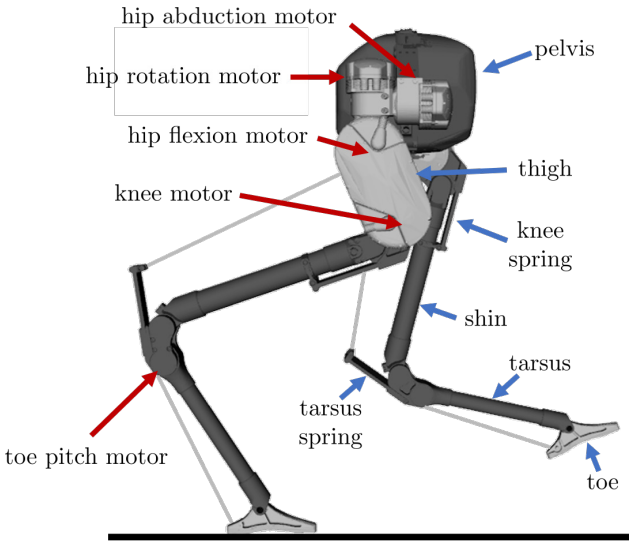
Figure 3.3: Cassie (Blue) is a biped robot developed by Agility Robotics. It has 20 degrees of freedom, 10 actuators and 4 springs. The robot is equipped with an inertial measurement unit, joint encoders. In this picture, a Multisense S7 stereo camera is also mounted.

motors are provided in Table 3.1.

The robot is equipped with an VN-100 inertial measurement unit (IMU) located in the pelvis and 14 encoders to measure each joint angle (including the springs). The robot is also equipped with a Multisense S7 stereo camera shown in Figure 3.3. When the robot is standing, the knee and tarsus springs will deflect from their nominal positions. These spring deflections can be measured by the joint encoders and are used to estimate ground reaction forces and to estimate which feet are in contact.

3.1.3.1 State Description

Cassie's configuration vector, $\mathbf{q} = [\mathbf{q}_B, \mathbf{q}_L, \mathbf{q}_R]^\mathsf{T}$, is comprised of 20 variables coming from the body pose, $\mathbf{q}_B$, the left leg, $\mathbf{q}_L$, and the right leg, $\mathbf{q}_R$. The body and leg configurations



Motor Name	Gear Ratio	Max Torque (Nm)	Max Speed (rad/s)
hip abduction	25:1	4.5	12.1475
hip rotation	25:1	4.5	12.1475
hip flexion	16:1	12.2	8.5085
knee	16:1	12.2	8.5085
toe pitch	50:1	0.9	11.5192

Table 3.1: Cassie motor properties

Figure 3.4: Location of Cassie's motors and links

are defined as

$$\mathbf{q}_B = \begin{bmatrix} x \\ y \\ z \\ q_z \\ q_y \\ q_x \end{bmatrix} \quad \text{and} \quad \mathbf{q}_{L/R} = \begin{bmatrix} q_{\text{ hip abduction}} \\ q_{\text{ hip rotation}} \\ q_{\text{ hip flexion}} \\ q_{\text{ knee}} \\ q_{\text{ knee spring}} \\ q_{\text{ ankle}} \\ q_{\text{ toe}} \end{bmatrix}, \tag{3.6}$$

where x, y, and z are the base positions in the world (inertial) frame, and q_z, q_y, and q_x denote the yaw, pitch, and roll angles of the base relative to the world frame. We adopt the intrinsic ZYX Euler angle convention for the rest of this dissertation. In other words, the rotation matrix describing the base orientation is given by: $\mathbf{R} = \mathbf{R}_z(q_z)\mathbf{R}_y(q_y)\mathbf{R}_x(q_x)$. Each leg joint angle is measured relative to the link before it. The ankle angle is defined as the angle between the shin and the tarsus.

3.1.3.2 Domains

Due to the high stiffness properties of the springs, for trajectory optimization we often use a springless model of Cassie. This can be achieved by simply imposing a set of holonomic constraints that forces the spring deflection to be always zero. The rigid knee spring constraint

can be written as

$$\mathbf{h}_{\text{knee spring}}(\mathbf{q}) \triangleq \begin{bmatrix} q^L_{\text{knee spring}} \\ q^R_{\text{knee spring}} \end{bmatrix} = \mathbf{0}_{2\times 1}. \tag{3.7}$$

The four-bar constraint (that keeps the tarsus spring deflection at zero) is defined by

$$\mathbf{h}_{\text{four-bar}}(\mathbf{q}) \triangleq \begin{bmatrix} q^L_{\text{knee}} + q^L_{\text{ankle}} - 13 \ \text{deg} \\ q^R_{\text{knee}} + q^R_{\text{ankle}} - 13 \ \text{deg} \end{bmatrix} = \mathbf{0}_{2\times 1}. \tag{3.8}$$

Cassie's feet are very narrow (only 3 cm), allowing us to model contact using a line contact model which constrains 5 DOF of the foot. Therefore, when one of Cassie's feet is on the ground, the position, yaw angle, and pitch angle is assumed to remain rigid with respect to the world frame. This can be represented using the holonomic constraint,

$$\mathbf{h}_{\text{stance foot}}(\mathbf{q}) \triangleq \mathbf{p}_{\text{st}}(q) - \mathbf{c}_{\text{st}} = \mathbf{0}, \tag{3.9}$$

where $\mathbf{p}_{\text{st}}(\mathbf{q})$ is the position, yaw angle, and pitch angle of the stance foot computed using the robot's base pose and configuration, while $\mathbf{c}_{\text{st}}$ is a constant vector denoting the position and angles that the stance foot will be constrained to.

Using these holonomic constraints, along with the continuous dynamics (3.1) and impact map equations (3.4), we can now define a set of domains for flat-foot walking. The *flight* domain is defined when both feet are off the ground. The holonomic constraints imposed are

$$\mathbf{h}(\mathbf{q}) = \begin{bmatrix} \mathbf{h}_{\text{knee spring}}(\mathbf{q}) \\ \mathbf{h}_{\text{four-bar}}(\mathbf{q}) \end{bmatrix} = \mathbf{0}_{4\times 1}. \tag{3.10}$$

The *single-support* domain is defined when only one foot is on ground. The holonomic constraints imposed are

$$\mathbf{h}(\mathbf{q}) = \begin{bmatrix} \mathbf{h}_{\text{knee spring}}(\mathbf{q}) \\ \mathbf{h}_{\text{four-bar}}(\mathbf{q}) \\ \mathbf{h}_{\text{stance foot}}(\mathbf{q}) \end{bmatrix} = \mathbf{0}_{9\times 1}. \tag{3.11}$$

The *double-support* domain is defined when both feed are on ground. The holonomic constraints imposed are

$$\mathbf{h}(\mathbf{q}) = \begin{bmatrix} \mathbf{h}_{\text{knee spring}}(\mathbf{q}) \\ \mathbf{h}_{\text{four-bar}}(\mathbf{q}) \\ \mathbf{h}_{\text{left foot}}(\mathbf{q}) \\ \mathbf{h}_{\text{right foot}}(\mathbf{q}) \end{bmatrix} = \mathbf{0}_{14\times 1}. \tag{3.12}$$

3.1.3.3 Computing Kinematics and Dynamics

After developing a URDF file describing the Cassie's kinematic tree and link properties, the kinematics and dynamics were computed using Fast Robot Optimization and Simulation Toolkit (FROST) [116]. This toolkit also provides methods for trajectory optimization and simulation of this hybrid dynamical system.

3.1.4 Virtual Constraints and Hybrid Zero Dynamics

The feedback controllers described in this dissertation follow the hybrid zero dynamics (HZD) design framework. The general idea is to design a set of virtual constraints that are enforced through feedback control [233]. These virtual constraints have the form:

$$\mathbf{y}(\mathbf{q}, \boldsymbol{\alpha}, \tau) = \mathbf{y}_a(\mathbf{q}) - \mathbf{y}_d(\boldsymbol{\alpha}, \tau) \tag{3.13}$$

where $\mathbf{y}_a(\mathbf{q})$ represents the actual outputs that are to be controlled, and $\mathbf{y}_d(\boldsymbol{\alpha}, \tau)$ represents their desired paths. Driving $\mathbf{y}$ to zero forces $\mathbf{y}_a$ to track $\mathbf{y}_d$. The desired paths can be parameterized by Bézier Polynomials of degree M:

$$y_d^i(\tau, \boldsymbol{\alpha}) = \sum_{k=0}^{M} \alpha_k^i \frac{M!}{k!(M-k)!} \tau^k (1-\tau)^{M-k} \tag{3.14}$$

Here $\tau \in (0, 1)$ is a monotonically increasing variable that can be parameterized by either time or by the state. Trajectory optimization methods [233, 117, 116] can be used to determine the Bézier coefficients, $\boldsymbol{\alpha}$, that correspond to a dynamically feasible gait.

Once the virtual constraints are designed, input-output feedback linearization can used to compute the torques necessary to drive the them to zero. Using the Lie derivative notation, the torque equation becomes:

$$\mathbf{u} = (-L_g L_f \mathbf{y}(\mathbf{x}))^{-1} (L_f^2 \mathbf{y}(\mathbf{x}) + \mathbf{K}_p \mathbf{y} + \mathbf{K}_d \dot{\mathbf{y}}) \tag{3.15}$$

It is also possible to use a constant decoupling matrix and to drop the feedforward term resulting in a simple PD tracking controller.

3.2 Extended Kalman Filtering

In general, filtering provides a method for estimating a robots current state based on a collection of noisy measurements. For linear systems with additive Gaussian noise, the

Kalman Filter [143] provides the optimal solution (minimum mean squared error) for state estimation. The extended Kalman filter (EKF) offers a way to extend this filter to the class of nonlinear systems by linearizing the dynamics and measurement models about the current estimate [219]. In the following section, the general equations for the continuous-dynamics, discrete-measurement EKF are summarized.

3.2.1 Continuous/Discrete

Let $\mathbf{x}$ be the state vector and $\mathbf{u}$ be the vector of system inputs. The subscript $(\cdot)_t$ denotes continuous time dependence while the subscript $(\cdot)_k$ denotes a discrete instance of time, t_k. The continuous system dynamics are assumed to be modeled using a noisy nonlinear system

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_t = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t^f), \tag{3.16}$$

where $\mathbf{w}_t^f \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_t^f)$ represents additive white Gaussian noise. We also assume that we have a set of noisy, discrete,, nonlinear measurements of the form:

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{w}_k^h), \tag{3.17}$$

where $\mathbf{w}_k^h \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_k^h)$ represents additive white Gaussian noise.

Let $\bar{\mathbf{x}}_t$ denote the current state estimate. The system dynamics, $f(\cdot)$, and the measurement model, $h(\cdot)$, can be linearized about the current estimate, $\bar{\mathbf{x}}_t$, to obtain the linear (continuous) dynamics matrix, $\mathbf{A}_t$, the dynamics noise matrix, $\mathbf{G}_t$, the linear (discrete) measurement matrix, $\mathbf{H}_k$, and the measurement noise matrix $\mathbf{N}_t$.

$$\begin{aligned} \mathbf{A}_t &= \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\mathbf{x}_t} & \mathbf{G}_t &= \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{w}^f}\Big|_{\mathbf{x}=\mathbf{x}_t} \\ \mathbf{H}_k &= \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\mathbf{x}_k} & \mathbf{N}_t &= \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{w}^h}\Big|_{\mathbf{x}=\mathbf{x}_t} \end{aligned} \tag{3.18}$$

During the prediction phase, the state estimate is propagated forward in time using the deterministic dynamics:

$$\frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{x}}_t = f(\bar{\mathbf{x}}_t, \mathbf{u}_t). \tag{3.19}$$

The state covariance matrix, $\mathbf{P}_t$ is computed using the following Riccati equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{P}_t = \mathbf{A}_t\mathbf{P}_t + \mathbf{P}_t\mathbf{A}_t^\mathsf{T} + \mathbf{G}_t\mathbf{\Sigma}_t^f\mathbf{G}_t^\mathsf{T}. \tag{3.20}$$

When a measurement is received, the state can be updated using

$$\bar{\mathbf{x}}_k^+ = \bar{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{y}_k - h(\bar{\mathbf{x}}_k^-) \right), \tag{3.21}$$

where the Kalman Gain, $\mathbf{K}$, is computed by

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^\mathsf{T} + \mathbf{N}_t \mathbf{\Sigma}_k^h \mathbf{N}_t^\mathsf{T} \qquad \mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^\mathsf{T} \mathbf{S}_k^{-1}. \tag{3.22}$$

The state covariance matrix is updated using

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-. \tag{3.23}$$

3.2.2 Discretization

The continuous dynamics can be discretized by assuming a zero-order hold on the inputs and integrating from t_k to t_{k+1}. This results in discrete dynamics that can be generally written as

$$\bar{\mathbf{x}}_{k+1} = f^d(\bar{\mathbf{x}}_k, \mathbf{u}_k). \tag{3.24}$$

In order to propagate the covariance, a continuous-time Riccati equation (3.20) needs to be solved. The analytical solution to that differential equation is given by [165]

$$\mathbf{P}_{t_{k+1}} = \mathbf{\Phi}(t_{k+1}, t_k) \mathbf{P}_{t_k} \mathbf{\Phi}(t_{k+1}, t_k)^\mathsf{T} + \mathbf{\Sigma}_t^{fd}, \tag{3.25}$$

where the discrete noise covariance matrix, $\mathbf{\Sigma}_t^{fd}$, is computed by

$$\mathbf{\Sigma}_t^{fd} = \int_{t_k}^{t_{k+1}} \mathbf{\Phi}(t_{k+1}, t) \mathbf{G}_t \mathbf{\Sigma}_t^f \mathbf{G}_t^\mathsf{T} \mathbf{\Phi}(t_{k+1}, t)^\mathsf{T} dt, \tag{3.26}$$

and the state transition matrix, $\mathbf{\Phi}(t_{k+1}, t_k)$, satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t} \mathbf{\Phi}(t, t_k) = \mathbf{A}_t \mathbf{\Phi}(t, t_k) \quad \text{with} \quad \mathbf{\Phi}(t_k, t_k) = \mathbf{I}. \tag{3.27}$$

3.3 Factor Graph based Smoothing

In contrast to filtering, most smoothing methods aim to estimate a discrete trajectory of the robot's state variables as opposed to only the current state. This trajectory of states (along with possible landmarks) is estimated by solving a nonlinear optimization problem that aims to find the set of states that best fits the entire history of noisy measurements.

Since previous states are not marginalized out (as in the extended Kalman filter (EKF)), loop closures can easily be incorporated. Furthermore, there are no commitments to previous (and potentially bad) linearizations, as the motion and measurement models can always be re-linearized around the current best estimate.

A factor graph is a bipartite graph that provides a convenient method for modeling the relationships between measurements and state variables. Once a factor graph is defined, inference is often performed by computing the *Maximum-A-Posteriori* (MAP) estimate, which maximizes the posteriori probability of the states (and landmarks) given the set of measurements [67].

Let $\mathcal{X} = \bigcup_{k \in \mathcal{K}_f} \mathbf{x}_k$ denote the set of all state (and landmark) variables, and $\mathcal{Z} = \bigcup_{k \in \mathcal{K}_h} \mathbf{y}_k$ denote the set of all measurements (as defined in Section 3.2). The Maximum-A-Posteriori (MAP) estimate of $\mathcal{X}$ can be computed by solving the following optimization problem:

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \; p(\mathcal{X}|\mathcal{Z}). \tag{3.28}$$

Using Bayes theorem, we can rewrite this maximization as:

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \; \frac{p(\mathcal{X})p(\mathcal{Z}|\mathcal{X})}{p(\mathcal{Z})} \quad \propto \quad \arg\max_{\mathcal{X}} \; p(\mathcal{X})p(\mathcal{Z}|\mathcal{X}) \tag{3.29}$$

where the constant denominator can be dropped since does not affect the solution of the optimization problem. In addition, since every input and measurement is assumed to be independent, the joint densities can be computed as a product of factors:

$$p(\mathcal{X}) = p(\mathbf{x}_0) \prod_{k \in \mathcal{K}_f} p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k), \quad \text{and} \quad p(\mathcal{Z}|\mathcal{X}) = \prod_{k \in \mathcal{K}_h} p(\mathbf{y}_k|\mathbf{x}_k), \tag{3.30}$$

where $p(\mathbf{x}_0)$ denotes the prior probability of the initial state. The multivariate Gaussian distributions can be computed from the motion model (3.24) and the measurement model (3.17).

$$\begin{aligned} p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) &= \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_k^{f_d}|}} \, e^{\left\{-\frac{1}{2}\|\mathbf{x}_{k+1} - f(\mathbf{x}_k, \mathbf{u}_k)\|^2_{\boldsymbol{\Sigma}_k^{f_d}}\right\}} \\ p(\mathbf{y}_k|\mathbf{x}_k) &= \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_k^{h}|}} \, e^{\left\{-\frac{1}{2}\|\mathbf{y}_k - h(\mathbf{x}_k)\|^2_{\boldsymbol{\Sigma}_k^{h}}\right\}} \end{aligned} \tag{3.31}$$

This posteriori maximization can be equivalently expressed as a minimization of the negative

log-likelihood:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \; -\log \; p(\mathcal{X}|\mathcal{Z}) \tag{3.32}$$

which can be written using as a sum of error "residuals" due to the factorization of the joint densities (3.30) and the zero-mean Gaussian noise assumptions (3.31). The resulting optimization problem becomes:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \; \|\mathbf{y}_0 - \mathbf{x}_0\|^2_{\mathbf{\Sigma}_0} + \sum_{k\in\mathcal{K}_f} \|\mathbf{x}_{k+1} - f(\mathbf{x}_k, \mathbf{u}_k)\|^2_{\mathbf{\Sigma}_k^{f_d}} + \sum_{k\in\mathcal{K}_h} \|\mathbf{y}_k - h(\mathbf{x}_k)\|^2_{\mathbf{\Sigma}_k^{h}} \tag{3.33}$$

where $\mathbf{y}_0$ and $\mathbf{\Sigma}_0$ are the prior mean and covariance associated with the initial state, and $\|\mathbf{e}\|^2_{\mathbf{\Sigma}} \triangleq \mathbf{e}^\mathsf{T}\mathbf{\Sigma}^{-1}\mathbf{e}$ is defined as the Mahalanobis distance squared.

This nonlinear weighted least-squares optimization problem can be solved using a variety of methods such as the gradient descent, Gauss-Newton, Levenberg-Marquardt, or Powell's Dogleg algorithms [68]. These algorithms generally work by iteratively linearizing the cost and taking appropriate steps towards a local minimum. The computational bottlenecks are often re-linearization and the factorization of the information matrix that is necessary for solving the approximated linear system at each time step [66].

Incremental methods, such as iSAM [136] and iSAM2 [137], allow the factorization of the information matrix to be updated incrementally. This reduces the computation time required to incorporate new measurements into the factor graph, allowing many problems to be solved in real-time. Periodic variable re-ordering, re-linearization, and batch solves are often used along-side these incremental methods to further improve state estimation [67].

Remark 2. Incorporating a new measurement model into the factor graph framework simply boils down to computing the measurement's error residuals and the associated covariance matrix.

3.4 Lie Groups

A *Lie group* is defined as a group that is also a smooth manifold [215, 104, 53]. In particular, we focus on matrix Lie groups, which appear throughout robotics and can be defined as any closed subgroup of the set of $n \times n$ invertible matrices. For the rest of this dissertation, any Lie groups are assumed to be matrix Lie groups.

3.4.1 Lie Algebra

Let $\mathcal{G}$ denote a particular matrix Lie group. The associated *Lie algebra*, denoted $\mathfrak{g}$, is defined as the tangent space at the identity element of the group, $\mathfrak{g} \triangleq T_e\mathcal{G}$. If elements of $\mathcal{G}$ are $n \times n$ matrices, then so are elements of $\mathfrak{g}$. The Lie algebra is a vector space and elements can be mapped back and forth between $\mathfrak{g}$ and $\mathbb{R}^{\dim\mathfrak{g}}$ using the following operators:

$$(\cdot)^\wedge : \mathbb{R}^{\dim\mathfrak{g}} \to \mathfrak{g} \quad \text{and} \quad (\cdot)^\vee : \mathfrak{g} \to \mathbb{R}^{\dim\mathfrak{g}}. \tag{3.34}$$

Therefore, the Lie algebra provides a natural vector space in which to define noise, errors, and covariance. This is heavily utilized when performing filtering, smoothing, or optimization on a manifold, as seen Section 3.4.6.

3.4.2 Exponential and Logarithm Maps

The *exponential map of the Lie group*, $\exp : \mathfrak{g} \to \mathcal{G}$, exactly maps an element of the Lie algebra to a corresponding point on the Lie group [104]. For matrix Lie groups, the exponential map is simply the matrix exponential. The *logarithm map of the Lie group* does the inverse, $\log : \mathcal{G} \to \mathfrak{g}$. It takes an element of the Lie group and maps it to a corresponding element in the Lie algebra. The relationship between the exponential and logarithm maps is given by:

$$\log(\exp(\boldsymbol{\xi}^\wedge)) = \boldsymbol{\xi}^\wedge, \tag{3.35}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{\dim\mathfrak{g}}$. To simplify notation, it is convenient to define "vectorized" exponential and logarithm maps, $\mathrm{Exp} : \mathbb{R}^{\dim\mathfrak{g}} \to \mathcal{G}$ and $\mathrm{Log} : \mathcal{G} \to \mathbb{R}^{\dim\mathfrak{g}}$;

$$\mathrm{Exp}(\boldsymbol{\xi}) = \exp(\boldsymbol{\xi}^\wedge) \quad \text{and} \quad \mathrm{Log}(\mathrm{Exp}(\boldsymbol{\xi})) = \boldsymbol{\xi}. \tag{3.36}$$

3.4.3 Adjoint Representation

The adjoint representation plays a key role in the theory of Lie groups and through this linear map we can capture the non-commutative structure of a Lie group.

Definition 1 (The Adjoint Map, see page 63 Hall [104])**.** Let $\mathcal{G}$ be a matrix Lie group with Lie algebra $\mathfrak{g}$. For any $\mathbf{X} \in \mathcal{G}$ the adjoint map, $\mathrm{Ad}_\mathbf{X} : \mathfrak{g} \to \mathfrak{g}$, is a linear map defined as $\mathrm{Ad}_\mathbf{X}(\boldsymbol{\xi}^\wedge) = \mathbf{X}\boldsymbol{\xi}^\wedge\mathbf{X}^{-1}$. Furthermore, we denote the matrix representation of the adjoint map by $\mathrm{Ad}_\mathbf{X}$.

The adjoint map allows us to linearly and exactly transform vectors from the tangent space about one group element to the (different) tangent space about another group element.

The matrix representation also provides an easy method for shifting an element of the group across the exponential map using the following relation:

$$\mathbf{X}\text{Exp}(\boldsymbol{\xi}) = \text{Exp}(\text{Ad}_{\mathbf{X}}\boldsymbol{\xi})\mathbf{X}. \tag{3.37}$$

3.4.4 Jacobians

The Jacobians of the group relate the derivative of the exponential coordinates of the Lie algebra to a generalized velocity of the corresponding Lie group element [21, 53]. Depending on the frame of measurement (body or spatial), the generalized velocity, $\boldsymbol{\varpi} \in \mathbb{R}^{\dim \mathfrak{g}}$, and the group Jacobian can be defined two ways. When defined in the body frame, the generalized velocity is defined as $\boldsymbol{\varpi}^{\wedge} \triangleq \mathbf{X}_t^{-1}\dot{\mathbf{X}}_t$, and the *right Jacobian* relates the derivative of the exponential coordinates, $\dot{\boldsymbol{\xi}}_t$ to this velocity,

$$\boldsymbol{\varpi} = \mathbf{J}_r(\boldsymbol{\xi})\dot{\boldsymbol{\xi}}_t. \tag{3.38}$$

When defined in the spatial frame, the generalized velocity is defined as $\boldsymbol{\varpi}^{\wedge} \triangleq \dot{\mathbf{X}}_t\,\mathbf{X}_t^{-1}$, and the *left Jacobian* relates the derivative of the exponential coordinates, $\dot{\boldsymbol{\xi}}_t$ to this velocity,

$$\boldsymbol{\varpi} = \mathbf{J}_l(\boldsymbol{\xi})\dot{\boldsymbol{\xi}}_t. \tag{3.39}$$

The left and right Jacobians are related through the group's adjoint map (4.19),

$$\mathbf{J}_l(\boldsymbol{\xi}) = \text{Ad}_{\text{Exp}(\boldsymbol{\xi})}\,\mathbf{J}_r(\boldsymbol{\xi}). \tag{3.40}$$

The right Jacobian, $\mathbf{J}_r$, is useful for relating additive increments in the tangent space to (right-hand-side) multiplicative increments on the manifold:

$$\text{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) \approx \text{Exp}(\boldsymbol{\phi})\text{Exp}(\mathbf{J}_r(\boldsymbol{\phi})\delta\boldsymbol{\phi}). \tag{3.41}$$

The *right Jacobian inverse*, $\mathbf{J}_r^{-1}$, can be used to provide a similar first-order approximation of the logarithm map:

$$\text{Log}(\text{Exp}(\boldsymbol{\phi})\text{Exp}(\delta\boldsymbol{\phi})) \approx \boldsymbol{\phi} + \mathbf{J}_r^{-1}(\boldsymbol{\phi})\delta\boldsymbol{\phi}. \tag{3.42}$$

Similarly, the left Jacobian, $\mathbf{J}_l$, is useful for relating additive increments in the tangent space to (left-hand-side) multiplicative increments on the manifold:

$$\text{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) \approx \text{Exp}(\mathbf{J}_r(\boldsymbol{\phi})\delta\boldsymbol{\phi})\text{Exp}(\boldsymbol{\phi}). \tag{3.43}$$

The *left Jacobian inverse*, $\mathbf{J}_l^{-1}$, can be used to provide a similar first-order approximation of the logarithm map:

$$\mathrm{Log}(\mathrm{Exp}(\delta\boldsymbol{\phi})\mathrm{Exp}(\boldsymbol{\phi})) \approx \boldsymbol{\phi} + \mathbf{J}_l^{-1}(\boldsymbol{\phi})\delta\boldsymbol{\phi}. \tag{3.44}$$

The Jacobian inverse can also be viewed as the first-order approximation of the Baker-Campbell-Hausdorff (BCH) formula [21]. The BCH formula relates addition of elements in the Lie algebra to multiplication of group elements.

Definition 2 (Baker-Campbell-Hausdorff (BCH) Formula)**.** Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors in the Lie algebra. Multiplication of the corresponding two group elements, $\mathrm{Exp}(\mathbf{x})$ and $\mathrm{Exp}(\mathbf{y})$, can be computed as an infinite sum of the vectors in the Lie algebra;

$$\begin{aligned}\mathbf{z} &= \mathrm{Log}\left(\mathrm{Exp}(\mathbf{x})\mathrm{Exp}(\mathbf{y})\right) \\ &= \mathbf{x} + \mathbf{y} + \frac{1}{2}[\mathbf{x},\mathbf{y}] + \frac{1}{12}\left([\mathbf{x},[\mathbf{x},\mathbf{y}]] + [\mathbf{y},[\mathbf{y},\mathbf{x}]]\right) - \frac{1}{24}[\mathbf{y},[\mathbf{x},[\mathbf{x},\mathbf{y}]]] + \cdots,\end{aligned} \tag{3.45}$$

where $[\cdot,\cdot]$ denotes the Lie bracket operation.

3.4.5 Useful Matrix Lie Groups

In this section we detail the Lie groups used throughout this dissertation, along with their algebras, exponential maps, logarithm maps, and adjoint representations. Each of these groups are a subgroup of the general linear group of degree n, denoted $\mathrm{GL}_n(\mathbb{R})$, which is the set of all $n \times n$ non-singular real matrices. The group binary operation is the standard matrix multiplication.

3.4.5.1 Group of Rotation Matricies, SO(3)

The three-dimensional (3D) special orthogonal group, defined by

$$\mathrm{SO}(3) = \left\{\mathbf{R} \in \mathrm{GL}_3(\mathbb{R}) \mid \mathbf{R}\mathbf{R}^\mathsf{T} = \mathbf{I}_3, \det\mathbf{R} = +1\right\}, \tag{3.46}$$

is the rotation group on $\mathbb{R}^3$. It can be used to represent the orientation of a 3D rigid body. The Lie algebra of SO(3), denoted by $\mathfrak{so}(3)$, is defined by the set of 3×3 skew-symmetric matrices such that for any $\boldsymbol{\phi} \triangleq \mathrm{vec}(\phi_1, \phi_2, \phi_3) \in \mathbb{R}^3$:

$$\boldsymbol{\phi}^\wedge \triangleq (\boldsymbol{\phi})_\times = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathfrak{so}(3). \tag{3.47}$$

If a skew-symmetric matrix is multiplied by a vector, one can swap the order using the following useful property:

$$(\mathbf{a})_\times \mathbf{b} = -(\mathbf{b})_\times \mathbf{a}. \tag{3.48}$$

In addition, if the skew operator is further applied to this quantity, it is possible to factor out the expression into two terms;

$$\big((\mathbf{a})_\times \mathbf{b}\big)_\times = (\mathbf{a})_\times (\mathbf{b})_\times - (\mathbf{b})_\times (\mathbf{a})_\times . \tag{3.49}$$

A closed-form expression for the exponential map of SO(3) is given by Rodrigues' rotation formula:

$$\mathrm{Exp}(\boldsymbol{\phi}) = \mathbf{I}_3 + \left(\frac{\sin\theta}{\theta}\right)(\boldsymbol{\phi})_\times + \left(\frac{1-\cos\theta}{\theta^2}\right)(\boldsymbol{\phi})_\times^2 \in \mathrm{SO}(3) \tag{3.50}$$

where $\theta \triangleq ||\boldsymbol{\phi}||$. The closed-form expression for the logarithm map is given by:

$$\mathrm{Log}(\mathbf{R}) = \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^\mathsf{T}) \in \mathbb{R}^3, \quad \text{with} \quad \theta = \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R})-1}{2}\right). \tag{3.51}$$

The matrix representation of the Adjoint map of SO(3) is simply the rotation element itself,

$$\mathrm{Ad}_\mathbf{R} = \mathbf{R} \in \mathbb{R}^{3\times 3}. \tag{3.52}$$

The right Jacobian of SO(3) is can be calculated using:

$$\mathbf{J}_r(\boldsymbol{\phi}) = \mathbf{I}_3 - \left(\frac{1-\cos\theta}{\theta^2}\right)(\boldsymbol{\phi})_\times + \left(\frac{\theta-\sin\theta}{\theta^3}\right)(\boldsymbol{\phi})_\times^2 \tag{3.53}$$

where, once again, $\theta \triangleq ||\boldsymbol{\phi}||$, [53, p. 40]. The right Jacobian inverse of SO(3) also has a closed-form expression given by:

$$\mathbf{J}_r^{-1}(\boldsymbol{\phi}) = \mathbf{I}_3 + \frac{1}{2}(\boldsymbol{\phi})_\times + \left(\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta}\right)(\boldsymbol{\phi})_\times^2 \tag{3.54}$$

The left Jacobian of SO(3) and its inverse can be computed using

$$\begin{aligned} \mathbf{J}_l(\boldsymbol{\phi}) &= \mathbf{I}_3 + \left(\frac{1-\cos\theta}{\theta^2}\right)(\boldsymbol{\phi})_\times + \left(\frac{\theta-\sin\theta}{\theta^3}\right)(\boldsymbol{\phi})_\times^2 \\ \mathbf{J}_l^{-1}(\boldsymbol{\phi}) &= \mathbf{I}_3 - \frac{1}{2}(\boldsymbol{\phi})_\times + \left(\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta}\right)(\boldsymbol{\phi})_\times^2 . \end{aligned} \tag{3.55}$$

Remark 3. In this dissertation, the skew-symmetric operator, $(\cdot)_\times$, is preferred over the *hat* operator, $(\cdot)^\wedge$, when mapping vectors to $\mathfrak{so}(3)$. This is done to prevent confusion and operator overload when working with other Lie algebras, such as $\mathfrak{se}(3)$.

3.4.5.2 Group of Direct Spatial Isometries, SE(3)

The 3D special Euclidean group, defined by

$$\mathrm{SE}(3) = \left\{ \mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \in \mathrm{GL}_4(\mathbb{R}) \mid \mathbf{R} \in \mathrm{SO}(3), \mathbf{p} \in \mathbb{R}^3 \right\} \tag{3.56}$$

is the group of rigid transformations on $\mathbb{R}^3$. It can be used to represent the pose (position and orientation) of a 3D rigid body. The Lie algebra of SE(3), denoted by $\mathfrak{se}(3)$, can be identified by 4×4 matrices such that for any $\boldsymbol{\xi} \triangleq \mathrm{vec}(\boldsymbol{\phi}, \boldsymbol{\rho}) \in \mathbb{R}^6$:

$$\boldsymbol{\xi}^\wedge \triangleq \begin{bmatrix} (\boldsymbol{\phi})_\times & \boldsymbol{\rho} \\ \mathbf{0}_{1\times 3} & 0 \end{bmatrix} \in \mathfrak{se}(3) \tag{3.57}$$

The exponential map of SE(3) can be computed using:

$$\mathrm{Exp}(\boldsymbol{\xi}) = \begin{bmatrix} \mathrm{Exp}(\boldsymbol{\phi}) & \mathbf{J}_l(\boldsymbol{\phi})\boldsymbol{\rho} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \in \mathrm{SE}(3), \tag{3.58}$$

where $\mathrm{Exp}(\boldsymbol{\phi})$ and $\mathbf{J}_l(\boldsymbol{\phi})$ are the exponential map and the left Jacobian of SO(3). The logarithm map of SE(3) is computed using:

$$\mathrm{Log}(\mathbf{H}) = \begin{bmatrix} \mathrm{Log}(\mathbf{R}) \\ \mathbf{J}_l^{-1}\left(\mathrm{Log}(\mathbf{R})\right)\mathbf{p} \end{bmatrix} \in \mathbb{R}^6, \tag{3.59}$$

where $\mathrm{Log}(\mathbf{R})$ is the logarithm map of SO(3). The matrix representation of the Adjoint map of SE(3) is given by:

$$\mathrm{Ad}_{\mathbf{H}} = \begin{bmatrix} \mathbf{R} & \mathbf{0}_{3\times 3} \\ (\mathbf{p})_\times \mathbf{R} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{3.60}$$

The left Jacobian of SE(3) given by [22, 21],

$$\begin{aligned}
\mathbf{J}_l(\boldsymbol{\xi}) &= \begin{bmatrix} \mathbf{J}_l(\boldsymbol{\phi}) & \mathbf{0}_{3\times 3} \\ \mathbf{Q}(\boldsymbol{\phi},\boldsymbol{\rho}) & \mathbf{J}_l(\boldsymbol{\phi}) \end{bmatrix} \\
\mathbf{Q}(\boldsymbol{\phi},\boldsymbol{\rho}) &\triangleq \frac{1}{2}(\boldsymbol{\rho})_\times + \frac{\theta-\sin(\theta)}{\theta^3}\left((\boldsymbol{\phi})_\times(\boldsymbol{\rho})_\times + (\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times + (\boldsymbol{\phi})_\times(\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times\right) \\
&\quad + \frac{\theta^2+2\cos(\theta)-2}{2\theta^4}\left((\boldsymbol{\phi})_\times^2(\boldsymbol{\rho})_\times + (\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times^2 - 3(\boldsymbol{\phi})_\times(\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times\right) \\
&\quad + \frac{4\theta+2\theta\cos(\theta)-6\sin(\theta)}{4\theta^5}((\boldsymbol{\phi})_\times(\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times^2 + (\boldsymbol{\phi})_\times^2(\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times),
\end{aligned} \tag{3.61}$$

where $\theta \triangleq ||\boldsymbol{\phi}||$, and $\mathbf{J}_l(\boldsymbol{\phi})$ is the left Jacobian of SO(3).

3.4.5.3 Group of K Direct Isometries, $\mathrm{SE}_K(3)$

The group of K direct isometries, defined by

$$\mathrm{SE}_K(3) = \left\{ \mathbf{X} = \begin{bmatrix} \mathbf{R} & \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_K \\ \mathbf{0}_{1\times 3} & 1 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & 0 & 0 & \cdots & 1 \end{bmatrix} \in \mathrm{GL}_{K+3}(\mathbb{R}) \mid \mathbf{R} \in \mathrm{SO}(3), \mathbf{p}_i \in \mathbb{R}^3 \right\}, \tag{3.62}$$

can be used to represent a 3D orientation along with K vectors in $\mathbb{R}^3$. This group will be useful for modeling the inertial measurement unit (IMU) state (pose plus velocity) along with other positions, such as contact points or landmarks. The Lie algebra of $\mathrm{SE}_K(3)$, denoted by $\mathfrak{se}_K(3)$, can be identified with $(K+3)\times(K+3)$ matrices such that for any $\boldsymbol{\xi} \triangleq \mathrm{vec}(\boldsymbol{\omega}, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2, \cdots, \boldsymbol{\rho}_K) \in \mathbb{R}^{3K+3}$:

$$\boldsymbol{\xi}^\wedge \triangleq \begin{bmatrix} (\boldsymbol{\omega})_\times & \boldsymbol{\rho}_1 & \boldsymbol{\rho}_2 & \cdots & \boldsymbol{\rho}_K \\ \mathbf{0}_{1\times 3} & 0 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathfrak{se}_K(3) \tag{3.63}$$

The exponential map of $\mathrm{SE}_K(3)$ can be computed using:

$$\mathrm{Exp}(\boldsymbol{\xi}) = \begin{bmatrix} \mathrm{Exp}(\boldsymbol{\phi}) & \mathbf{J}_l(\boldsymbol{\phi})\boldsymbol{\rho}_1 & \mathbf{J}_l(\boldsymbol{\phi})\boldsymbol{\rho}_2 & \cdots & \mathbf{J}_l(\boldsymbol{\phi})\boldsymbol{\rho}_K \\ \mathbf{0}_{1\times 3} & 1 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & 0 & 0 & \cdots & 1 \end{bmatrix} \in \mathrm{SE}_K(3), \tag{3.64}$$

where $\mathrm{Exp}(\boldsymbol{\phi})$ and $\mathbf{J}_t(\boldsymbol{\phi})$ are the exponential map and the left Jacobian of SO(3). The logarithm map of $\mathrm{SE}_K(3)$ is computed using:

$$\mathrm{Log}(\mathbf{X}) = \begin{bmatrix} \mathrm{Log}(\mathbf{R}) \\ \mathbf{J}_l^{-1}\left(\mathrm{Log}(\mathbf{R})\right)\mathbf{p}_1 \\ \mathbf{J}_l^{-1}\left(\mathrm{Log}(\mathbf{R})\right)\mathbf{p}_2 \\ \vdots \\ \mathbf{J}_l^{-1}\left(\mathrm{Log}(\mathbf{R})\right)\mathbf{p}_K \end{bmatrix} \in \mathbb{R}^{3K+3}, \tag{3.65}$$

where $\mathrm{Log}(\mathbf{R})$ is the logarithm map of SO(3). The matrix representation of the Adjoint map of $\mathrm{SE}_K(3)$ is given by:

$$\mathrm{Ad}_{\mathbf{X}} = \begin{bmatrix} \mathbf{R} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \cdots & \mathbf{0}_{3\times 3} \\ (\mathbf{p}_1)_\times \mathbf{R} & \mathbf{R} & \mathbf{0}_{3\times 3} & \cdots & \mathbf{0}_{3\times 3} \\ (\mathbf{p}_2)_\times \mathbf{R} & \mathbf{0}_{3\times 3} & \mathbf{R} & \cdots & \mathbf{0}_{3\times 3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (\mathbf{p}_K)_\times \mathbf{R} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \cdots & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{(3K+3)\times(3K+3)}. \tag{3.66}$$

The left Jacobian of $\mathrm{SE}_K(3)$ given by,

$$\begin{aligned}
\mathbf{J}_l(\boldsymbol{\xi}) &= \begin{bmatrix} \mathbf{J}_l(\boldsymbol{\phi}) & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \cdots & \mathbf{0}_{3\times3} \\ \mathbf{Q}(\boldsymbol{\phi},\boldsymbol{\rho}_1) & \mathbf{J}_l(\boldsymbol{\phi}) & \mathbf{0}_{3\times3} & \cdots & \mathbf{0}_{3\times3} \\ \mathbf{Q}(\boldsymbol{\phi},\boldsymbol{\rho}_2) & \mathbf{0}_{3\times3} & \mathbf{J}_l(\boldsymbol{\phi}) & \cdots & \mathbf{0}_{3\times3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}(\boldsymbol{\phi},\boldsymbol{\rho}_K) & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \cdots & \mathbf{J}_l(\boldsymbol{\phi}) \end{bmatrix} \\
\mathbf{Q}(\boldsymbol{\phi},\boldsymbol{\rho}) &\triangleq \frac{1}{2}(\boldsymbol{\rho})_\times + \frac{\theta-\sin(\theta)}{\theta^3}\left((\boldsymbol{\phi})_\times(\boldsymbol{\rho})_\times + (\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times + (\boldsymbol{\phi})_\times(\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times\right) \\
&+ \frac{\theta^2+2\cos(\theta)-2}{2\theta^4}\left((\boldsymbol{\phi})_\times^2(\boldsymbol{\rho})_\times + (\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times^2 - 3(\boldsymbol{\phi})_\times(\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times\right) \\
&+ \frac{4\theta+2\theta\cos(\theta)-6\sin(\theta)}{4\theta^5}((\boldsymbol{\phi})_\times(\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times^2 + (\boldsymbol{\phi})_\times^2(\boldsymbol{\rho})_\times(\boldsymbol{\phi})_\times),
\end{aligned} \tag{3.67}$$

where $\theta \triangleq ||\boldsymbol{\phi}||$, and $\mathbf{J}_l(\boldsymbol{\phi})$ is the left Jacobian of SO(3).

3.4.6 Uncertainty and Optimizations on Lie Groups

For optimization problems where the unknown variables are vectors, e.g.(3.33), simple subtraction can be used to define errors, $\mathbf{e}_k = \bar{\mathbf{x}}_k - \mathbf{x}_k$, while addition can be used to update the variables, i.e. $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} + \delta\mathbf{x}$. However, when the unknown variables live on manifolds, extra steps need to be taken. In this section, the key ideas are outlined through a simple example.

Lets assume that our state, $\mathbf{X} \in \mathcal{G}$, is a matrix lie group. The noisy system dynamics (3.24) can be rewritten as:

$$\mathbf{X}_{k+1} = f(\mathbf{X}_k, \mathbf{u}_k)\mathrm{Exp}(\mathbf{w}_k^{f_d}), \tag{3.68}$$

where $f(\cdot)$ is the discrete dynamics function and $\mathbf{w}_k^f \in \mathbb{R}^{\dim\mathfrak{g}}$ is zero-mean Gaussian noise with a covariance of $\boldsymbol{\Sigma}_k^{f_d}$. Since $f(\mathbf{X}_k, \mathbf{u}_k) \in \mathcal{G}$ is a matrix, the noise is multiplicative instead of additive.

First, the error residual that we aim to minimize is defined in the tangent space. The residual for the discrete dynamics (3.68) can be written as:

$$\mathbf{r}_{k+1} = \mathrm{Log}\left(f(\mathbf{X}_k, \mathbf{u}_k)^{-1}\mathbf{X}_{k+1}\right) \in \mathbb{R}^{\dim\mathfrak{g}}. \tag{3.69}$$

Next, a bijective *retraction*, $R_X : T_{\mathbf{X}}\mathcal{G} \mapsto \mathcal{G}$, is defined that maps between an element of the tangent space, $\delta\mathbf{x}$, and a neighborhood of $\mathbf{X} \in \mathcal{G}$ [2]. A natural choice of a retraction map for Lie groups is the exponential map,

$$R_X(\delta\mathbf{x}) \triangleq \mathbf{X}\mathrm{Exp}(\delta\mathbf{x}) \quad \text{or} \quad R_X(\delta\mathbf{x}) \triangleq \mathrm{Exp}(\delta\mathbf{x})\mathbf{X}. \tag{3.70}$$

The optimization problem can now be *lifted* using this retraction,

$$\underset{\mathbf{X}}{\text{minimize}} \quad f(\mathbf{X}) \quad \rightarrow \quad \underset{\delta\mathbf{x}\in\mathbb{R}^{\dim\mathfrak{g}}}{\text{minimize}} \quad f(R_X(\delta\mathbf{x})), \tag{3.71}$$

and the re-parameterized problem can be iteratively solved using standard optimization techniques [85].

In our example, let $\mathcal{X} \triangleq \bigcup_{k\in\mathcal{K}_f} \mathbf{X}_k$ denote the set of all unknown states. The original optimization problem involves minimizing the sum of the squared norm of all error residuals:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{k\in\mathcal{K}_f} \|\text{Log}\left(f(\mathbf{X}_k, \mathbf{u}_k)^{-1}\mathbf{X}_{k+1}\right)\|^2_{\mathbf{\Sigma}_k^{f_d}} \tag{3.72}$$

This optimization problem can be solved by iteratively lifting the problem to the tangent space, solving the re-parameterized problem, then mapping the updated solution back to the manifold using the retraction map. Using the exponential map as the retraction, the re-parameterized optimization problem becomes:

$$\delta^* = \arg\min_{\delta} \sum_{k\in\mathcal{K}_f} \|\text{Log}\left(f(\mathbf{X}_k\text{Exp}(\delta\mathbf{x}_k), \mathbf{u}_k)^{-1}\mathbf{X}_{k+1}\text{Exp}(\delta\mathbf{x}_{k+1})\right)\|^2_{\mathbf{\Sigma}_k^{f_d}} \tag{3.73}$$

where $\delta \triangleq \bigcup_{k\in\mathcal{K}_f} \delta\mathbf{x}_k$ is the set of all increments in the tangent space. After each iteration of the solver, the increments get mapped back to the Lie group, $\forall k \in \mathcal{K}_f,\ \mathbf{X}_k \leftarrow \mathbf{X}_k\text{Exp}(\delta\mathbf{x}_k)$. This lift-solve-retract process is repeated until a local minimum is reached. Refer to [2, 1, 206] for more information about performing optimization on manifolds.

3.5 Forward Kinematics

Forward kinematics refers to the process of computing the relative pose transformation between two frames of a multi-link system. Each individual joint displacement describes how the child link moves with respect to the parent one. This joint displacement can either be an angle (revolute joints) or a distance (prismatic joints).

Let $\boldsymbol{\alpha} \in \mathbb{R}^N$ denote the vector of joint displacements for a general robot. Without loss of generality, we define a *base* and an *end-effector* frame on the robot, donated B and E respectively. The homogeneous transformation between the base frame and the end-effector frame is defined by:

$$\mathbf{H}_{\text{BE}}(\boldsymbol{\alpha}) \triangleq \begin{bmatrix} \mathbf{R}_{\text{BE}}(\boldsymbol{\alpha}) & {}_{\text{B}}\mathbf{p}_{\text{BE}}(\boldsymbol{\alpha}) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{3.74}$$

where $\mathbf{R}_{\mathrm{BE}}(\boldsymbol{\alpha})$ and ${}_{\mathrm{B}}\mathbf{p}_{\mathrm{BE}}(\boldsymbol{\alpha})$ denote the relative orientation and position of the end-effector frame with respect to the base frame. This homogeneous transformation defines the *forward kinematics* function and can be constructed from the product of all relative joint transformations between the base (i) and end-effector (j) frames:

$$\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha}) = \prod_{k=i}^{j-1} \mathbf{H}_{\mathrm{k,k+1}}(\boldsymbol{\alpha}). \tag{3.75}$$

The *manipulator Jacobian*[1], $\mathbf{J}(\boldsymbol{\alpha}) \triangleq \mathrm{vec}(\mathbf{J}^{\omega}(\boldsymbol{\alpha}), \mathbf{J}^{v}(\boldsymbol{\alpha}))$, relates joint displacement rates to rotational and translational velocities of the end-effector. Together, these velocities form a generalized velocity vector, $\boldsymbol{\varpi} \triangleq \mathrm{vec}(\boldsymbol{\omega}, \mathbf{v}) \in \mathfrak{se}(3)$ [171]. The manipulator Jacobian can either be a *spatial* Jacobian, where the resulting velocities are measured in the first frame (base here),

$${}_{\mathrm{B}}\boldsymbol{\varpi}_{\mathrm{BE}} = \begin{bmatrix} {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{BE}} \\ {}_{\mathrm{B}}\mathbf{v}_{\mathrm{E}} \end{bmatrix} = {}_{\mathrm{B}}\mathbf{J}_{\mathrm{BE}}(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}}, \tag{3.76}$$

or a *body* Jacobian, where the velocities are measured in the second frame (end-effector here),

$${}_{\mathrm{E}}\boldsymbol{\varpi}_{\mathrm{BE}} = \begin{bmatrix} {}_{\mathrm{E}}\boldsymbol{\omega}_{\mathrm{BE}} \\ {}_{\mathrm{E}}\mathbf{v}_{\mathrm{E}} \end{bmatrix} = {}_{\mathrm{E}}\mathbf{J}_{\mathrm{BE}}(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}}. \tag{3.77}$$

As noted in Section 3.4.4, the frame of measurement for the twist specifies which side it appears on when looking at the time derivative of the homogeneous transformation;

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha}) &= ({}_{\mathrm{B}}\boldsymbol{\xi}_{\mathrm{BE}})^{\wedge}\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha}) = \big({}_{\mathrm{B}}\mathbf{J}_{\mathrm{BE}}(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}}\big)^{\wedge}\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha}) \quad \text{or} \\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha}) &= \mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha})({}_{\mathrm{E}}\boldsymbol{\xi}_{\mathrm{BE}})^{\wedge} = \mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha})\big({}_{\mathrm{E}}\mathbf{J}_{\mathrm{BE}}(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}}\big)^{\wedge}. \end{aligned} \tag{3.78}$$

These spatial and body Jacobians are related through the adjoint map of SE(3) (3.60),

$${}_{\mathrm{B}}\mathbf{J}_{\mathrm{BE}}(\boldsymbol{\alpha}) = \mathrm{Ad}_{\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha})}\, {}_{\mathrm{E}}\mathbf{J}_{\mathrm{BE}}(\boldsymbol{\alpha}). \tag{3.79}$$

Sometimes, it is useful to define a Jacobian for the position kinematics only. Starting

[1] also called the geometric Jacobian

with (3.78), we can express the pose derivative in matrix form,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha}) &= \left({}_{\mathrm{B}}\mathbf{J}_{\mathrm{BE}}(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}}\right)^{\wedge}\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha}) \\ &= \begin{bmatrix}\left({}_{\mathrm{B}}\mathbf{J}^{\omega}_{\mathrm{BE}}\dot{\boldsymbol{\alpha}}\right)_{\times} & {}_{\mathrm{B}}\mathbf{J}^{v}_{\mathrm{BE}}\dot{\boldsymbol{\alpha}} \\ \mathbf{0}_{1\times 3} & 0\end{bmatrix}\begin{bmatrix}\mathbf{R}_{\mathrm{BE}} & {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BE}} \\ \mathbf{0}_{1\times 3} & 1\end{bmatrix} \\ &= \begin{bmatrix}\left({}_{\mathrm{B}}\mathbf{J}^{\omega}_{\mathrm{BE}}\dot{\boldsymbol{\alpha}}\right)_{\times}\mathbf{R}_{\mathrm{BE}} & \left({}_{\mathrm{B}}\mathbf{J}^{\omega}_{\mathrm{BE}}\dot{\boldsymbol{\alpha}}\right)_{\times}{}_{\mathrm{B}}\mathbf{p}_{\mathrm{BE}} + {}_{\mathrm{B}}\mathbf{J}^{v}_{\mathrm{BE}}\dot{\boldsymbol{\alpha}} \\ \mathbf{0}_{1\times 3} & 0\end{bmatrix} \\ &= \begin{bmatrix}\left({}_{\mathrm{B}}\mathbf{J}^{\omega}_{\mathrm{BE}}\dot{\boldsymbol{\alpha}}\right)_{\times}\mathbf{R}_{\mathrm{BE}} & \left({}_{\mathrm{B}}\mathbf{J}^{v}_{\mathrm{BE}} - \left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{BE}}\right)_{\times}{}_{\mathrm{B}}\mathbf{J}^{\omega}_{\mathrm{BE}}\right)\dot{\boldsymbol{\alpha}} \\ \mathbf{0}_{1\times 3} & 0\end{bmatrix}.\end{aligned} \tag{3.80}$$

Extracting out just the position term, and using (3.48) to bring $\dot{\boldsymbol{\alpha}}$ outside the skew operator gives

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}{}_{\mathrm{B}}\mathbf{p}_{\mathrm{BE}} &= \left({}_{\mathrm{B}}\mathbf{J}^{v}_{\mathrm{BE}} - \left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{BE}}\right)_{\times}{}_{\mathrm{B}}\mathbf{J}^{\omega}_{\mathrm{BE}}\right)\dot{\boldsymbol{\alpha}} \\ &\triangleq {}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{BE}}(\boldsymbol{\alpha})\dot{\boldsymbol{\alpha}},\end{aligned} \tag{3.81}$$

where the Jacobian ${}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{BE}}$ relates joint displacement rates to the derivative of the position directly. This Jacobian can alternatively be computed using

$${}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{BE}}(\boldsymbol{\alpha}) = \frac{\partial {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BE}}}{\partial \boldsymbol{\alpha}}. \tag{3.82}$$

These Jacobians are useful for making first-order approximations of how encoder noise gets propagated through forward kinematics. Assume $\delta\boldsymbol{\alpha}$ is a small perturbation to the joint displacements. This term can be factored out of the forward kinematics using the various definitions of the manipulator Jacobians. The full pose approximation is given by

$$\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha}+\delta\boldsymbol{\alpha}) \approx \mathrm{Exp}\left({}_{\mathrm{B}}\mathbf{J}_{\mathrm{BE}}(\boldsymbol{\alpha})\delta\boldsymbol{\alpha}\right)\mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha}) = \mathbf{H}_{\mathrm{BE}}(\boldsymbol{\alpha})\mathrm{Exp}\left({}_{\mathrm{E}}\mathbf{J}_{\mathrm{BE}}(\boldsymbol{\alpha})\delta\boldsymbol{\alpha}\right), \tag{3.83}$$

whereas the position only approximation is given body

$${}_{\mathrm{B}}\mathbf{p}_{\mathrm{BE}}(\boldsymbol{\alpha}+\delta\boldsymbol{\alpha}) \approx {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BE}}(\boldsymbol{\alpha}) + {}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{BE}}(\boldsymbol{\alpha})\delta\boldsymbol{\alpha}. \tag{3.84}$$

Refer to Murray [171] and Spong et al. [211] for more information about kinematics and the manipulator Jacobians.

CHAPTER 4

Contact-Aided Invariant Extended Kalman Filtering

4.1 Overview

The core content in this chapter was previously published in [109] with co-authors Maani Ghaffari Jadidi, Jessy W Grizzle, and Ryan M. Eustice. Several extensions are made, however, most text and results remain unchanged.

4.1.1 Introduction and Objective

Legged robots have the potential to transform the logistics and package delivery industries, become assistants in our homes, and aide in search and rescue [55]. In order to develop motion planning algorithms and robust feedback controllers for these tasks, accurate estimates of the robot's state are needed. Some states, such as joint angles, can be directly measured using encoders, while other states, such as the robot's pose and velocity, require additional sensors. Most legged robots are equipped with an inertial measurement unit (IMU) that can measure linear acceleration and angular velocity, albeit with noise and bias perturbations. Consequently, nonlinear observers are typically used to fuse leg odometry and inertial measurements to infer trajectory, velocity, and calibration parameters [196, 27, 78, 151]. In view of a practical solution, designing a globally convergent observer is sacrificed for one with at best local properties, such as the extended Kalman filter (EKF) [103, 146, 222, 207]. This EKF-based approach is computationally efficient and easily customizable, allowing successful implementation on a number of legged robots with rigorous real-time performance requirements [35, 33, 81, 177].

Accurate pose estimation can be combined with visual data to build maps of the environment [82]. Then such maps can be used in gait selection to improve the stability of a robot while walking on uneven terrains and as a basis for high-level motion planning. Although

there have been many recent advancements in visual-inertial-odometry and simultaneous localization and mapping (SLAM) [186, 129, 83], these algorithms often rely on visual data for pose estimation. This means that the observer (and ultimately the feedback controller) can be adversely affected by rapid changes in lighting as well as the operating environment. It is therefore beneficial to develop a low-level state estimator that fuses data only from proprioceptive sensors to form accurate high-frequency state estimates. This approach was taken by Bloesch et al. [35] when developing a quaternion-based extended Kalman filter (QEKF) that combines inertial, contact, and kinematic data to estimate the robot's base pose, velocity, and a number of contact states. In this article, we expand upon these ideas to develop an invariant extended Kalman filter (InEKF) that has improved convergence and consistency properties allowing for a more robust observer that is suitable for long-term autonomy.

The theory of invariant observer design is based on the estimation error being invariant under the action of a matrix Lie group [3, 38], which has recently led to the development of the InEKF[1] [37, 23, 24, 25] with successful applications and promising results in simultaneous localization and mapping [23, 243] and aided inertial navigation systems [18, 19, 23, 237]. The invariance of the estimation error with respect to a Lie group action is referred to as a symmetry of the system [19]. Summarized briefly, Barrau and Bonnabel [24] showed that if the state is defined on a Lie group, and the dynamics satisfy a particular "group affine" property, then the symmetry leads to the estimation error satisfying a "log-linear" autonomous differential equation on the Lie algebra. In the deterministic case, this linear system can be used to exactly recover the estimated state of the nonlinear system as it evolves on the group. The log-linear property therefore allows the design of a nonlinear observer or state estimator with strong convergence properties.

4.1.1.1 Contribution

In this chapter, we develop a contact-aided invariant extended Kalman filter (InEKF) using the theory of Lie groups and invariant observer design. This filter combines an contact-inertial dynamics model with forward kinematic corrections to estimate the base pose and velocity along with all current contact points. We show that the error dynamics of this contact-inertial system follow a log-linear autonomous differential equation; hence, the observable state variables can be rendered convergent with a domain of attraction that is independent of the system's trajectory. Therefore unlike the standard EKF, both the linearized error dynamics and the linearized forward kinematic measurement model do not depend on current state estimate, which leads to these improved convergence properties.

[1] We use the InEKF acronym to distinguish from an iterated-EKF (IEKF).

We further demonstrate how to augment the state with IMU biases, as the online estimation of these parameters can have a crucial impact on system performance. The expressions for both the right-invariant and left-invariant error dynamics are given along with both a world-centric and robo-centric estimator formulation. We compare the convergence of the proposed InEKF with the commonly used quaternion-based EKF though both simulations and experiments on a Cassie-series bipedal robot. The accuracy of the filter is demonstrated using motion capture, while a LiDAR mapping experiment provides a practical use case. In addition, an open-source C++ library implementing the developed filter is provided. Overall, the developed contact-aided InEKF provides better performance in comparison with the quaternion-based EKF as a result of exploiting symmetries present in the system dynamics.

In summary, this work has the following contributions:

1. Derivation of a continuous-time right-invariant EKF for an IMU/contact process model with a forward kinematic measurement model; the observability analysis is also presented;
2. State augmentation with IMU biases;
3. Evaluations of the derived observer in simulation and hardware experiments using a 3D bipedal robot;
4. Alternative derivation of the observer using a left-invariant error definition;
5. Detailed explanation of the connection between the invariant error choice and the world-centric and robo-centric estimator formulations;
6. Equations provided for analytical discretization of the proposed observers; and
7. Development of an open-source C++ library for aided-inertial navigation using the InEKF `https://github.com/RossHartley/invariant-ekf`.

4.1.1.2 Outline

The remainder of this chapter is organized as follows. Section 4.1.2 provides the necessary preliminary material needed for understanding the InEKF formulation, which is motived by an attitude dynamics example in Section 4.1.3. Section 4.2 provides the derivation of a right invariant extended Kalman filter (RIEKF) for contact-inertial navigation with a right-invariant forward kinematic measurement model. In Section 4.3 we present simulation results comparing the convergence properties to a state-of-the-art QEKF. Section 4.4 discusses the state augmentation of the previously derived InEKF with IMU bias. The

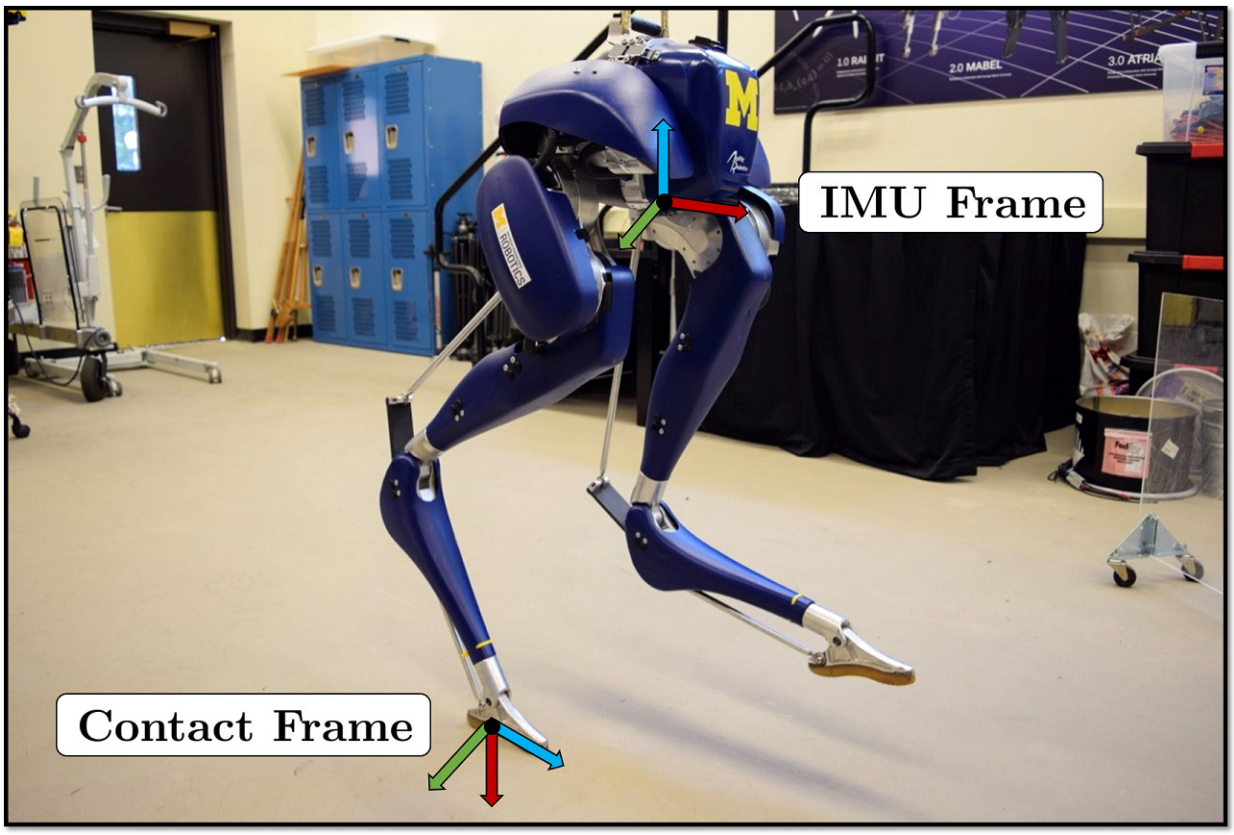


Figure 4.1: A Cassie-series biped robot is used for both simulation and experimental results. The robot was developed by Agility Robotics and has 20 degrees of freedom, 10 actuators, joint encoders, and an inertial measurement unit (IMU). The contact and IMU frames used in this work are depicted above.

consequences of the adding and removing contact points in the estimator are described in Section 4.5. Experimental evaluations on a 3D biped robot are presented in Section 4.6 along with an application towards LiDAR-based terrain mapping. Section 4.7 provides an alternative derivation of the proposed observer using the left-invariant dynamics along with an explanation of how to easily switch between the two formulations. Section 4.8 details how these equations can be modified to create a "robot-centric" estimator. Section 4.9 itemizes additional sensor measurements that fit within the InEKF framework and describes the relation between the developed filter and landmark-based SLAM. Finally, Section 4.10 gives details about discretization.

4.1.2 Preliminaries on Invariant Filtering

4.1.2.1 Autonomous Error Equations

Let $\mathbf{X}_t \in \mathcal{G}$ be a $N \times N$ matrix Lie group denoting the state at time t. A review of Lie groups and our adopted notation is provided in Section 3.4. The process dynamics evolving

on the Lie group can be defined by:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{X}_t = f_{u_t}(\mathbf{X}_t), \tag{4.1}$$

where u_t denotes the system inputs. The estimate of the state at time t is denoted by $\bar{\mathbf{X}}_t$. The state estimation error is invariant to the action of the Lie group and is defined using right or left multiplication of $\mathbf{X}_t^{-1}$ as follows.

Definition 3 (Left and Right Invariant Error)**.** The right- and left-invariant errors between two trajectories $\mathbf{X}_t \in \mathcal{G}$ and $\bar{\mathbf{X}}_t \in \mathcal{G}$ are:

$$\begin{aligned} \boldsymbol{\eta}_t^r &= \bar{\mathbf{X}}_t \mathbf{X}_t^{-1} = (\bar{\mathbf{X}}_t \mathbf{L})(\mathbf{X}_t \mathbf{L})^{-1} \quad \text{(Right-Invariant)} \\ \boldsymbol{\eta}_t^l &= \mathbf{X}_t^{-1} \bar{\mathbf{X}}_t = (\mathbf{L}\bar{\mathbf{X}}_t)^{-1}(\mathbf{L}\mathbf{X}_t), \quad \text{(Left-Invariant)} \end{aligned} \tag{4.2}$$

where $\mathbf{L} \in \mathcal{G}$ is an arbitrary element of the group.

The following two theorems are the fundamental results for deriving an InEKF and show that by correct parametrization of the error variable, a wide range of nonlinear problems can lead to linear error equations.

Theorem 1 (Autonomous Error Dynamics [24])**.** *A system is group affine if the dynamics, $f_{u_t}(\cdot)$, satisfies:*

$$f_{u_t}(\mathbf{X}_1\mathbf{X}_2) = f_{u_t}(\mathbf{X}_1)\mathbf{X}_2 + \mathbf{X}_1 f_{u_t}(\mathbf{X}_2) - \mathbf{X}_1 f_{u_t}(\mathbf{I}_d)\mathbf{X}_2 \tag{4.3}$$

for all $t > 0$ and $\boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathcal{G}$. Furthermore, if this condition is satisfied, the right- and left-invariant error dynamics are trajectory independent and satisfy:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\eta}_t^r &= g_{u_t}(\boldsymbol{\eta}_t^r) \quad \textit{where} \quad g_{u_t}(\boldsymbol{\eta}^r) = f_{u_t}(\boldsymbol{\eta}^r) - \boldsymbol{\eta}^r f_{u_t}(\mathbf{I}_d) \\ \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\eta}_t^l &= g_{u_t}(\boldsymbol{\eta}_t^l) \quad \textit{where} \quad g_{u_t}(\boldsymbol{\eta}^l) = f_{u_t}(\boldsymbol{\eta}^l) - f_{u_t}(\mathbf{I}_d)\boldsymbol{\eta}^l \end{aligned}$$

In the above, $\mathbf{I}_d \in \mathcal{G}$ denotes the group identity element; to avoid confusion, we use $\mathbf{I}_n$ for the $n \times n$ case.

In the following, for simplicity, we will use only the right-invariant error dynamics. Define

$\mathbf{A}_t$ to be a $\dim\mathfrak{g} \times \dim\mathfrak{g}$ matrix satisfying [2]

$$g_{u_t}(\mathrm{Exp}(\boldsymbol{\xi})) \triangleq (\mathbf{A}_t\boldsymbol{\xi})^\wedge + \mathcal{O}(\|\boldsymbol{\xi}\|^2). \tag{4.4}$$

For all $t \geq 0$, let $\boldsymbol{\xi}_t$ be the solution of the linear differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi}_t = \mathbf{A}_t\boldsymbol{\xi}_t. \tag{4.5}$$

Theorem 2 (Log-Linear Property of the Error [24])**.** *Consider the right-invariant error, $\boldsymbol{\eta}_t$, between two trajectories (possibly far apart). For arbitrary initial error $\boldsymbol{\xi}_0 \in \mathbb{R}^{\dim\mathfrak{g}}$, if $\boldsymbol{\eta}_0 = \mathrm{Exp}(\boldsymbol{\xi}_0)$, then for all $t \geq 0$,*

$$\boldsymbol{\eta}_t = \exp(\boldsymbol{\xi}_t); \tag{4.6}$$

that is, the nonlinear estimation error $\boldsymbol{\eta}_t$ can be exactly recovered from the time-varying linear differential equation (4.5)*.*

This theorem states that (4.5) is not the typical Jacobian linearization along a trajectory because the (left- or) right-invariant error on the Lie group can be exactly recovered from its solution. This result is of major importance for the propagation (prediction) step of the InEKF [24].

Remark 4. This indirect way of expressing the Jacobian of g_{u_t} is from Barrau and Bonnabel [24]; it is used because we are working with a matrix Lie group viewed as an embedded submanifold of a set of $n \times n$ matrices.

4.1.2.2 Autonomous Innovation Equations

The above section described how dynamics satisfying the group affine property (4.3) leads to autonomous (and log-linear) error equations. A similar property holds for innovation equations. If the measurement model fits one of the following forms:

$$\mathbf{Y}_t = \mathbf{X}_t\mathbf{b} + \mathbf{V}_t \quad \text{(Left-Invariant Observation)} \quad \text{or} \tag{4.7}$$

$$\mathbf{Y}_t = \mathbf{X}_t^{-1}\mathbf{b} + \mathbf{V}_t \quad \text{(Right-Invariant Observation)}, \tag{4.8}$$

where $\mathbf{b} \in \mathbb{R}^N$ denotes a constant vector and $\mathbf{V}_t$ is a vector of Gaussian noise, then the innovation for the right/left-invariant error is autonomous and given by $\bar{\mathbf{X}}_t\mathbf{Y}_t$ and $\bar{\mathbf{X}}_t^{-1}\mathbf{Y}_t$

[2] This indirect way of expressing the Jacobian of g_{u_t} is from [24]; it is used because we are working with a matrix Lie group viewed as an embedded submanifold of a set of $n \times n$ matrices.

respectively [24]. The update equations now take the following form:

$$\bar{\mathbf{X}}_t^+ = \bar{\mathbf{X}}_t \exp\left(\mathbf{L}_t\left(\bar{\mathbf{X}}_t^{-1}\mathbf{Y}_t - \mathbf{b}\right)\right) \quad \text{(Left-Invariant Observation)} \tag{4.9}$$

$$\bar{\mathbf{X}}_t^+ = \exp\left(\mathbf{L}_t\left(\bar{\mathbf{X}}_t\mathbf{Y}_t - \mathbf{b}\right)\right)\bar{\mathbf{X}}_t \quad \text{(Right-Invariant Observation)} \tag{4.10}$$

$$\tag{4.11}$$

where $\mathbf{L}_t : \mathbb{R}^N \mapsto \mathbb{R}^{\dim \mathfrak{g}}$ is a gain function that is computed through error linearizations. For the InEKF, $\mathbf{L}_t$ is essentially the Kalman gain (see Section 3.2). For more details on the material discussed above, we refer the reader to Barrau [23], Barrau and Bonnabel [24, 25].

4.1.3 A motivating example: 3D orientation propagation

Suppose we are interested in estimating the 3D orientation of a rigid body given angular velocity measurements in the body frame, $\tilde{\boldsymbol{\omega}}_t \triangleq \text{vec}(\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$. This type of measurement can be easily obtained from a gyroscope.

There are several different parameterizations of SO(3); Euler angles, quaternions, and rotation matrices being the most common. If we let $\mathbf{q}_t \triangleq \text{vec}(q_x, q_y, q_z)$ be a vector of Euler angles using the $R_z R_y R_x$ convention, then the orientation dynamics can be expressed as [**?**]

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} 1 & \sin(q_x)\tan(q_y) & \cos(q_x)\tan(q_y) \\ 0 & \cos(q_x) & -\sin(q_x) \\ 0 & \sin(q_x)\sec(q_y) & \cos(q_x)\sec(q_y) \end{bmatrix}\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}. \tag{4.12}$$

Let $\delta\mathbf{q}_t \triangleq \mathbf{q}_t - \bar{\mathbf{q}}_t \in \mathbb{R}^3$ be the error between the true and estimated Euler angles. The error dynamics can be written as a nonlinear function of the error variable, the inputs, and the state

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta\mathbf{q}_t = g(\delta\mathbf{q}_t, \tilde{\boldsymbol{\omega}}_t, \mathbf{q}_t). \tag{4.13}$$

In order to propagate the covariance in an EKF, we need to linearize the error dynamics at the current state estimate, $\mathbf{q}_t = \bar{\mathbf{q}}_t$ (i.e. zero error). This leads to a linear error dynamics of the form:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\delta\mathbf{q}_t &\approx \begin{bmatrix} 0 & (\omega_z\bar{c}_x + \omega_y\bar{s}_x)/\bar{c}_y^2 & \bar{t}_y(\omega_y\bar{c}_x - \omega_z\bar{s}_x) \\ 0 & 0 & \omega_z\bar{c}_x + \omega_y\bar{s}_x \\ 0 & (\bar{s}_y(\omega_z\bar{c}_x + \omega_y\bar{s}_x))/\bar{c}_y^2 & (\omega_y\bar{c}_x - \omega_z\bar{s}_x)/\bar{c}_y \end{bmatrix}\delta\mathbf{q}_t \\ &\triangleq \mathbf{A}(\tilde{\boldsymbol{\omega}}_t, \bar{\mathbf{q}}_t)\delta\mathbf{q}_t,\end{aligned} \tag{4.14}$$

where $\bar{c}_x$, $\bar{s}_x$, and $\bar{t}_x$ are shorthand for $\cos(\bar{q}_x)$, $\sin(\bar{q}_x)$, and $\tan(\bar{q}_x)$. The linear dynamics matrix clearly depends on the estimated angles, therefore bad estimates will affect the accuracy of the linearization and ultimately the performance and consistency of the filter.

Now instead, lets use a rotation matrix to represent the 3D orientation, $\mathbf{R}_t \in \mathrm{SO}(3)$. The dynamics can now be simply expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_t = \mathbf{R}_t \left(\tilde{\boldsymbol{\omega}}_t\right)_\times, \tag{4.15}$$

where $(\cdot)_\times$ denotes a 3×3 skew-symmetric matrix. If we define the error between the true and estimated orientation as $\boldsymbol{\eta}_t \triangleq \mathbf{R}_t^\mathsf{T} \bar{\mathbf{R}}_t \in \mathrm{SO}(3)$, then the (left-invariant) error dynamics becomes

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\eta}_t &= \mathbf{R}_t^\mathsf{T}\bar{\mathbf{R}}_t \left(\tilde{\boldsymbol{\omega}}_t\right)_\times + \left(\mathbf{R}_t \left(\tilde{\boldsymbol{\omega}}_t\right)_\times\right)^\mathsf{T} \bar{\mathbf{R}}_t \\
&= \mathbf{R}_t^\mathsf{T}\bar{\mathbf{R}}_t \left(\tilde{\boldsymbol{\omega}}_t\right)_\times - \left(\tilde{\boldsymbol{\omega}}_t\right)_\times \mathbf{R}_t^\mathsf{T}\bar{\mathbf{R}}_t \\
&= \boldsymbol{\eta}_t \left(\tilde{\boldsymbol{\omega}}_t\right)_\times - \left(\tilde{\boldsymbol{\omega}}_t\right)_\times \boldsymbol{\eta}_t \\
&= g(\boldsymbol{\eta}_t, \tilde{\boldsymbol{\omega}}_t).
\end{aligned} \tag{4.16}$$

Using this particular choice of state and error variable yields an autonomous error dynamics function (independent of the state directly). Since, SO(3) is a Lie Group, we can look at the dynamics of a redefined error that resides in the tangent space, $\boldsymbol{\eta}_t \triangleq \mathrm{Exp}(\boldsymbol{\xi}_t)$.

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{Exp}(\boldsymbol{\xi}_t)) &= \mathrm{Exp}(\boldsymbol{\xi}_t)\left(\tilde{\boldsymbol{\omega}}_t\right)_\times - \left(\tilde{\boldsymbol{\omega}}_t\right)_\times \mathrm{Exp}(\boldsymbol{\xi}_t) \\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{I} + \left(\boldsymbol{\xi}_t\right)_\times\right) &\approx \left(\mathbf{I} + \left(\boldsymbol{\xi}_t\right)_\times\right)\left(\tilde{\boldsymbol{\omega}}_t\right)_\times - \left(\tilde{\boldsymbol{\omega}}_t\right)_\times \left(\mathbf{I} + \left(\boldsymbol{\xi}_t\right)_\times\right) \\
\implies \frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{\xi}_t\right)_\times &= \left(\boldsymbol{\xi}_t\right)_\times \left(\tilde{\boldsymbol{\omega}}_t\right)_\times - \left(\tilde{\boldsymbol{\omega}}_t\right)_\times \left(\boldsymbol{\xi}_t\right)_\times \\
&= \left(-\left(\tilde{\boldsymbol{\omega}}_t\right)_\times \boldsymbol{\xi}_t\right)_\times \\
\implies \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi}_t &= \left(-\tilde{\boldsymbol{\omega}}_t\right)_\times \boldsymbol{\xi}_t
\end{aligned} \tag{4.17}$$

After making a first-order approximation of the exponential map, the tangent space error dynamics become linear. In addition, this linear system only depends on the error, not the state estimate directly. In other words, wrong state estimates will not affect the accuracy of the linearization, which leads to better accuracy and consistency of the filter. For SO(3), this effect is well studied and has been leveraged to design the commonly used QEKFs[3], [222, 207]. However, the extension to general matrix Lie groups, called the InEKF, was only recently developed by Barrau and Bonnabel [24].

In the above example, we utilized the first-order approximation for the exponential map of SO(3); $\mathrm{Exp}(\boldsymbol{\xi}_t) \approx \mathbf{I} + \left(\boldsymbol{\xi}_t\right)_\times$. In general, one may ask how much accuracy is lost when

[3]The set of quaternions, along with quaternion multiplication, actually forms a Lie group.

making this approximation. The surprising result by Barrau and Bonnabel [24] is that this linearization is, in fact, exact. This is the basis of Theorem 2. If the initial error is known, the nonlinear error dynamics can be exactly recovered from this linear system. In this work, we leverage these ideas to develop a contact-aided inertial observer for legged robots.

4.2 World-centric Right-Invariant EKF

In this section, we derive a "world-centric", right invariant extended Kalman filter (RIEKF) using inertial measurement unit (IMU) and contact motion models with corrections made through forward kinematic measurements. This RIEKF can be used to estimate the state of a robot that has an arbitrary (finite) number of points in contact with the static environment. The state is measured in the world frame (hence "world-centric"), and the number of contact points can vary over time if appropriate steps are taken when initializing or removing the contact states.

While the filter is particularly useful for legged robots, the same theory can be applied for manipulators as long as the contact assumptions (presented in Section 4.2.2) are verified. In order to be consistent with the standard InEKF theory, IMU biases are neglected for now. Section 4.4 provides a method for reintroducing the bias terms.

4.2.1 State Representation

As with typical aided inertial navigation, we wish to estimate the orientation, velocity, and position of the IMU (body) in the world frame [160, 83, 240]. These states are represented by $\mathbf{R}_{\mathrm{WB}}(t), {}_{\mathrm{W}}\mathbf{v}_{\mathrm{B}}(t)$, and ${}_{\mathrm{W}}\mathbf{p}_{\mathrm{WB}}(t)$ respectively. In addition, we append the position of all contact points (in the world frame), ${}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}_i}(t)$, to the list of state variables. This is similar to the approach taken in [35, 33].

The above collection of state variables forms a matrix Lie group, $\mathcal{G}$. Specifically, for N contact points, $\mathbf{X}_t \in \mathrm{SE}_{N+2}(3)$ can be represented by the following matrix:

$$\mathbf{X}_t \triangleq \begin{bmatrix} \mathbf{R}_{\mathrm{WB}}(t) & {}_{\mathrm{W}}\mathbf{v}_{\mathrm{B}}(t) & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WB}}(t) & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}_1}(t) & \cdots & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}_N}(t) \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Because the process and measurements models for each contact point, ${}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}_i}(t)$, are iden-

tical, without loss of generality, we will derive all further equations assuming only a single contact point. Furthermore, for the sake of readability, we introduce the following shorthand notation:

$$\mathbf{X}_t \triangleq \begin{bmatrix} \mathbf{R}_t & \mathbf{v}_t & \mathbf{p}_t & \mathbf{d}_t \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{u}_t = \begin{bmatrix} {}_{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{WB}}(t) \\ {}_{\mathrm{B}}\tilde{\mathbf{a}}_{\mathrm{WB}}(t) \end{bmatrix} \triangleq \begin{bmatrix} \tilde{\boldsymbol{\omega}}_t \\ \tilde{\mathbf{a}}_t \end{bmatrix}, \tag{4.18}$$

where the input $\mathbf{u}_t$ is formed from the angular velocity and linear acceleration measurements coming from the IMU. It is important to note that these measurements are taken in the body (or IMU) frame. The Lie algebra of $\mathcal{G}$, denoted by $\mathfrak{g}$, is an $N+5$ dimensional square matrix. We use the *hat* operator, $(\cdot)^\wedge : \mathbb{R}^{3N+9} \to \mathfrak{g}$, to map a vector to the corresponding element of the Lie algebra. In the case of a single contact, for example, this function is defined by:

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \left(\boldsymbol{\xi}^R\right)_\times & \boldsymbol{\xi}^v & \boldsymbol{\xi}^p & \boldsymbol{\xi}^d \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 \end{bmatrix},$$

where $(\cdot)_\times$ denotes a 3×3 skew-symmetric matrix. The inverse operation is defined using the *vee* operator, $(\cdot)^\vee : \mathfrak{g} \to \mathbb{R}^{3N+9}$. The matrix representation of the adjoint is given by:

$$\mathrm{Ad}_{\mathbf{X}_t} = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\mathbf{v}_t)_\times \mathbf{R}_t & \mathbf{R}_t & \mathbf{0} & \mathbf{0} \\ (\mathbf{p}_t)_\times \mathbf{R}_t & \mathbf{0} & \mathbf{R}_t & \mathbf{0} \\ (\mathbf{d}_t)_\times \mathbf{R}_t & \mathbf{0} & \mathbf{0} & \mathbf{R}_t \end{bmatrix}. \tag{4.19}$$

More details about the $\mathrm{SE}_K(3)$ Lie group are given in Section 3.4.5.3

Remark 5. The set of 3×3 skew-symmetric matrices actually forms $\mathfrak{so}(3)$, the Lie algebra of SO(3). Therefore, the $(\cdot)^\wedge$ notation could also be used to map from $\mathbb{R}^3$ to a skew-symmetric matrix. However, to avoid confusion from overloaded notation, we use $\left(\boldsymbol{\xi}^R\right)_\times$ instead of $(\boldsymbol{\xi}^R)^\wedge$ when when dealing with the rotation term.

4.2.2 Continuous System Dynamics

The IMU measurements are modeled as being corrupted by additive white Gaussian noise, per

$$\begin{aligned}\tilde{\boldsymbol{\omega}}_t &= \boldsymbol{\omega}_t + \mathbf{w}_t^g, \qquad \mathbf{w}_t^g \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}^g)\\ \tilde{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{w}_t^a, \qquad \mathbf{w}_t^a \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}^a);\end{aligned}$$

these are explicit measurements coming directly from a physical sensor. In contrast, the velocity of the contact point is implicitly *inferred* through a contact sensor; specifically, when a binary sensor indicates contact, the position of the contact point is *assumed to remain fixed in the world frame*, i.e. the measured velocity is zero. In order to accommodate potential slippage, the measured velocity is assumed to be the actual velocity plus white Gaussian noise, namely

$${}_{\mathrm{W}}\tilde{\mathbf{v}}_{\mathrm{C}}(t) = \mathbf{0}_{3\times1} = {}_{\mathrm{C}}\mathbf{v}_{\mathrm{C}}(t) + \mathbf{w}^v(t), \quad \mathbf{w}^v(t) \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}^v). \tag{4.20}$$

Using the IMU and contact measurements, the individual terms of the system dynamics can be written as:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_t &= \mathbf{R}_t\,(\tilde{\boldsymbol{\omega}}_t - \mathbf{w}_t^g)_\times\\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}_t &= \mathbf{R}_t(\tilde{\mathbf{a}}_t - \mathbf{w}_t^a) + \mathbf{g}\\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}_t &= \mathbf{v}_t\\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{d}_t &= \mathbf{R}_t\,\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t)(-\mathbf{w}_t^v),\end{aligned} \tag{4.21}$$

where $\mathbf{g}$ is the gravity vector and $\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t)$, arising from forward kinematics, is the measured orientation of the contact frame with respect to the IMU frame. Therefore, $\mathbf{R}_t\,\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t)$ is a rotation matrix that transforms a vector from the contact frame to the world frame.

In matrix form, the dynamics can be expressed as

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{X}_t &= \begin{bmatrix}\mathbf{R}_t\,(\tilde{\boldsymbol{\omega}}_t)_\times & \mathbf{R}_t\tilde{\mathbf{a}}_t + \mathbf{g} & \mathbf{v}_t & \mathbf{0}_{3\times1}\\ \mathbf{0}_{1\times3} & 0 & 0 & 0\\ \mathbf{0}_{1\times3} & 0 & 0 & 0\\ \mathbf{0}_{1\times3} & 0 & 0 & 0\end{bmatrix} - \begin{bmatrix}\mathbf{R}_t & \mathbf{v}_t & \mathbf{p}_t & \mathbf{d}_t\\ \mathbf{0}_{1\times3} & 1 & 0 & 0\\ \mathbf{0}_{1\times3} & 0 & 1 & 0\\ \mathbf{0}_{1\times3} & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}(\mathbf{w}_t^g)_\times & \mathbf{w}_t^a & \mathbf{0}_{3\times1} & \mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v\\ \mathbf{0}_{1\times3} & 0 & 0 & 0\\ \mathbf{0}_{1\times3} & 0 & 0 & 0\\ \mathbf{0}_{1\times3} & 0 & 0 & 0\end{bmatrix}\\ &\triangleq f_{u_t}(\mathbf{X}_t) - \mathbf{X}_t{\mathbf{w}_t}^\wedge,\end{aligned} \tag{4.22}$$

with $\mathbf{w}_t \triangleq \text{vec}(\mathbf{w}_t^g,\ \mathbf{w}_t^a,\ \mathbf{0}_{3\times1}, \mathbf{R}_{\text{BC}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v)$. The deterministic system dynamics, $f_{u_t}(\cdot)$, can be shown to satisfy the group affine property, (4.3). Therefore, following Theorem 1, the left- and right-invariant error dynamics will evolve independently of the system's state.

Using Theorem 1, the right-invariant error dynamics is

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\eta}_t^r &= f_{u_t}(\boldsymbol{\eta}_t^r) - \boldsymbol{\eta}_t^r f_{u_t}(\mathbf{I}_d) + (\bar{\mathbf{X}}_t \mathbf{w}_t{}^{\wedge} \bar{\mathbf{X}}_t^{-1})\boldsymbol{\eta}_t^r \\ &\triangleq g_{u_t}(\boldsymbol{\eta}_t^r) + \bar{\mathbf{w}}_t^{\wedge}\boldsymbol{\eta}_t^r\end{aligned} \tag{4.23}$$

where the second term arises from the additive noise. The derivation follows the results in Barrau and Bonnabel [24] and is not repeated here.

Furthermore, Theorem 2 specifies that the invariant error satisfies a log-linear property. Namely, if $\mathbf{A}_t$ is defined by $g_{u_t}(\text{Exp}(\boldsymbol{\xi})) \triangleq (\mathbf{A}_t\boldsymbol{\xi})^{\wedge} + \mathcal{O}(||\boldsymbol{\xi}||^2)$, then the log of the invariant error, $\boldsymbol{\xi} \in \mathbb{R}^{\dim\mathfrak{g}}$, approximately satisfies[4] the linear system

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi}_t &= \mathbf{A}_t^r\boldsymbol{\xi}_t + \bar{\mathbf{w}}_t = \mathbf{A}_t\boldsymbol{\xi}_t + \text{Ad}_{\bar{\mathbf{X}}_t}\mathbf{w}_t \\ \boldsymbol{\eta}_t^r &= \text{Exp}(\boldsymbol{\xi}_t).\end{aligned} \tag{4.24}$$

To compute the matrix $\mathbf{A}_t$, we linearize the invariant error dynamics, $g_{u_t}(\cdot)$, using the first order approximation $\boldsymbol{\eta}_t^r = \text{Exp}(\boldsymbol{\xi}_t) \approx \mathbf{I}_d + \boldsymbol{\xi}_t{}^{\wedge}$ to yield

$$\begin{aligned}&g_{u_t}(\text{Exp}(\boldsymbol{\xi}_t)) \approx \\ &\begin{bmatrix} \left(\mathbf{I} + (\boldsymbol{\xi}_t^R)_\times\right)(\tilde{\boldsymbol{\omega}}_t)_\times & \left(\mathbf{I} + (\boldsymbol{\xi}_t^R)_\times\right)\tilde{\mathbf{a}}_t + \mathbf{g} & \boldsymbol{\xi}_t^v & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} \mathbf{I} + (\boldsymbol{\xi}_t^R)_\times & \boldsymbol{\xi}_t^v & \boldsymbol{\xi}_t^p & \boldsymbol{\xi}_t^d \\ \mathbf{0}_{3\times1} & 1 & 0 & 0 \\ \mathbf{0}_{3\times1} & 0 & 1 & 0 \\ \mathbf{0}_{3\times1} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} (\tilde{\boldsymbol{\omega}}_t)_\times & \tilde{\mathbf{a}} + \mathbf{g} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}_{3\times3} & (\mathbf{g})_\times\boldsymbol{\xi}_t^R & \boldsymbol{\xi}_t^v & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3\times1} \\ (\mathbf{g})_\times\boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^v \\ \mathbf{0}_{3\times1} \end{bmatrix}^{\wedge}.\end{aligned} \tag{4.25}$$

With the above, we can express the prediction step of the RIEKF. The state estimate, $\bar{\mathbf{X}}_t$, is propagated though the deterministic system dynamics, while the covariance matrix, $\mathbf{P}_t$, is computed using the Riccati equation [165], namely,

$$\frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{X}}_t = f_{u_t}(\bar{\mathbf{X}}_t) \text{ and } \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{P}_t = \mathbf{A}_t\mathbf{P}_t + \mathbf{P}_t\mathbf{A}_t^{\mathsf{T}} + \bar{\mathbf{Q}}_t, \tag{4.26}$$

[4]With input noise, Theorem 2 no longer holds, and the linearization is only approximate.

where the matrices $\mathbf{A}_t$ and $\bar{\mathbf{Q}}_t$ are obtained from (4.25) and (4.24),

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\mathbf{g})_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \text{ and } \bar{\mathbf{Q}}_t = \mathrm{Ad}_{\bar{\mathbf{X}}_t} \mathrm{Cov}\left(\mathbf{w}_t\right) \mathrm{Ad}_{\bar{\mathbf{X}}_t}^\mathsf{T}. \tag{4.27}$$

Remark 6. For the right-invariant case, expression (4.27), $\mathbf{A}_t$ is time-invariant and the time subscript could be dropped. However, in general it can be time-varying, therefore, we use $\mathbf{A}_t$ throughout the paper.

4.2.3 Right-invariant Forward Kinematic Measurement Model

Let $\boldsymbol{\alpha}_t \in \mathbb{R}^M$ denote the vector of joint displacements (prismatic or revolute) between the body and the contact point. We assume that the encoder measurements are corrupted by additive white Gaussian noise.

$$\tilde{\boldsymbol{\alpha}}_t = \boldsymbol{\alpha}_t + \mathbf{w}_t^\alpha, \quad \mathbf{w}_t^\alpha \sim \mathcal{N}(\mathbf{0}_{M,1}, \boldsymbol{\Sigma}^\alpha) \tag{4.28}$$

Using forward kinematics, we measure the relative position of the contact point with respect to the body,

$${}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\boldsymbol{\alpha}_t) \triangleq {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t - \mathbf{w}_t^\alpha) \approx {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t) - {}_{\mathrm{B}}\mathbf{J}_{\mathrm{BC}}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\, \mathbf{w}_t^\alpha, \tag{4.29}$$

where ${}_{\mathrm{B}}\mathbf{J}_{\mathrm{BC}}^{\dot{p}}$ denotes the components of the analytical Jacobian mapping encoder rates to the time derivative of position. See Section 3.5 for more details. Using the state variables, the forward-kinematics position measurement becomes

$${}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t) = \mathbf{R}_t^\mathsf{T}(\mathbf{d}_t - \mathbf{p}_t) + {}_{\mathrm{B}}\mathbf{J}_{\mathrm{BC}}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha. \tag{4.30}$$

Re-written in matrix form, this measurement has the right-invariant observation form (4.7),

$$\underbrace{\begin{bmatrix} {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t) \\ 0 \\ 1 \\ -1 \end{bmatrix}}_{\mathbf{Y}_t} = \underbrace{\begin{bmatrix} \mathbf{R}_t^\mathsf{T} & -\mathbf{R}_t^\mathsf{T}\mathbf{v}_t & -\mathbf{R}_t^\mathsf{T}\mathbf{p}_t & -\mathbf{R}_t^\mathsf{T}\mathbf{d}_t \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{X}_t^{-1}} \underbrace{\begin{bmatrix} \mathbf{0}_{3\times 1} \\ 0 \\ 1 \\ -1 \end{bmatrix}}_{\mathbf{b}} + \underbrace{\begin{bmatrix} {}_{\mathrm{B}}\mathbf{J}_{\mathrm{BC}}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{V}_t}. \tag{4.31}$$

Therefore, the innovation depends solely on the invariant error and the update equations take the form [24, Section 3.1.2]

$$\begin{aligned}\bar{\mathbf{X}}_t^+ &= \operatorname{Exp}\left(\mathbf{L}_t\left(\bar{\mathbf{X}}_t\mathbf{Y}_t - \mathbf{b}\right)\right)\bar{\mathbf{X}}_t \\ \boldsymbol{\eta}_t^{r+} &= \operatorname{Exp}\left(\mathbf{L}_t\left(\boldsymbol{\eta}_t^r\mathbf{b} - \mathbf{b} + \bar{\mathbf{X}}_t\mathbf{V}_t\right)\right)\boldsymbol{\eta}_t^r,\end{aligned} \tag{4.32}$$

where $\operatorname{Exp}(\cdot)$ is the exponential map corresponding to the state matrix Lie group, $\mathcal{G}$, $\mathbf{L}_t$ is a gain matrix to be defined later, $\mathbf{b}^\mathsf{T} = \begin{bmatrix}\mathbf{0}_{1\times 3} & 0 & 1 & -1\end{bmatrix}$, and $\mathbf{Y}_t^\mathsf{T} = \begin{bmatrix}{}_\mathrm{B}\mathbf{p}_\mathrm{BC}^\mathsf{T}(\tilde{\boldsymbol{\alpha}}_t) & 0 & 1 & -1\end{bmatrix}$. Because the last three rows of $\bar{\mathbf{X}}_t\mathbf{Y}_t - \mathbf{b}$ are identically zero, we can express the update equations using a reduced dimensional gain, $\mathbf{K}_t$, and an auxiliary selection matrix $\boldsymbol{\Pi} \triangleq \begin{bmatrix}\mathbf{I} & \mathbf{0}_{3\times 3}\end{bmatrix}$, so that $\mathbf{L}_t\left(\bar{\mathbf{X}}_t\mathbf{Y}_t - \mathbf{b}\right) = \mathbf{K}_t\boldsymbol{\Pi}\bar{\mathbf{X}}_t\mathbf{Y}_t$ as detailed in Barrau [23].

Using the first order approximation of the exponential map, $\boldsymbol{\eta}_t^r = \operatorname{Exp}(\boldsymbol{\xi}_t) \approx \mathbf{I}_d + \boldsymbol{\xi}_t{}^\wedge$, and dropping higher-order terms, we can linearize the update equation (4.32),

$$\boldsymbol{\eta}_t^{r+} \approx \mathbf{I}_d + \boldsymbol{\xi}_t^{+\wedge} \approx \mathbf{I}_d + \boldsymbol{\xi}_t{}^\wedge + \left(\mathbf{K}_t\boldsymbol{\Pi}\left((\mathbf{I}_d + \boldsymbol{\xi}_t{}^\wedge)\begin{bmatrix}\mathbf{0}_{3\times 1}\\ 0\\ 1\\ -1\end{bmatrix} + \bar{\mathbf{X}}_t\begin{bmatrix}{}_\mathrm{B}\mathbf{J}_\mathrm{BC}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha\\ 0\\ 0\\ 0\end{bmatrix}\right)\right)^\wedge. \tag{4.33}$$

Therefore,

$$\begin{aligned}\boldsymbol{\xi}_t^{+\wedge} &= \boldsymbol{\xi}_t{}^\wedge + \left(\mathbf{K}_t\boldsymbol{\Pi}\left(\begin{bmatrix}\mathbf{I} + (\boldsymbol{\xi}_t^R)_\times & \boldsymbol{\xi}_t^v & \boldsymbol{\xi}_t^p & \boldsymbol{\xi}_t^d\\ \mathbf{0}_{3\times 1} & 1 & 0 & 0\\ \mathbf{0}_{3\times 1} & 0 & 1 & 0\\ \mathbf{0}_{3\times 1} & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{0}_{3\times 1}\\ 0\\ 1\\ -1\end{bmatrix} + \bar{\mathbf{X}}_t\begin{bmatrix}{}_\mathrm{B}\mathbf{J}_\mathrm{BC}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha\\ 0\\ 0\\ 0\end{bmatrix}\right)\right)^\wedge \\ &= \boldsymbol{\xi}_t{}^\wedge + \left(\mathbf{K}_t\boldsymbol{\Pi}\left(\begin{bmatrix}\boldsymbol{\xi}_t^p - \boldsymbol{\xi}_t^d\\ 0\\ 1\\ -1\end{bmatrix} + \bar{\mathbf{X}}_t\begin{bmatrix}{}_\mathrm{B}\mathbf{J}_\mathrm{BC}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha\\ 0\\ 0\\ 0\end{bmatrix}\right)\right)^\wedge.\end{aligned}$$

Taking $(\cdot)^\vee$ of both sides yields the linear update equation,

$$\begin{aligned}\boldsymbol{\xi}_t^+ &= \boldsymbol{\xi}_t - \mathbf{K}_t\left(\begin{bmatrix}\mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & -\mathbf{I} & \mathbf{I}\end{bmatrix}\boldsymbol{\xi}_t - \bar{\mathbf{R}}_t\,{}_\mathrm{B}\mathbf{J}_\mathrm{BC}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha\right) \\ &\triangleq \boldsymbol{\xi}_t - \mathbf{K}_t\left(\mathbf{H}_t\boldsymbol{\xi}_t - \bar{\mathbf{R}}_t\,{}_\mathrm{B}\mathbf{J}_\mathrm{BC}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha\right).\end{aligned}$$

Finally, we can write down the full state and covariance update equations of the RIEKF

using the derived linear update equation and the theory of Kalman filtering [165, 14, 17] as

$$\begin{aligned}\bar{\mathbf{X}}_t^+ &= \operatorname{Exp}\left(\mathbf{K}_t \boldsymbol{\Pi} \bar{\mathbf{X}}_t \mathbf{Y}_t\right) \bar{\mathbf{X}}_t \\ \mathbf{P}_t^+ &= (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_t (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)^\mathsf{T} + \mathbf{K}_t \bar{\mathbf{N}}_t \mathbf{K}_t^\mathsf{T},\end{aligned} \tag{4.34}$$

where the gain $\mathbf{K}_t$ is computed using

$$\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_t \mathbf{H}_t^\mathsf{T} + \bar{\mathbf{N}}_t \qquad \mathbf{K}_t = \mathbf{P}_t \mathbf{H}_t^\mathsf{T} \mathbf{S}_t^{-1}$$

and from (4.34), the matrices $\mathbf{H}_t$ and $\bar{\mathbf{N}}_t$ are given by

$$\begin{aligned}\mathbf{H}_t &= \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{I} & \mathbf{I}\end{bmatrix}, \\ \bar{\mathbf{N}}_t &= \bar{\mathbf{R}}_t \, {}_{\mathrm{B}}\mathbf{J}_{\mathrm{BC}}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t) \, \operatorname{Cov}(\mathbf{w}_t^\alpha) \, ({}_{\mathrm{B}}\mathbf{J}_{\mathrm{BC}}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t))^\mathsf{T} \, \bar{\mathbf{R}}_t^\mathsf{T}.\end{aligned} \tag{4.35}$$

4.2.4 Observability Analysis

Because the error dynamics are log-linear (c.f., Theorem 2), we can determine the unobservable states of the filter without having to perform a nonlinear observability analysis [23]. Noting that the linear error dynamics matrix in our case is time-invariant and nilpotent (with a degree of 3), the discrete-time state transition matrix is a polynomial in $\mathbf{A}_t$,

$$\boldsymbol{\Phi} = \operatorname{Exp}_m(\mathbf{A}_t \Delta t) = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\mathbf{g})_\times \Delta t & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \frac{1}{2}(\mathbf{g})_\times \Delta t^2 & \mathbf{I}\Delta t & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{4.36}$$

It follows that the discrete-time observability matrix is

$$\mathcal{O} = \begin{bmatrix} \mathbf{H} \\ \mathbf{H}\boldsymbol{\Phi} \\ \mathbf{H}\boldsymbol{\Phi}^2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{I} \\ -\frac{1}{2}(\mathbf{g})_\times \Delta t^2 & -\mathbf{I}\Delta t & -\mathbf{I} & \mathbf{I} \\ -2(\mathbf{g})_\times \Delta t^2 & -2\mathbf{I}\Delta t^2 & -\mathbf{I} & \mathbf{I} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \tag{4.37}$$

The last six columns (i.e., two matrix columns) of the observability matrix are clearly linearly dependent, which indicates the absolute position of the robot is unobservable. In addition, since the gravity vector only has a z component, the third column of $\mathcal{O}$ is all zeros. Therefore, a rotation about the gravity vector (yaw) is also unobservable. This linear observability analysis agrees with the nonlinear observability results of Bloesch et al. [35], albeit with

much less computation. Furthermore, as the error dynamics do not depend on the estimated state, there is no chance of the linearization spuriously increasing the numerical rank of the observability matrix [23]. This latter effect was previously known and studied by Bloesch et al. [35], and in order to resolve this problem, an observability-constrained EKF [130] was developed. In our proposed framework, by default, the discrete RIEKF has the same unobservable states as the underlying nonlinear system; hence, the developed discrete RIEKF intrinsically solves this problem.

4.3 Simulation Results

To investigate potential benefits or drawbacks of the proposed filter, we compare it against a state-of-the-art quaternion-based extended Kalman filter (QEKF), similar to those described by Bloesch et al. [35], Rotella et al. [196]. For implementation, the filter equations were discretized; see Section 4.10 for more details.

4.3.1 Quaternion-Based Filter Equations

The choice of error variables is the main difference between the invariant extended Kalman filter (InEKF) and the QEKF. Instead of the right-invariant error (4.2), a QEKF typically uses decoupled error states

$$\begin{aligned} \operatorname{Exp}(\delta\boldsymbol{\theta}_t) &\triangleq \mathbf{R}_t^\mathsf{T}\bar{\mathbf{R}}_t \\ \delta\mathbf{v}_t &\triangleq \mathbf{v}_t - \bar{\mathbf{v}}_t \\ \delta\mathbf{p}_t &\triangleq \mathbf{p}_t - \bar{\mathbf{p}}_t. \\ \delta\mathbf{d}_t &\triangleq \mathbf{d}_t - \bar{\mathbf{d}}_t. \end{aligned} \tag{4.38}$$

Using this definition of error, the QEKF deterministic error dynamics can be approximated as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\delta\boldsymbol{\theta}_t\\ \delta\mathbf{v}_t\\ \delta\mathbf{p}_t\\ \delta\mathbf{d}_t\end{bmatrix} = \begin{bmatrix} -(\tilde{\boldsymbol{\omega}}_t)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ -\bar{\mathbf{R}}_t(\tilde{\mathbf{a}}_t)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\theta}_t\\ \delta\mathbf{v}_t\\ \delta\mathbf{p}_t\\ \delta\mathbf{d}_t\end{bmatrix}, \tag{4.39}$$

while the linearized observation matrix becomes

$$\mathbf{H}_t = \begin{bmatrix}\left(\bar{\mathbf{R}}_t^\mathsf{T}(\bar{\mathbf{d}}_t - \bar{\mathbf{p}}_t)\right)_\times & \mathbf{0} & -\bar{\mathbf{R}}_t^\mathsf{T} & \bar{\mathbf{R}}_t^\mathsf{T}\end{bmatrix}. \tag{4.40}$$

The above linearizations are clearly dependent on the state estimate. Therefore, when

the estimated state deviates from the true state, the linearizations are potentially wrong, reducing accuracy and consistency in the QEKF. In contrast, the deterministic right-invariant error dynamics are exactly log-linear (4.27). In addition, the linearized observation matrix for our InEKF (4.35) is also independent of the state estimate.

4.3.2 Convergence Comparison

A dynamic *simulation* of a Cassie-series bipedal robot (described in Section 4.6) was performed in which the robot slowly walked forward after a small drop, accelerating from 0.0 to 0.3 m / sec. The discrete, simulated measurements were corrupted by additive white Gaussian noise, which are specified in Table 4.1 along with the initial state covariance values. The same values were used in both simulation and experimental convergence evaluations of the filters. The IMU bias estimation was turned off for these simulations. The simulation was performed with MATLAB and Simulink (Simscape Multibody™) where the simulation environment models ground contact forces with a linear force law (having a stiffness and damping term) and a Coulomb friction model. A typical walking gait is shown in Figure 4.2.

Table 4.1: Experiment Discrete Noise Statistics and Initial Covariance

Measurement Type	noise st. dev.
Linear Acceleration	0.04 m / sec^2
Angular Velocity	0.002 rad / sec
Accelerometer Bias	0.001 m / sec^3
Gyroscope Bias	0.001 rad / sec^2
Contact Linear Velocity	0.05 m / sec
Joint Encoders	1.0 deg

State Element	initial st. dev.
Orientation of IMU	30.0 deg
Velocity of IMU	1.0 m / sec
Position of IMU	0.1 m
Position of Right Foot	0.1 m
Position of Left Foot	0.1 m
Gyroscope Bias	0.005 rad / sec
Accelerometer Bias	0.05 m / sec^2

To compare the convergence properties of the two filters, 100 simulations of each filter were performed using identical measurements, noise statistics, initial covariance, and various random initial orientations and velocities. The initial Euler angle estimates were sampled uniformly from $-30\,\text{deg}$ to $30\,\text{deg}$. The initial velocity estimates were sampled uniformly from -1.0 m / sec to 1.0 m / sec. The pitch and roll estimates as well as the (body frame) velocity estimates for both filters are shown in Figure 4.3. Although both filters converge for this set of initial conditions, the proposed right invariant extended Kalman filter (RIEKF) converges considerably faster than the standard quaternion-based EKF.

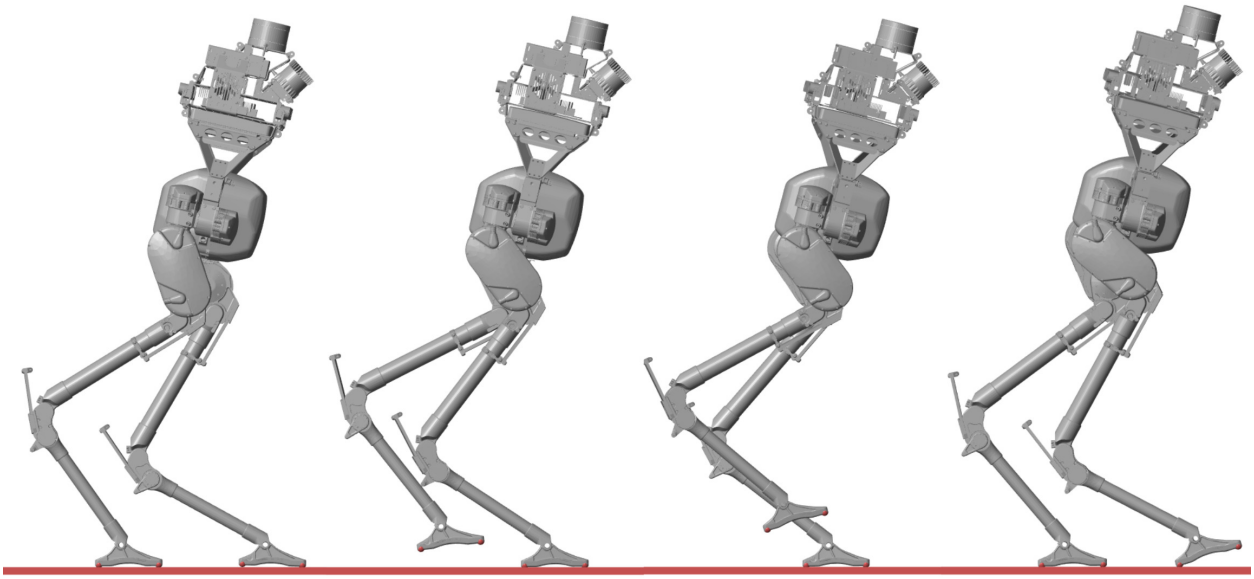


Figure 4.2: A typical walking gait that is used for filter comparisons. The Cassie bipedal robot is simulated using Simscape Multibody™.

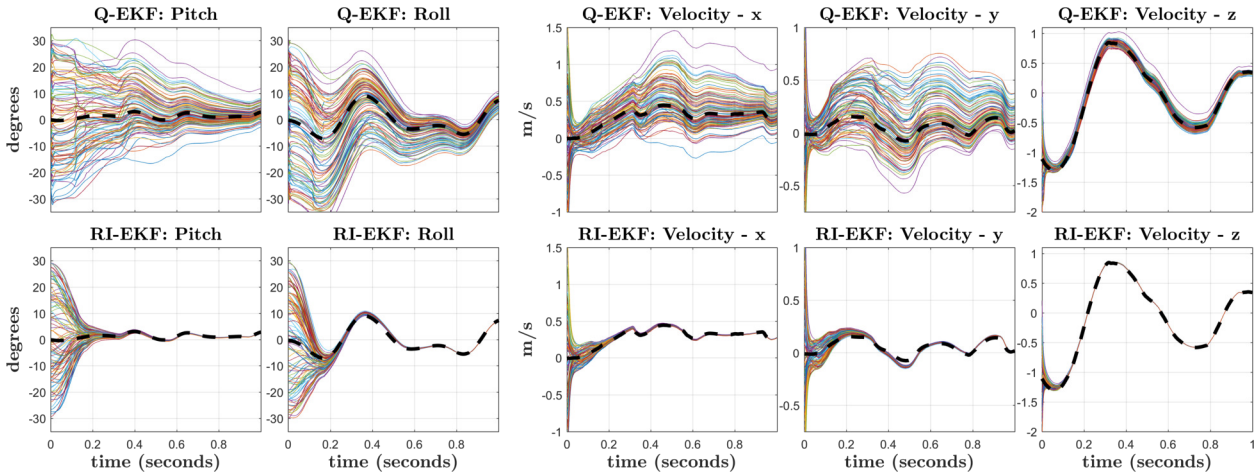


Figure 4.3: A quaternion-based extended Kalman filter (QEKF) and the proposed right invariant extended Kalman filter (RIEKF) were run 100 times using the same measurements, noise statistics, and initial covariance, but with random initial orientations and velocities. The noisy measurements came from a dynamic simulation of a Cassie-series biped robot where the robot walks forwards after a small drop, accelerating from 0.0 to 0.3 m / sec. The above plots show the state estimate for the initial second of data, where the dashed black line represents the true state. The RIEKF (bottom row) converges considerably faster than the QEKF (top row) for all observable states. The estimated yaw angle (not shown) does not converge for either filter because it is unobservable. Therefore, to compare convergence, the velocities shown are represented in the estimated IMU (body) frame.

4.3.3 Accuracy of Linearized Dynamics

The superior performance of the InEKF over the QEKF comes from the improved accuracy of the linearized error dynamics. As indicated by Theorem 2, the deterministic error dynamics of the InEKF are actually exact, while the QEKF version is only an approximation. To demonstrate this, a simulation was performed where propagation of the true error is compared to the propagation of the linearized error dynamics.

We first analyzed the deterministic dynamics. Given an initial error vector, $\boldsymbol{\xi}_0^{\text{true}}$, the initial state estimate for the InEKF was computed using the definition of right-invariant error (4.2), and the initial state estimate for the QEKF was computed using equation (4.38). The true state was initialized to the identity element. The true and estimated states for both filters were then propagated for 1 second (1000 time steps), using randomly sampled inertial measurement unit (IMU) measurements. The error states for both the InEKF and the QEKF were also propagated using their respective linearized error dynamics. The resulting error, $\boldsymbol{\xi}_1^{\text{true}}$, between the final estimated and true states were computed and compared to the propagated error states, $\boldsymbol{\xi}_1^{\text{prop}}$ to yield a measure of linearization accuracy, $\|\boldsymbol{\xi}_1^{\text{true}} - \boldsymbol{\xi}_1^{\text{prop}}\|$. This test was performed multiple times while linearly scaling the initial orientation error from $\text{vec}(0,0,0)$ to $\text{vec}(\pi/2, \pi/2, \pi/2)$. The results are shown in Figure 4.4. As expected,

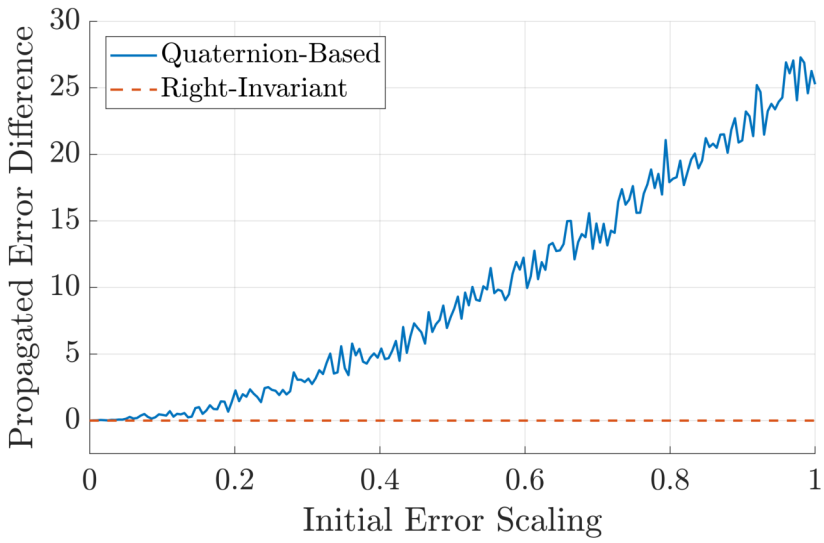


Figure 4.4: Analyzing accuracy of the deterministic error dynamics. This Figure shows the difference between the true error and the propagated error as the initial true error increases. The state and errors were propagated for 1 second using randomly sampled IMU measurements.

when the initial error is zero, the difference between the true and propagated error is zero. This indicates that the linearized error dynamics for both the InEKF and the QEKF are correct. As the initial error increases, the difference between the true and propagated error

states for the QEKF grows due to the decreased accuracy of the linearization. In contrast, the difference between the true and propagated error states for the InEKF are always exactly zero regardless of the initial error. In other words, assuming the initial error is known, the true propagated state can be exactly recovered from solving the linearized error dynamics system; see Theorem 2.

In the non-deterministic case (with sensor noise), Theorem 2 no longer holds. This can be seen in Figure 4.5, where the same test was performed, but with sensor noise corrupting the propagated state estimate. The difference between the true and propagated error is no longer exactly zero for the InEKF. However, the InEKF linearization remains more accurate due

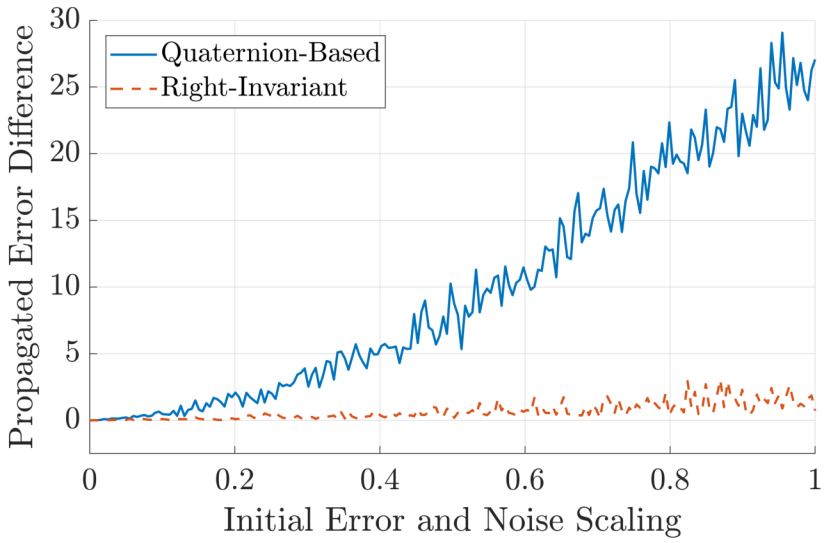


Figure 4.5: Difference between the true and propagated errors when measurements contain noise. The log-linear error dynamics of the InEKF are no longer exact.

to the reduced sensitivity to initial state errors. This helps to further explain the improved convergence properties shown in Figure 4.3.

4.3.4 Covariance Ellipse Comparison

The error states in both the QEKF and the InEKF are assumed to be zero-mean Gaussian random vectors. However, due to the differing choice of error variables, the state uncertainty will differ. In the QEKF, all states and errors are decoupled (4.38). For example, the true position only depends on the position estimate and the position error, $\mathbf{p} = \bar{\mathbf{p}} + \delta\mathbf{p}$. Therefore, the position estimate is a Gaussian centered at $\bar{\mathbf{p}}$. In contrast, when using the InEKF, the position and orientation are actually coupled together, $\mathbf{X} = \mathrm{Exp}(\boldsymbol{\xi})\bar{\mathbf{X}}$. Although $\boldsymbol{\xi}$ is a Gaussian random vector, after applying the group's exponential map and matrix

multiplication, the state estimate's uncertainty distribution is no longer Gaussian. This distribution is known as a *concentrated Gaussian on a Lie group* [229, 230]. This type of distribution can often capture the underlying system uncertainty better than a standard Gaussian defined in Euclidean space [159, 22].

To demonstrate the difference, a simple simulation was performed where the Cassie robot walked forward for 8 sec at an average speed of 1 m/s. The standard deviation for the initial position uncertainty was set to 0.1 m about each axis, while the standard deviation of the initial yaw uncertainty was set to 10 deg. A set of 10,000 particles sampled from this distribution were propagated forward to represent the robot's true uncertainty distribution. After running both the InEKF and the QEKF, particles were sampled from the resulting filter covariances to provide a picture of the estimated position uncertainties. This result is shown in Figure 4.6. The curved position distribution comes from the initial yaw uncertainty

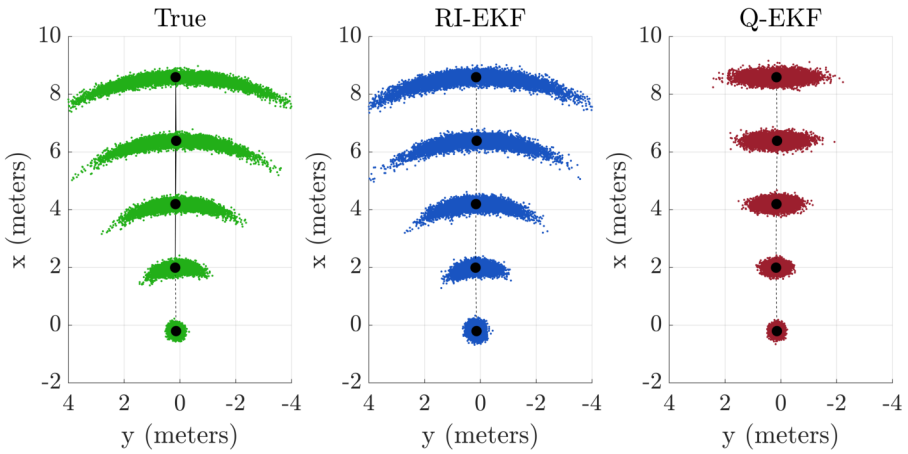


Figure 4.6: 10,000 samples taken from the estimated filter covariances for a simulation where Cassie walked forward with an average speed of 1 m/s. The position distributions at times 0, 2, 4, 6, and 8 sec are shown.

that continually grows due to its unobservability (Section 4.2.4). The InEKF is able to closely match this distribution since the samples are taken in the Lie algebra and are mapped to the group through the exponential map. This couples the orientation and position errors leading to a curved position distribution. In contrast, the QEKF position uncertainty can only have the shape of the standard Gaussian ellipse, which may not represent the true uncertainty well.

The InEKF can even accurately model the case of complete yaw uncertainty. To demonstrate this, the initial yaw standard deviation was set to 360 deg, and the same 8 sec simulation was performed. Each ring in Figure 4.7 shows the sample position distribution

spaced 2 sec apart. This type of uncertainty cannot be captured with a standard Gaussian covariance ellipse. These examples illustrate that even if the means are identical, the covariance estimate of the InEKF can provide a more accurate representation of the state's uncertainty than the standard QEKF.

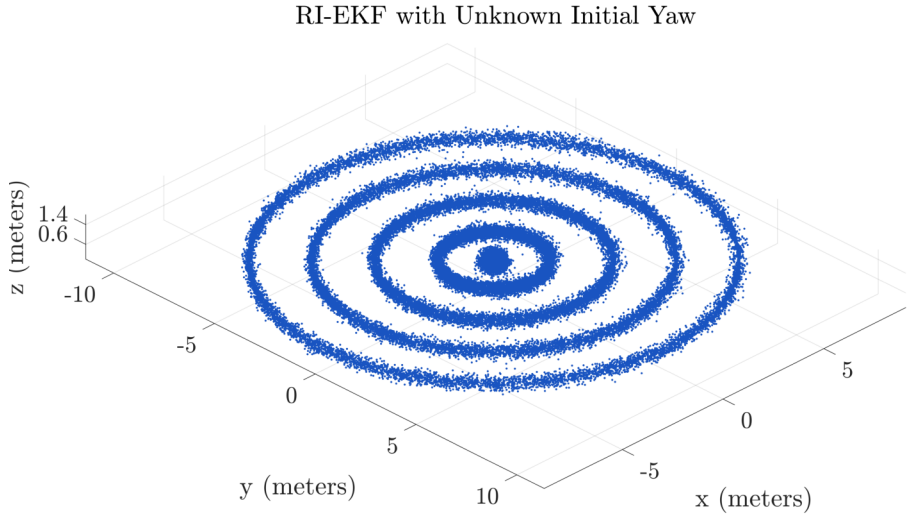


Figure 4.7: Samples taken from the InEKF's estimated position distribution for a walking simulation with a completely uncertain initial yaw angle. The robot moved forward at an average speed of 1 m/s. Each ring represents the position uncertainty at times 0, 2, 4, 6, and 8 sec.

4.4 IMU bias augmentation

Implementation of an IMU-based state estimator on hardware typically requires modeling additional states, such as gyroscope and accelerometer biases. Unfortunately, as noted in Barrau [23], there is no Lie group that includes the bias terms while also having the dynamics satisfy the group affine property (4.3). Even though many of the theoretical properties of the right invariant extended Kalman filter (RIEKF) will no longer hold, it is possible to design an "imperfect invariant extended Kalman filter (InEKF)" that still outperforms the standard EKF [23].

4.4.1 State Representation

The IMU biases are slowly varying signals that corrupt the measurements in an additive manner:

$$\begin{aligned}\tilde{\boldsymbol{\omega}}_t &= \boldsymbol{\omega}_t + \mathbf{b}_t^g + \mathbf{w}_t^g, \qquad & \mathbf{w}_t^g &\sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}^g)\\ \tilde{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{b}_t^a + \mathbf{w}_t^a, \qquad & \mathbf{w}_t^a &\sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}^a).\end{aligned}$$

These biases form a parameter vector that needs to be estimated as part of the RIEKF state,

$$\boldsymbol{\theta}_t \triangleq \begin{bmatrix}\mathbf{b}^g(t)\\ \mathbf{b}^a(t)\end{bmatrix} \triangleq \begin{bmatrix}\mathbf{b}_t^g\\ \mathbf{b}_t^a\end{bmatrix} \in \mathbb{R}^6. \tag{4.41}$$

The model's state now becomes a tuple of our original matrix Lie group and the parameter vector, $(\mathbf{X}_t, \boldsymbol{\theta}_t) \in \mathcal{G} \times \mathbb{R}^6$. The augmented right-invariant error is now defined as

$$\mathbf{e}_t^r \triangleq (\bar{\mathbf{X}}_t \mathbf{X}_t^{-1}, \bar{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t) \triangleq (\boldsymbol{\eta}_t^r, \boldsymbol{\zeta}_t). \tag{4.42}$$

Written explicitly, the right-invariant error is

$$\boldsymbol{\eta}_t^r = \begin{bmatrix}\bar{\mathbf{R}}_t\mathbf{R}_t^\mathsf{T} & \bar{\mathbf{v}}_t - \bar{\mathbf{R}}_t\mathbf{R}_t^\mathsf{T}\mathbf{v}_t & \bar{\mathbf{p}}_t - \bar{\mathbf{R}}_t\mathbf{R}_t^\mathsf{T}\mathbf{p}_t & \bar{\mathbf{d}}_t - \bar{\mathbf{R}}_t\mathbf{R}_t^\mathsf{T}\mathbf{d}_t\\ \mathbf{0}_{1\times3} & 1 & 0 & 0\\ \mathbf{0}_{1\times3} & 0 & 1 & 0\\ \mathbf{0}_{1\times3} & 0 & 0 & 1\end{bmatrix}, \tag{4.43}$$

while the parameter vector error is given by

$$\boldsymbol{\zeta}_t = \begin{bmatrix}\bar{\mathbf{b}}_t^g - \mathbf{b}_t^g\\ \bar{\mathbf{b}}_t^a - \mathbf{b}_t^a\end{bmatrix} \triangleq \begin{bmatrix}\boldsymbol{\zeta}_t^g\\ \boldsymbol{\zeta}_t^a\end{bmatrix}. \tag{4.44}$$

4.4.2 System Dynamics

With inertial measurement unit (IMU) biases included, the system dynamics are now expressed as

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_t &= \mathbf{R}_t \left(\tilde{\boldsymbol{\omega}}_t - \mathbf{b}_t^g - \mathbf{w}_t^g\right)_\times \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}_t &= \mathbf{R}_t(\tilde{\mathbf{a}}_t - \mathbf{b}_t^a - \mathbf{w}_t^a) + \mathbf{g} \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}_t &= \mathbf{v}_t \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{d}_t &= \mathbf{R}_t \; \boldsymbol{h}_R(\tilde{\boldsymbol{\alpha}}_t)(-\mathbf{w}_t^v).
\end{aligned} \tag{4.45}$$

The IMU bias dynamics are modeled using the typical "Brownian motion" model, i.e., the derivatives are white Gaussian noise, to capture the slowly time-varying nature of these parameters,

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{b}_t^g &= \mathbf{w}_t^{bg}, \qquad & \mathbf{w}_t^{bg} &\sim \mathcal{N}(\mathbf{0}_{3\times 1}, \boldsymbol{\Sigma}^{bg}) \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{b}_t^a &= \mathbf{w}_t^{ba}, \qquad & \mathbf{w}_t^{ba} &\sim \mathcal{N}(\mathbf{0}_{3\times 1}, \boldsymbol{\Sigma}^{ba}).
\end{aligned} \tag{4.46}$$

The deterministic system dynamics now depend on both the inputs, $\mathbf{u}_t$, and the parameters, $\boldsymbol{\theta}_t$

$$f_{\mathbf{u}_t}(\bar{\mathbf{X}}_t, \bar{\boldsymbol{\theta}}_t) = \begin{bmatrix} \bar{\mathbf{R}}_t \left(\bar{\boldsymbol{\omega}}_t\right)_\times & \bar{\mathbf{R}}_t \bar{\mathbf{a}}_t + \mathbf{g} & \bar{\mathbf{v}}_t & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 \end{bmatrix}, \tag{4.47}$$

where $\bar{\boldsymbol{\omega}}_t \triangleq \tilde{\boldsymbol{\omega}}_t - \bar{\mathbf{b}}_t^g$ and $\bar{\mathbf{a}}_t \triangleq \tilde{\mathbf{a}}_t - \bar{\mathbf{b}}_t^a$ are the "bias-corrected" inputs. To compute the linearized error dynamics, the augmented right-invariant error (4.42) is first differentiated with respect to time,

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{e}_t^r = \left(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\eta}_t^r, \begin{bmatrix} \mathbf{w}_t^{bg} \\ \mathbf{w}_t^{ba} \end{bmatrix}\right). \tag{4.48}$$

After carrying out the chain rule and making the first order approximation, $\boldsymbol{\eta}_t^r = \mathrm{Exp}(\boldsymbol{\xi}_t) \approx \mathbf{I}_d + \boldsymbol{\xi}_t{}^\wedge$,

the individual terms of the invariant error dynamics become

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(\bar{\mathbf{R}}_t\mathbf{R}_t^{\mathsf{T}}\right) &\approx \left(\bar{\mathbf{R}}_t\left(\mathbf{w}_t^g-\boldsymbol{\zeta}_t^g\right)\right)_\times \\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\bar{\mathbf{v}}_t-\bar{\mathbf{R}}_t\mathbf{R}_t^{\mathsf{T}}\mathbf{v}_t\right) &\approx (\mathbf{g})_\times\boldsymbol{\xi}_t^R+(\bar{\mathbf{v}}_t)_\times\bar{\mathbf{R}}_t(\mathbf{w}_t^g-\boldsymbol{\zeta}_t^g) \\
&\quad+\bar{\mathbf{R}}_t(\mathbf{w}_t^a-\boldsymbol{\zeta}_t^a) \\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\bar{\mathbf{p}}_t-\bar{\mathbf{R}}_t\mathbf{R}_t^{\mathsf{T}}\mathbf{p}_t\right) &\approx \boldsymbol{\xi}_t^v+(\bar{\mathbf{p}}_t)_\times\bar{\mathbf{R}}_t(\mathbf{w}_t^g-\boldsymbol{\zeta}_t^g) \\
\frac{\mathrm{d}}{\mathrm{d}t}\left(\bar{\mathbf{d}}_t-\bar{\mathbf{R}}_t\mathbf{R}_t^{\mathsf{T}}\mathbf{d}_t\right) &\approx \left(\bar{\mathbf{d}}_t\right)_\times\bar{\mathbf{R}}_t(\mathbf{w}_t^g-\boldsymbol{\zeta}_t^g) \\
&\quad+\bar{\mathbf{R}}_t\,\boldsymbol{h}_R(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v.
\end{aligned} \tag{4.49}$$

Importantly, the augmented invariant error dynamics only depends on the estimated trajectory though the noise and bias errors, $\boldsymbol{\zeta}_t$ (this is expected because when there are no bias errors, there is no dependence on the estimated trajectory). A linear system can now be constructed from (4.49) to yield,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{\xi}_t\\ \boldsymbol{\zeta}_t\end{bmatrix}=\mathbf{A}_t\begin{bmatrix}\boldsymbol{\xi}_t\\ \boldsymbol{\zeta}_t\end{bmatrix}+\begin{bmatrix}\mathrm{Ad}_{\bar{\mathbf{X}}_t} & \mathbf{0}_{12,6}\\ \mathbf{0}_{6,12} & \mathbf{I}_6\end{bmatrix}\mathbf{w}_t, \tag{4.50}$$

where the noise vector is augmented to include the bias terms,

$$\mathbf{w}_t\triangleq\mathrm{vec}(\mathbf{w}_t^g,\,\mathbf{w}_t^a,\,\mathbf{0}_{3\times1},\,\boldsymbol{h}_R(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v,\mathbf{w}_t^{bg},\mathbf{w}_t^{ba}).$$

4.4.3 Forward Kinematic Measurements

The forward kinematics position measurement (4.30) does not depend on the IMU biases. Therefore, the $\mathbf{H}_t$ matrix can simply be appended with zeros to account for the augmented variables. The linear update equation becomes

$$\begin{bmatrix}\boldsymbol{\xi}_t^+\\ \boldsymbol{\zeta}_t^+\end{bmatrix}=\begin{bmatrix}\boldsymbol{\xi}_t\\ \boldsymbol{\zeta}_t\end{bmatrix}-\begin{bmatrix}\mathbf{K}_t^{\boldsymbol{\xi}}\\ \mathbf{K}_t^{\boldsymbol{\zeta}}\end{bmatrix}\left(\mathbf{H}_t\begin{bmatrix}\boldsymbol{\xi}_t\\ \boldsymbol{\zeta}_t\end{bmatrix}-\bar{\mathbf{R}}_t\left(\mathbf{J}_v(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha\right)\right). \tag{4.51}$$

4.4.4 Final Continuous RIEKF Equations

The final "imperfect" RIEKF equations that include IMU biases can now be written down. The estimated state tuple is predicted using the following set of differential equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\bar{\mathbf{X}}_t\,,\bar{\boldsymbol{\theta}}_t\right)=\left(f_{\mathbf{u}_t}(\bar{\mathbf{X}}_t,\bar{\boldsymbol{\theta}}_t),\mathbf{0}_{6,1}\right). \tag{4.52}$$

The covariance of the augmented right invariant error dynamics is computed by solving the Riccati equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{P}_t = \mathbf{A}_t\mathbf{P}_t + \mathbf{P}_t\mathbf{A}_t^\mathsf{T} + \bar{\mathbf{Q}}_t, \tag{4.53}$$

where the matrices $\mathbf{A}_t$ and $\bar{\mathbf{Q}}_t$ are now defined using (4.49),

$$\begin{aligned}
\mathbf{A}_t &= \begin{bmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\bar{\mathbf{R}}_t & \mathbf{0} \\
(\mathbf{g})_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & -(\bar{\mathbf{v}}_t)_\times \bar{\mathbf{R}}_t & -\bar{\mathbf{R}}_t \\
\mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & -(\bar{\mathbf{p}}_t)_\times \bar{\mathbf{R}}_t & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -(\bar{\mathbf{d}}_t)_\times \bar{\mathbf{R}}_t & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{bmatrix} \\
\bar{\mathbf{Q}}_t &= \begin{bmatrix} \mathrm{Ad}_{\bar{\mathbf{X}}_t} & \mathbf{0}_{12,6} \\ \mathbf{0}_{6,12} & \mathbf{I}_6 \end{bmatrix} \mathrm{Cov}(\mathbf{w}_t) \begin{bmatrix} \mathrm{Ad}_{\bar{\mathbf{X}}_t} & \mathbf{0}_{12,6} \\ \mathbf{0}_{6,12} & \mathbf{I}_6 \end{bmatrix}^\mathsf{T}.
\end{aligned} \tag{4.54}$$

The estimated state tuple and its covariance are corrected though the update equations

$$\begin{aligned}
\left(\bar{\mathbf{X}}_t^+, \boldsymbol{\theta}_t^+\right) &= \left(\mathrm{Exp}\left(\mathbf{K}_t^{\boldsymbol{\xi}}\boldsymbol{\Pi}\bar{\mathbf{X}}_t\mathbf{Y}_t\right)\bar{\mathbf{X}}_t\,,\ \ \bar{\boldsymbol{\theta}}_t + \mathbf{K}_t^{\boldsymbol{\zeta}}\boldsymbol{\Pi}\bar{\mathbf{X}}_t\mathbf{Y}_t\right) \\
\mathbf{P}_t^+ &= (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_t(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)^\mathsf{T} + \mathbf{K}_t\bar{\mathbf{N}}_t\mathbf{K}_t^\mathsf{T},
\end{aligned} \tag{4.55}$$

where the gains $\mathbf{K}_t^{\boldsymbol{\xi}}$ and $\mathbf{K}_t^{\boldsymbol{\zeta}}$ are computed from

$$\mathbf{S}_t = \mathbf{H}_t\mathbf{P}_t\mathbf{H}_t^\mathsf{T} + \bar{\mathbf{N}}_t \qquad \mathbf{K}_t = \begin{bmatrix} \mathbf{K}_t^{\boldsymbol{\xi}} \\ \mathbf{K}_t^{\boldsymbol{\zeta}} \end{bmatrix} = \mathbf{P}_t\mathbf{H}_t^\mathsf{T}\mathbf{S}_t^{-1},$$

with the following measurement, output, and noise matrices,

$$\begin{aligned}
\mathbf{Y}_t^\mathsf{T} &= \begin{bmatrix} \boldsymbol{h}_p^\mathsf{T}(\tilde{\boldsymbol{\alpha}}_t) & 0 & 1 & -1 \end{bmatrix}, \\
\mathbf{H}_t &= \begin{bmatrix} \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \\
\bar{\mathbf{N}}_t &= \bar{\mathbf{R}}_t\, \mathbf{J}_v(\tilde{\boldsymbol{\alpha}}_t)\, \mathrm{Cov}(\mathbf{w}_t^\alpha)\, \mathbf{J}_v^\mathsf{T}(\tilde{\boldsymbol{\alpha}}_t)\, \bar{\mathbf{R}}_t^\mathsf{T}.
\end{aligned}$$

Remark 7. The upper-right block of the new linearized dynamics matrix (4.54) is related to the adjoint of the current state estimate (4.19). Intuitively, this maps the bias error (measured in the body frame) to the world frame.

4.5 Addition and Removal of Contact Points

Sections 4.2 and 4.4 derived the equations for the right invariant extended Kalman filter (RIEKF) under the assumption that the contact point is unchanging with time. However, for legged robots, contacts are discrete events that are created and broken as a robot navigates through the environment. Therefore, it is important to be able to conveniently add and remove contact points states to and from the observer's state.

4.5.1 Removing Contact Points

To remove a previous contact point from the state, we marginalize the corresponding state variable by simply removing the corresponding column and row from the matrix Lie group. The corresponding elements of the covariance matrix are also eliminated. This can be done through a simple linear transformation. For example, if the robot is going from one contact to zero contacts, then the newly reduced covariance would be computed by

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^v \\ \boldsymbol{\xi}_t^p \end{bmatrix} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^v \\ \boldsymbol{\xi}_t^p \\ \boldsymbol{\xi}_t^d \end{bmatrix} \\ \boldsymbol{\xi}_t^{\text{new}} &\triangleq \mathbf{M}\,\boldsymbol{\xi}_t \\ \implies \mathbf{P}_t^{\text{new}} &= \mathbf{M}\,\mathbf{P}_t\,\mathbf{M}^\mathsf{T}. \end{aligned} \tag{4.56}$$

Remark 8. This marginalization matrix, $\mathbf{M}$, does not depend on the choice of right or left invariant error.

4.5.2 Adding Contact Points

When the robot makes a new contact with the environment, the state and covariance matrices need to be augmented. Special attention needs to be given to initialize the mean and covariance for the new estimated contact point. For example, if the robot is going from zero contacts to one contact, the initial mean is obtained though the forward kinematics relation

$$\bar{\mathbf{d}}_t = \bar{\mathbf{p}}_t + \bar{\mathbf{R}}_t\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t). \tag{4.57}$$

In order to compute the new covariance, we need to look at the right-invariant error,

$$\begin{aligned}
\boldsymbol{\eta}_t^d &= \bar{\mathbf{d}}_t - \bar{\mathbf{R}}_t \mathbf{R}_t^\mathsf{T} \mathbf{d}_t \\
&= \bar{\mathbf{p}}_t + \bar{\mathbf{R}}_{t\,\mathrm{B}} \mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t) - \bar{\mathbf{R}}_t \mathbf{R}_t^\mathsf{T} \mathbf{d}_t \\
&= \bar{\mathbf{p}}_t + \bar{\mathbf{R}}_{t\,\mathrm{B}} \mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t) - \bar{\mathbf{R}}_t \mathbf{R}_t^\mathsf{T} \left(\mathbf{p}_t + \mathbf{R}_{t\,\mathrm{B}} \mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t - \mathbf{w}_t^\alpha)\right) \\
&\approx \boldsymbol{\eta}_t^p + \bar{\mathbf{R}}_{t\,\mathrm{B}} \mathbf{J}_{\mathrm{BC}}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\, \mathbf{w}_t^\alpha \\
\implies \boldsymbol{\xi}_t^d &\approx \boldsymbol{\xi}_t^p + \bar{\mathbf{R}}_{t\,\mathrm{B}} \mathbf{J}_{\mathrm{BC}}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t)\, \mathbf{w}_t^\alpha.
\end{aligned} \tag{4.58}$$

Therefore, covariance augmentation can be done using the following linear map,

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^v \\ \boldsymbol{\xi}_t^p \\ \boldsymbol{\xi}_t^d \end{bmatrix} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^v \\ \boldsymbol{\xi}_t^p \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \bar{\mathbf{R}}_{t\,\mathrm{B}} \mathbf{J}_{\mathrm{BC}}^{\dot{p}}(\tilde{\boldsymbol{\alpha}}_t) \end{bmatrix} \mathbf{w}_t^\alpha \\
\boldsymbol{\xi}_t^{\text{new}} &\triangleq \mathbf{F}_t\, \boldsymbol{\xi}_t + \mathbf{G}_t \mathbf{w}_t^\alpha \\
\implies \mathbf{P}_t^{\text{new}} &= \mathbf{F}_t\, \mathbf{P}_t\, \mathbf{F}_t^\mathsf{T} + \mathbf{G}_t\, \mathrm{Cov}(\mathbf{w}_t^\alpha)\, \mathbf{G}_t^\mathsf{T}.
\end{aligned} \tag{4.59}$$

Remark 9. The error augmentation matrix, $\mathbf{F}_t$, and the noise matrix, $\mathbf{G}_t$, will depend on the choice of error variable. Here they are derived for the right invariant error case. The matrices will differ in the left invariant error formulation, as detailed in Section 4.7.

4.6 Experimental Results on Cassie Robot

We now present an experimental evaluation of the proposed contact-aided right invariant extended Kalman filter (RIEKF) observer using a 3D bipedal robot. The Cassie-series robot, shown in Figure 4.1, developed by Agility Robotics, has 20 degrees of freedom coming from the body pose, 10 actuators, and 4 springs. The robot is equipped with an IMU along with 14 joint encoders that can measure all actuator and spring angles. The proposed and baseline algorithms (along with the robot's feedback controller) are implemented in MATLAB (Simulink Real-Time). The IMU (model VN-100) is located in the robot's torso and provides angular velocity and linear acceleration measurements at 800 Hz. The encoders provide joint angle measurements at 2000 Hz. The robot has two springs on each leg that are compressed when the robot is standing on the ground. The spring deflections are measured by encoders and serve as a binary contact sensor. The controller used for these experiments

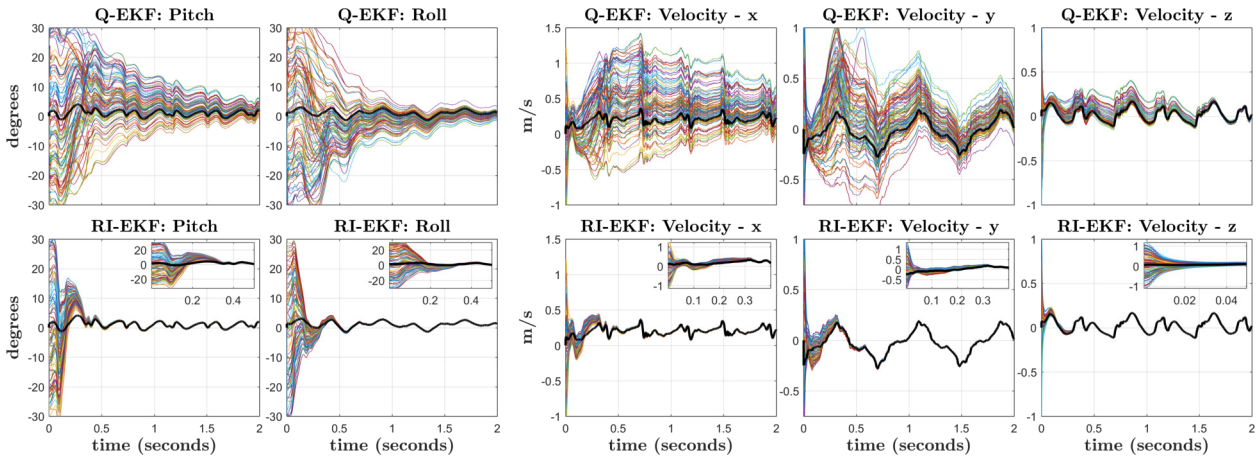


Figure 4.8: An experiment was performed where an actual Cassie-series robot slowly walked forward at approximately 0.3 m / sec. The noisy measurements came from the on-board inertial measurement unit (IMU) (VN-100) and the robot's joint encoders. The quaternion-based extended Kalman filter (QEKF) and the proposed right invariant extended Kalman filter (RIEKF) were run (off-line) 100 times using the same measurements, noise statistics, and initial covariance, but with random initial orientations and velocities. The black line represents the filter state estimates when initialized with a good estimate. The RIEKF (bottom row) converges considerably faster than the QEKF (top row) for all observable states. Zoomed-in plots of the RIEKF performance is provided in the top-right corner.

was developed by Gong et al. [91].

4.6.1 Convergence Comparison

An experiment was performed where the robot walked forwards at approximately 0.3 m / sec. The quaternion-based extended Kalman filter (QEKF) and the proposed RIEKF were run (off-line) 100 times using the same logged measurements, noise statistics, and initial covariance with random initial orientations and velocities. The noise statistics and initial covariance estimates are provided in Table 4.1. As with the simulation comparison presented in Section 4.3, the initial mean estimate for the Euler angles were uniformly sampled from -30 deg to 30 deg and the initial mean estimate for velocities were sampled uniformly from -1.0 m / sec to 1.0 m / sec. Bias estimation was turned on and the initial bias estimate was obtained from processing the IMU data when the robot was static. The pitch and roll estimates as well as the (body frame) velocity estimates for both filters are shown in Figure 4.8. The experimental results for comparing filter convergence matches those of the simulation. The proposed RIEKF converges faster and more reliably in all 100 runs than the QEKF; therefore, due to the convenience of initialization and reliability for tracking the developed RIEKF is the preferred observer.

When the state estimate is initialized close to the true value, the RIEKF and QEKF have similar performance, because the linearization of the error dynamics accurately reflects the

underlying nonlinear dynamics. However, when the state estimate is far from the true value, the simulation and experimental results show that RIEKF consistently converges faster than the QEKF. The relatively poor performance of the QEKF is due to the error dynamics being linearized around the wrong operating point, in which case the linear system does not accurately reflect the nonlinear dynamics. In addition, when bias estimation is turned off, the invariant error dynamics of the RIEKF do not depend on the current state estimate. As a result, the linear error dynamics can be accurately used even when the current state estimate is far from its true value, leading to better performance over the QEKF. Although this theoretical advantage is lost when bias estimation is turned on, the experimental results (shown in Figure 4.8) indicate that the RIEKF still is the preferred observer due to less sensitivity to initialization.

4.6.2 Motion Capture Experiment

In order to verify the accuracy of the invariant extended Kalman filter (InEKF) state estimate, we performed a motion capture experiment in the University of Michigan's M-Air facility. This outdoor space is equipped with 18 Qualisys cameras that allows for position tracking. We had the Cassie robot walk untethered for 60 sec along an approximately 15 m path. A top-down view of the estimated trajectory is shown in Figure 4.9. Although there

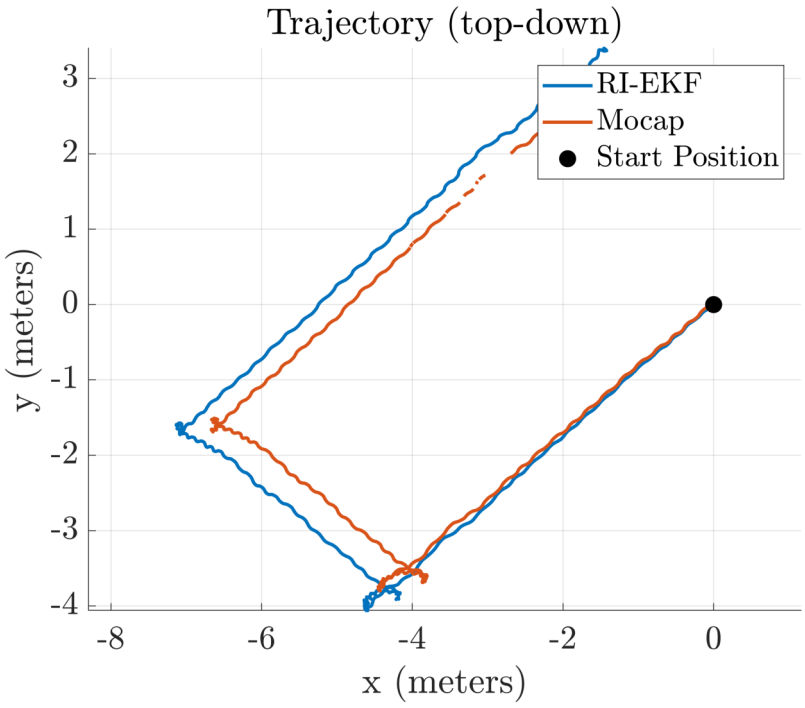


Figure 4.9: Top-down view of the InEKF's estimated trajectory for a motion capture experiment. The position drift is unobservable, however, the final drift is less than 5% of the distance traveled.

is noticeable drift due to the unobservability of the position and yaw, the final position error accounts for less than 5% of the distance traveled. This drift error is due to a combination of sensor noise and imperfect modeling of the robot's kinematics which may introduce biases to the forward kinematic measurements. The orientation, velocity, and position estimates along with their 3σ covariance hulls are shown in Figure 4.10. Due to the unobservability of the yaw angle, the velocity estimate is given in the body frame instead of the world frame. The orientation is plotted using exponential coordinates, $\text{Exp}(\boldsymbol{\phi}) = \mathbf{R}$. Due to an inaccurate orientation estimate from the motion capture system, the "ground truth" for the orientation is given by the VectorNav-100, which runs a state-of-the-art QEKF that fuses angular velocity, linear acceleration, and magnetometer measurements to estimate orientation only. As expected, the error for all observable states remains small. In order to plot the 3σ covariance hull, the right-invariant error covariance needed to be converted to a covariance where the error is defined by Euclidean distance. Up to a first-order approximation, this mapping is done using:

$$\begin{bmatrix} \delta\boldsymbol{\phi}_t \\ \delta\mathbf{v}_t \\ \delta\mathbf{p}_t \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\Gamma}_1^{-1}(\bar{\boldsymbol{\phi}}_t) & \mathbf{0} & \mathbf{0} \\ (\bar{\mathbf{v}}_t)_\times & -\mathbf{I} & \mathbf{0} \\ (\bar{\mathbf{p}}_t)_\times & \mathbf{0} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^v \\ \boldsymbol{\xi}_t^p \end{bmatrix}, \tag{4.60}$$

where the "Euclidean orientation error" is defined as $\delta\boldsymbol{\phi}_t \triangleq \boldsymbol{\phi}_t - \bar{\boldsymbol{\phi}}_t$, and the velocity and position errors match the QEKF error states (4.38). The matrix $\boldsymbol{\Gamma}_1(\bar{\boldsymbol{\phi}}_t)$ is known as the left Jacobian of SO(3) and has an analytical form. Further explanation and the derivation of the above equation is given in Section 4.11.

4.6.3 Long Odometry Experiment

In addition to providing accurate estimates of states vital for legged robot control (orientation and velocity), this InEKF can also provide reliable odometry for a higher-level mapping or simultaneous localization and mapping (SLAM) system. To demonstrate the accuracy of long-term odometry, we had Cassie walk about 200 m along a sidewalk around the University of Michigan's Wave Field. In total, the walk took 7 minutes and 45 seconds. The estimated path from the InEKF overlaid onto satellite imagery is shown in Figure 4.11. Even though the absolute position is unobservable, the odometry estimate from the InEKF contains low enough drift to keep the estimate on the sidewalk for the duration of the experiment. The final position estimate is within a few meters of the true position, and the yaw drift is imperceptible. This odometry estimate is readily available, as it only depends on inertial, contact, and kinematic data, which barring sensor failure, always exist. It does not require the use of any vision systems that may be susceptible to changes in environment or lighting

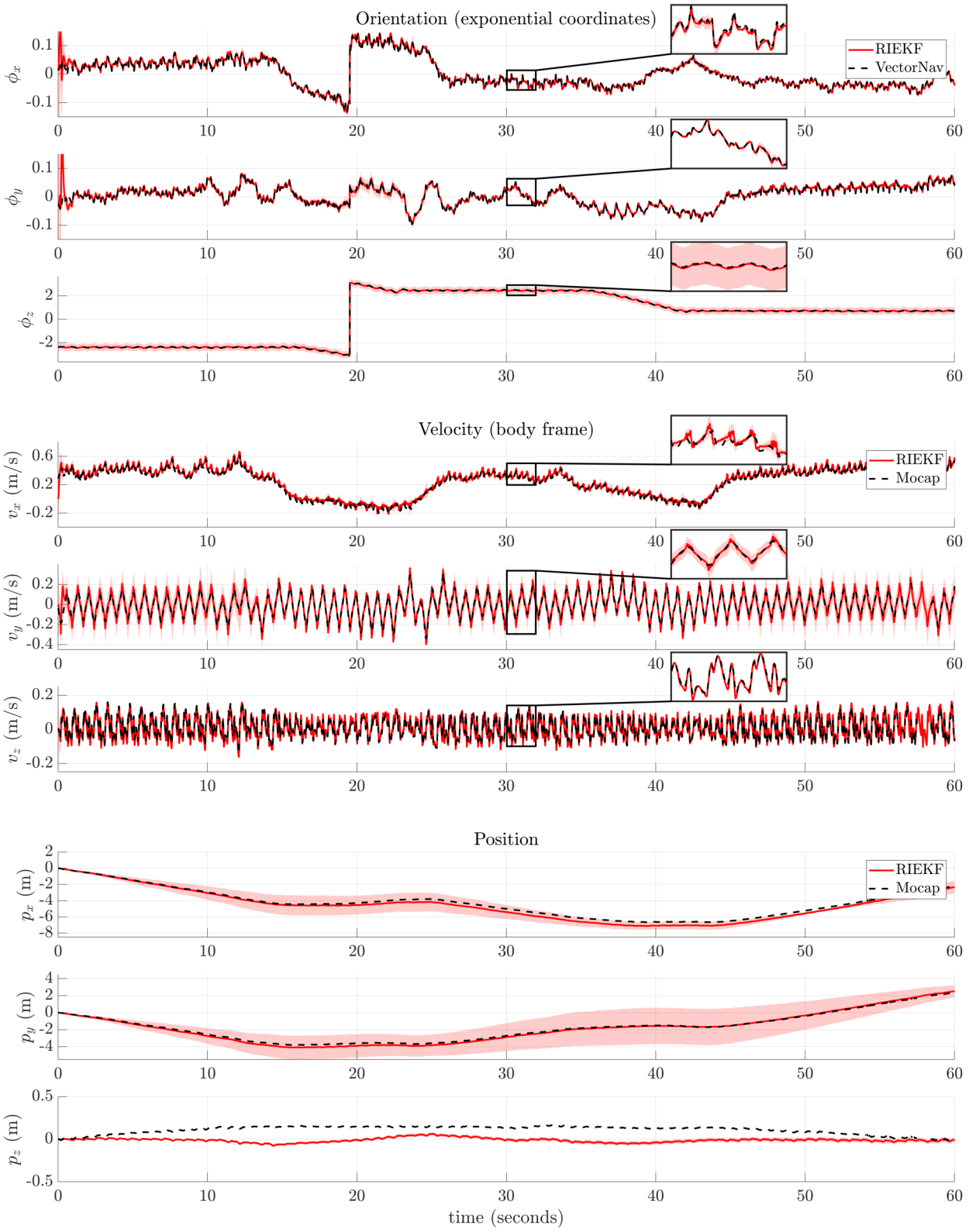


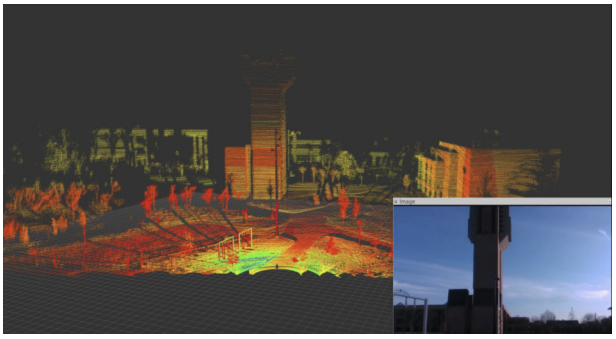
Figure 4.10: Motion capture experiment conducted in the University of Michigan's M-Air facility. The dashed black line represents ground truth, the solid red line is the right-invariant EKF estimate, and the red shaded area represents the 3σ covariance hull. The ground truth for position and velocity were obtained using 18 Qualisys cameras. Due to poor orientation estimates from the motion capture system, the "ground truth" for orientation was obtained from the VectorNav-100, which runs a highly accurate on-board QEKF.

Figure 4.11: Long outdoor odometry experiment where Cassie walked roughly 200 m along a sidewalk over 7 minutes and 45 seconds.

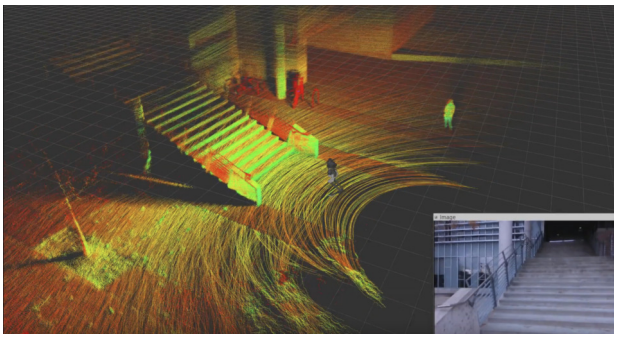
conditions.

4.6.4 LiDAR Mapping Application

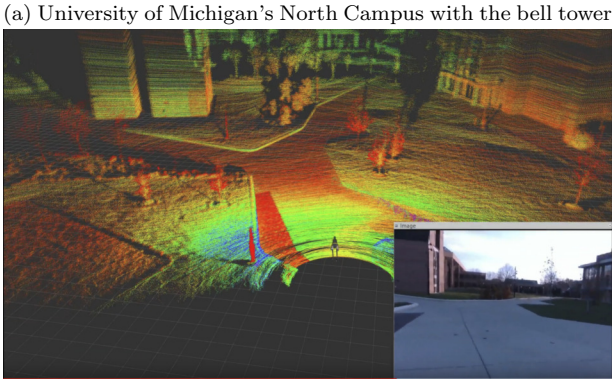
One application for the InEKF odometry is the building of local maps of the environment. We equipped the Cassie-series robot with a new torso that houses a Velodyne VLP-32C LiDAR. With the filter running, we can project each received packet of point cloud data into the world frame based on the current state estimate. This point cloud data can then be accumulated to create a map of the environment. Figure 4.12 shows a few still frames from several LiDAR mapping experiments. A video of these results can be viewed at: `https://youtu.be/pNyXsZ5zVZk`.



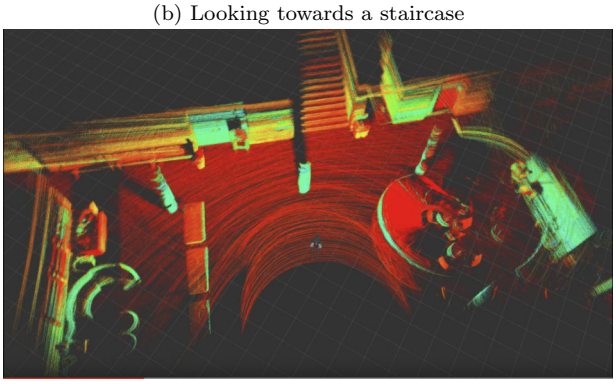
(a) University of Michigan's North Campus with the bell tower



(b) Looking towards a staircase



(c) Walking along a sidewalk



(d) Inside the Bob and Betty Beyster Building

Figure 4.12: LiDAR maps created by transforming 10 seconds of point cloud data onto the pose trajectory estimated by the InEKF. The high frequency odometry estimate allows for motion compensation with a single scan of the LiDAR (10Hz); `https://youtu.be/pNyXsZ5zVZk`

4.7 Alternative Left-Invariant Formulation

For the derivations in Sections 4.2-4.5, we were assuming the use of the right-invariant error. This choice was due to the forward kinematic measurement having the right-invariant observation form. However, it is possible to derive a left-invariant form of this filter, which may be more appropriate to use when dealing with left-invariant observations. For example, GPS measurements are left-invariant observations for the world-centric observer; see Section 4.9. Written explicitly, the left-invariant error is

$$\begin{aligned}\boldsymbol{\eta}_t^l &\triangleq \mathbf{X}_t^{-1}\bar{\mathbf{X}}_t \\ &= \begin{bmatrix} \mathbf{R}_t^\mathsf{T}\bar{\mathbf{R}}_t & \mathbf{R}_t^\mathsf{T}(\bar{\mathbf{v}}_t - \mathbf{v}_t) & \mathbf{R}_t^\mathsf{T}(\bar{\mathbf{p}}_t - \mathbf{p}_t) & \mathbf{R}_t^\mathsf{T}(\bar{\mathbf{d}}_t - \mathbf{d}_t) \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 \end{bmatrix}.\end{aligned} \tag{4.61}$$

After carrying out the chain rule and making the first order approximation, $\boldsymbol{\eta}_t^l = \mathrm{Exp}(\boldsymbol{\xi}_t) \approx \mathbf{I}_d + \boldsymbol{\xi}_t^\wedge$, the individual terms of the left-invariant error dynamics become:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_t^\mathsf{T}\bar{\mathbf{R}}_t &\approx \left(-\left(\tilde{\boldsymbol{\omega}}_t - \boldsymbol{\zeta}_t^g\right)_\times \boldsymbol{\xi}_t^R - \boldsymbol{\zeta}_t^g + \mathbf{w}_t^g\right)_\times \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_t^\mathsf{T}(\bar{\mathbf{v}}_t - \mathbf{v}_t) &\approx -\left(\tilde{\mathbf{a}}_t - \bar{\mathbf{b}}_t^a\right)_\times \boldsymbol{\xi}_t^R \\
&\quad - \left(\tilde{\boldsymbol{\omega}}_t - \bar{\mathbf{b}}_t^g\right)_\times \boldsymbol{\xi}_t^v - \boldsymbol{\zeta}_t^a + \mathbf{w}_t^a \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_t^\mathsf{T}(\bar{\mathbf{p}}_t - \mathbf{p}_t) &\approx \boldsymbol{\xi}_t^v - \left(\tilde{\boldsymbol{\omega}}_t - \bar{\mathbf{b}}_t^g\right)_\times \boldsymbol{\xi}_t^p \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_t^\mathsf{T}(\bar{\mathbf{d}}_t - \mathbf{d}_t) &\approx -\left(\tilde{\boldsymbol{\omega}}_t - \bar{\mathbf{b}}_t^g\right)_\times \boldsymbol{\xi}_t^d + \boldsymbol{h}_R(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v.
\end{aligned} \tag{4.62}$$

Using these results, the log-linear left-invariant dynamics can be expressed using the following linear system

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\xi}_t = \mathbf{A}_t\boldsymbol{\xi}_t + \mathbf{w}_t \\
\implies &\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{P}_t = \mathbf{A}_t\mathbf{P}_t + \mathbf{P}_t\mathbf{A}_t^\mathsf{T} + \bar{\mathbf{Q}}_t,
\end{aligned} \tag{4.63}$$

where the dynamics and noise matrices are

$$\begin{aligned}
\mathbf{A}_t^l &= \begin{bmatrix}
-\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\
-\left(\bar{\mathbf{a}}_t\right)_\times & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} \\
\mathbf{0} & \mathbf{I} & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{bmatrix} \\
\bar{\mathbf{Q}}_t &= \mathrm{Cov}(\mathbf{w}_t).
\end{aligned} \tag{4.64}$$

Similar to the right-invariant case, the dynamics only depend on the state through the IMU bias. When a left-invariant observation comes in, the state estimate is corrected using

$$\left(\bar{\mathbf{X}}_t^+, \boldsymbol{\theta}_t^+\right) = \left(\bar{\mathbf{X}}_t \mathrm{Exp}\left(\mathbf{K}_t^{\boldsymbol{\xi}}\boldsymbol{\Pi}\bar{\mathbf{X}}_t^{-1}\mathbf{Y}_t\right),\ \bar{\boldsymbol{\theta}}_t + \mathbf{K}_t^{\boldsymbol{\zeta}}\boldsymbol{\Pi}\bar{\mathbf{X}}_t^{-1}\mathbf{Y}_t\right), \tag{4.65}$$

where the exponential map is now multiplied on the right side [24].

4.7.1 Switching Between Left and Right-Invariant Errors

Because forward kinematic measurements have the right invariant observation form, the innovation equations are only autonomous when the right invariant error is used. Fortunately, it is possible to switch between the left and right error forms through the use of the adjoint map.

$$\begin{aligned} \boldsymbol{\eta}_t^r = \bar{\mathbf{X}}_t \mathbf{X}_t^{-1} = \bar{\mathbf{X}}_t \boldsymbol{\eta}_t^l \bar{\mathbf{X}}_t^{-1} \\ \implies \operatorname{Exp}(\boldsymbol{\xi}_t^r) = \bar{\mathbf{X}}_t \operatorname{Exp}(\boldsymbol{\xi}_t^l) \bar{\mathbf{X}}_t^{-1} = \operatorname{Exp}(\operatorname{Ad}_{\bar{\mathbf{X}}_t} \boldsymbol{\xi}_t^l) \\ \implies \boldsymbol{\xi}_t^r = \operatorname{Ad}_{\bar{\mathbf{X}}_t} \boldsymbol{\xi}_t^l \end{aligned} \tag{4.66}$$

This transformation is exact, which means that we can easily switch between the covariance of the left and right invariant errors using

$$\mathbf{P}_t^r = \operatorname{Ad}_{\bar{\mathbf{X}}_t} \mathbf{P}_t^l \operatorname{Ad}_{\bar{\mathbf{X}}_t}^{\mathsf{T}}. \tag{4.67}$$

Therefore, when handling a right-invariant observation, we can map the propagated left-invariant covariance to the right-invariant covariance temporarily, apply the right-invariant update equations (4.55), then map the corrected covariance back to the left-invariant form.

It is also possible to compute the log-linear left-invariant dynamics starting from the right-invariant form. Substituting (4.67) into the (right-invariant) covariance propagation equation (4.26) and solving for the left-invariant covariance yields

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{P}_t^l = &\left(\operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}} \mathbf{A}_t^r \operatorname{Ad}_{\bar{\mathbf{X}}_t} - \operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}} \frac{\mathrm{d}}{\mathrm{d}t} \left(\operatorname{Ad}_{\bar{\mathbf{X}}_t} \right) \right) \mathbf{P}_t^l \\ &+ \mathbf{P}_t^l \left(\operatorname{Ad}_{\bar{\mathbf{X}}_t}^{\mathsf{T}} \mathbf{A}_t^{r\mathsf{T}} \operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}}^{\mathsf{T}} - \frac{\mathrm{d}}{\mathrm{d}t} \left(\operatorname{Ad}_{\bar{\mathbf{X}}_t}^{\mathsf{T}} \right) \operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}}^{\mathsf{T}} \right) \\ &+ \operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}} \bar{\mathbf{Q}}_t^r \operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}}^{\mathsf{T}}. \end{aligned} \tag{4.68}$$

Therefore, we learn that the right and left dynamics and noise matrices are related by the following expressions

$$\begin{aligned} \mathbf{A}_t^l &\triangleq \operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}} \mathbf{A}_t^r \operatorname{Ad}_{\bar{\mathbf{X}}_t} - \operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}} \frac{\mathrm{d}}{\mathrm{d}t} \left(\operatorname{Ad}_{\bar{\mathbf{X}}_t} \right) \\ \bar{\mathbf{Q}}_t^l &\triangleq \operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}} \bar{\mathbf{Q}}_t^r \operatorname{Ad}_{\bar{\mathbf{X}}_t^{-1}}^{\mathsf{T}}. \end{aligned} \tag{4.69}$$

Remark 10. Intuitively, the left-invariant error represents an error measured in the body frame of the robot, while the right-invariant error represents an error measured in the world or spatial frame. The frame of measurement dictates whether the exponential map appears on the right or left in the update equations. The error can be moved between these two frames using the adjoint map of the Lie group.

4.7.2 Adding New Contact Points

The process for removing a contact point from the state (marginalization) is identical to the right-invariant error case, described in Section 4.5.1. Likewise, when a new contact is detected, the state can be augmented using the same kinematics relation (4.57) as before. However, due to the change in error variable, the process for augmenting the covariance will be different.

In order to compute the new covariance, we need to look at the left-invariant error,

$$\begin{aligned} \boldsymbol{\eta}_t^d &= \mathbf{R}_t^\mathsf{T}(\bar{\mathbf{d}}_t - \mathbf{d}_t) \\ &= \mathbf{R}_t^\mathsf{T}\left(\bar{\mathbf{p}}_t + \bar{\mathbf{R}}_t\, \boldsymbol{h}_p(\tilde{\boldsymbol{\alpha}}_t)\right) - \mathbf{R}_t^\mathsf{T}\left(\mathbf{p}_t + \mathbf{R}_t\, \boldsymbol{h}_p(\tilde{\boldsymbol{\alpha}}_t - \mathbf{w}_t^\alpha)\right) \\ &\approx \boldsymbol{\eta}_t^p + \boldsymbol{\eta}_t^R\, \boldsymbol{h}_p(\tilde{\boldsymbol{\alpha}}_t) - \boldsymbol{h}_p(\tilde{\boldsymbol{\alpha}}_t) + \mathbf{J}_v(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha \\ \implies \boldsymbol{\xi}_t^d &\approx \boldsymbol{\xi}_t^p - (\, \boldsymbol{h}_p(\tilde{\boldsymbol{\alpha}}_t))_\times\, \boldsymbol{\xi}_t^R + \mathbf{J}_v(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha. \end{aligned} \tag{4.70}$$

Therefore, covariance augmentation can be done using the following linear map,

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^v \\ \boldsymbol{\xi}_t^p \\ \boldsymbol{\xi}_t^d \end{bmatrix} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \\ (-\,\boldsymbol{h}_p(\tilde{\boldsymbol{\alpha}}_t))_\times & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^v \\ \boldsymbol{\xi}_t^p \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{J}_v(\tilde{\boldsymbol{\alpha}}_t) \end{bmatrix} \mathbf{w}_t^\alpha \\ \boldsymbol{\xi}_t^{\text{new}} &\triangleq \mathbf{F}_t\, \boldsymbol{\xi}_t + \mathbf{G}_t \mathbf{w}_t^\alpha \\ \implies \mathbf{P}_t^{\text{new}} &= \mathbf{F}_t\, \mathbf{P}_t\, \mathbf{F}_t^\mathsf{T} + \mathbf{G}_t\, \mathrm{Cov}(\mathbf{w}_t^\alpha)\, \mathbf{G}_t^\mathsf{T}. \end{aligned} \tag{4.71}$$

4.8 Robo-centric Estimator

In this section, we derive a "robot-centric" version of the contact-aided invariant extended Kalman filter (InEKF) where the estimated state is measured in the robot's base (inertial measurement unit (IMU)) frame. When switching to a robot-centric model, the forward kinematics measurements take the left-invariant observation form. In addition, the left/right-invariant error dynamics equations are identical to the world-centric form, albeit swapped.

The right-invariant error dynamics for the world-centric estimator are equivalent to the left-invariant error dynamics for the robo-centric estimator.

The properties of this filter are identical to the right invariant extended Kalman filter (RIEKF) derived in Section 4.2. However, in some cases this filter may be preferred as it directly estimates states that are useful for controlling a legged robot (namely the velocity measured in the body frame).

4.8.1 State and Dynamics

We are interested in estimating the same states as before, though measured in the robot's body frame. Again, the state variables can form a matrix Lie group, $\mathcal{G}$. Specifically, for N contact points, $\mathbf{X}_t \in \mathrm{SE}_{N+2}(3)$ can be represented by the following matrix (which is simply the inverse of the world-centric state)[5]:

$$\mathbf{X}_t \overset{\text{redefine}}{\triangleq} \begin{bmatrix} \mathbf{R}_{\mathrm{BW}}(t) & -{}_{\mathrm{B}}\mathbf{v}_{\mathrm{B}}(t) & {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BW}}(t) & {}_{\mathrm{B}}\mathbf{p}_{\mathrm{C}_1\mathrm{W}}(t) & \cdots & {}_{\mathrm{B}}\mathbf{p}_{\mathrm{C}_N\mathrm{W}}(t) \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.72}$$

Without loss of generality, assume a single contact point. Furthermore, for the sake of readability, we redefine our shorthand notation to be body-centric states,

$$\mathbf{X}_t \overset{\text{redefine}}{\triangleq} \begin{bmatrix} \mathbf{R}_t & \mathbf{v}_t & \mathbf{p}_t & \mathbf{d}_t \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 \end{bmatrix}. \tag{4.73}$$

Using the bias corrected IMU measurements, the individual terms of the new robot-

[5]The negative sign on body velocity appears when inverting the world-centric state; $-{}_{\mathrm{B}}\mathbf{v}_{\mathrm{B}} = -\mathbf{R}_{\mathrm{WB}}^{\mathsf{T}}\,{}_{\mathrm{W}}\mathbf{v}_{\mathrm{B}}$. This sign is removed on the position vectors by swapping the start and end points, ${}_{\mathrm{B}}\mathbf{p}_{\mathrm{BW}} = -\mathbf{R}_{\mathrm{WB}}^{\mathsf{T}}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{WB}}$.

centric system dynamics can be derived as [35]

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_t &= -\left(\bar{\boldsymbol{\omega}}_t - \mathbf{w}_t^g\right)_\times \mathbf{R}_t \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}_t &= -(\bar{\mathbf{a}}_t - \mathbf{w}_t^a) - \mathbf{R}_t\,\mathbf{g} - \left(\bar{\boldsymbol{\omega}}_t - \mathbf{w}_t^g\right)_\times \mathbf{v}_t \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}_t &= \mathbf{v}_t - \left(\bar{\boldsymbol{\omega}}_t - \mathbf{w}_t^g\right)_\times \mathbf{p}_t \\
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{d}_t &= -\left(\bar{\boldsymbol{\omega}}_t - \mathbf{w}_t^g\right)_\times \mathbf{d}_t - \mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v.
\end{aligned} \tag{4.74}$$

Written in matrix form, this becomes

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{X}_t = &\begin{bmatrix}
-\left(\bar{\boldsymbol{\omega}}_t\right)_\times \mathbf{R}_t & -\bar{\mathbf{a}}_t - \mathbf{R}_t\mathbf{g} - \left(\tilde{\boldsymbol{\omega}}_t\right)_\times \mathbf{v}_t & \mathbf{v}_t - \left(\tilde{\boldsymbol{\omega}}_t\right)_\times \mathbf{p}_t & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times \mathbf{d}_t \\
\mathbf{0}_{1\times 3} & 0 & 0 & 0 \\
\mathbf{0}_{1\times 3} & 0 & 0 & 0 \\
\mathbf{0}_{1\times 3} & 0 & 0 & 0
\end{bmatrix} \\
&- \begin{bmatrix}
\left(\mathbf{w}_t^g\right)_\times & \mathbf{w}_t^a & \mathbf{0}_{3\times 1} & \mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v \\
\mathbf{0}_{1\times 3} & 0 & 0 & 0 \\
\mathbf{0}_{1\times 3} & 0 & 0 & 0 \\
\mathbf{0}_{1\times 3} & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\mathbf{R}_t & \mathbf{v}_t & \mathbf{p}_t & \mathbf{d}_t \\
\mathbf{0}_{1\times 3} & 1 & 0 & 0 \\
\mathbf{0}_{1\times 3} & 0 & 1 & 0 \\
\mathbf{0}_{1\times 3} & 0 & 0 & 1
\end{bmatrix} \\
\triangleq\ & f_u(\mathbf{X}_t, \boldsymbol{\theta}_t) - \mathbf{w}_t{}^\wedge \mathbf{X}_t
\end{aligned} \tag{4.75}$$

with $\mathbf{w}_t \triangleq \mathrm{vec}(\mathbf{w}_t^g, \mathbf{w}_t^a, \mathbf{0}_{3\times 1}, \mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v)$. The deterministic dynamics function $f_u(\cdot)$ can be shown to satisfy the group affine property (4.3). Therefore, the left (and right) invariant error dynamics depends solely on the invariant error.

We can derive the log-linear dynamics matrices for the body-centric estimator following a similar derivation process to the world-centric version in Section 4.2. In fact, without IMU bias, the linearization of the left-invariant dynamics for the body-centric estimator is the same as the right-invariant dynamics for the world-centric estimator;

$$\begin{aligned}
\mathbf{A}_t^l \textit{ (body-centric)} &= \mathbf{A}_t^r \textit{ (world-centric)} \\
\mathbf{A}_t^r \textit{ (body-centric)} &= \mathbf{A}_t^l \textit{ (world-centric).}
\end{aligned} \tag{4.76}$$

When IMU bias is included, the above relation still holds, though with the bias terms negated[6]. The noise covariance matrices for the body-centric left/right-invariant propagation

[6]If definition of bias error is negated, even these terms would remain the same.

models are given by:

$$\begin{aligned}\bar{\mathbf{Q}}_t^l &= \begin{bmatrix}\mathrm{Ad}_{\bar{\mathbf{X}}_t^{-1}} & \mathbf{0}_{15,6}\\ \mathbf{0}_{6,15} & \mathbf{I}_6\end{bmatrix} \mathrm{Cov}(\mathbf{w}_t) \begin{bmatrix}\mathrm{Ad}_{\bar{\mathbf{X}}_t^{-1}} & \mathbf{0}_{15,6}\\ \mathbf{0}_{6,15} & \mathbf{I}_6\end{bmatrix}^\mathsf{T}\\ \bar{\mathbf{Q}}_t^r &= \mathrm{Cov}(\mathbf{w}_t),\end{aligned} \tag{4.77}$$

which are also swapped versions of the world-centric noise matrices, after accounting for the redefinition of $\bar{\mathbf{X}}_t$ as its inverse. A comparison of the world-centric and body-centric equations are given in Tables 2 and 3.

4.8.2 Left-Invariant Forward Kinematic Measurement Model

We use forward-kinematics to measure the relative position of the contact point with respect to the body, ${}_\mathrm{B}\mathbf{p}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_t)$. Using the new robot-centric state variables, the measurement model (4.29) becomes

$${}_\mathrm{B}\mathbf{p}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_t) = \mathbf{p}_t - \mathbf{d}_t + {}_\mathrm{B}\mathbf{J}^{\dot{p}}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t. \tag{4.78}$$

Re-written in matrix form, this measurement will have now have the left-invariant observation form (4.7),

$$\underbrace{\begin{bmatrix}{}_\mathrm{B}\mathbf{p}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_t)\\ 0\\ 1\\ -1\end{bmatrix}}_{\mathbf{Y}_t} = \underbrace{\begin{bmatrix}\mathbf{R}_t & \mathbf{v}_t & \mathbf{p}_t & \mathbf{d}_t\\ \mathbf{0}_{1\times3} & 1 & 0 & 0\\ \mathbf{0}_{1\times3} & 0 & 1 & 0\\ \mathbf{0}_{1\times3} & 0 & 0 & 1\end{bmatrix}}_{\mathbf{X}_t} \underbrace{\begin{bmatrix}0\\0\\1\\-1\end{bmatrix}}_{\mathbf{b}} + \underbrace{\begin{bmatrix}{}_\mathrm{B}\mathbf{J}^{\dot{p}}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^\alpha\\ 0\\ 0\\ 0\end{bmatrix}}_{\mathbf{V}_t}. \tag{4.79}$$

The state update equation will take the left-invariant form (4.65), where the matrices $\mathbf{H}_t$ and $\bar{\mathbf{N}}_t$ can be derived to be:

$$\begin{aligned}\mathbf{H}_t &= \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I} & -\mathbf{I}\end{bmatrix},\\ \bar{\mathbf{N}}_t &= \bar{\mathbf{R}}_t^\mathsf{T}\ {}_\mathrm{B}\mathbf{J}^{\dot{p}}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_t)\ \mathrm{Cov}(\mathbf{w}_t^\alpha)\ ({}_\mathrm{B}\mathbf{J}^{\dot{p}}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_t))^\mathsf{T}\ \bar{\mathbf{R}}_t.\end{aligned} \tag{4.80}$$

It is important to note that the forward kinematics measurement is a right-invariant observation for the world-centric estimator, while the same measurement becomes a left-invariant observation for the body-centric version. Also, the above linearization (4.80) is similar to the world-centric, right-invariant one (4.35). Namely, $\mathbf{H}_t$ is simply negated, while $\bar{\mathbf{N}}_t$ is identical after accounting for the redefinition of the state as its inverse.

4.9 Additional Sensor Observations and Filter Summary

This chapter extends upon our conference paper results [109] where we originally presented the contact-aided invariant extended Kalman filter (InEKF). As described in earlier sections, this filter uses an inertial-contact dynamics model with corrections coming from forward kinematics. In addition to forward kinematics, other groups have discovered that a number of measurements common to robotics can also fit the invariant observation model (4.7). Bonnabel [37] developed an invariant observer that uses magnetometer and acceleration measurements to solve the attitude estimation problem. Barczyk and Lynch [18], Barrau [23] described methods for invariant observer design for GPS and magnetometer-aided navigation. Wu et al. [237] developed an InEKF to solve visual-inertial navigation. It has also been shown that an InEKF can be used for simultaneous localization and mapping (SLAM) [243].

In particular, it is interesting to note the similarities between our contact-aided InEKF and landmark-based SLAM. In the simplest case, this SLAM problem involves jointly estimating the robots state along with the position of static landmarks in the environment. The robot is often assumed to have a sensor capable of measuring the position of the landmark relative to the robot. This formulation is identical to our developed InEKF with the contact positions acting as landmarks and forward kinematics measuring the relative translation between the base and contact frames. This similarity was also mentioned by Bloesch et al. [34]. However, there are a few notable differences. The contact frame velocity is assumed to be white noise to allow for foot slip, while landmarks are usually treated as static. Forward kinematics measurements often come at high frequencies (2000 Hz on Cassie). In contrast, landmarks measurements are often at a much lower frequency. Finally, with landmark observations, a data association problem often has to be solved which associates the measurement with a particular landmark state. This problem does not exist with forward kinematic measurements.

Due to the similarities between contacts and landmarks, in our state matrix, the landmark

position states can be easily appended,

$$\mathbf{X}_t \triangleq \begin{bmatrix} \mathbf{R}_{\mathrm{WB}}(t) & {}_{\mathrm{W}}\mathbf{v}_{\mathrm{B}}(t) & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WB}}(t) & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}_1}(t) & \cdots & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}_N}(t) & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WL}_1}(t) & \cdots & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WL}_M}(t) \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix},$$

where each of the N contacts are represented by ${}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}_i}(t)$ and each of the M landmarks are represented by ${}_{\mathrm{W}}\mathbf{p}_{\mathrm{WL}_i}(t)$. In this way, it is possible to develop an observer that contains no unobservable states. Although, like extended Kalman filter (EKF)-SLAM, the filter can become too computationally expensive to run in real-time if the number of landmarks grows too large.

Tables 4.2 and 4.3 give a summary of the left/right world-centric and robo-centric InEKF equations assuming a single contact and landmark position. The linearized observation matrix and observation type for several different sensors are also provided. In these tables, $\mathbf{l}_t$ is shorthand for the true landmark position, and $\mathbf{m}$ denotes the true magnetic field vector. Using these tables, it is clear to see the relation between the left/right invariant error dynamics and the world/robo-centric formulations. All of these dynamics and observations are supported in an open-source C++ Library released alongside this document; available at `https://github.com/RossHartley/invariant-ekf`.

Table 4.2: Summary of World-centric State Estimator

State Definition

$$\mathbf{X}_t \triangleq \begin{bmatrix} \mathbf{R}_{\mathrm{WB}} & {}_{\mathrm{W}}\mathbf{v}_{\mathrm{B}} & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WB}} & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}} & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WL}} \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 1 \end{bmatrix}$$

Deterministic Nonlinear Dynamics

$$f_{\mathbf{u}_t}(\bar{\mathbf{X}}_t, \bar{\boldsymbol{\theta}}_t) = \begin{bmatrix} \bar{\mathbf{R}}_t \left(\bar{\boldsymbol{\omega}}_t\right)_\times & \bar{\mathbf{R}}_t \bar{\mathbf{a}}_t + \mathbf{g} & \bar{\mathbf{v}}_t & \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 0 \end{bmatrix}$$

Log-Linear Right-Invariant Dynamics

$$\mathbf{A}_t^r = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\bar{\mathbf{R}}_t & \mathbf{0} \\ (\mathbf{g})_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\left(\bar{\mathbf{v}}_t\right)_\times \bar{\mathbf{R}}_t & -\bar{\mathbf{R}}_t \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\left(\bar{\mathbf{p}}_t\right)_\times \bar{\mathbf{R}}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\left(\bar{\mathbf{d}}_t\right)_\times \bar{\mathbf{R}}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\left(\bar{\mathbf{l}}_t\right)_\times \bar{\mathbf{R}}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\hat{\mathbf{Q}}_t^r = \begin{bmatrix} \mathrm{Ad}_{\bar{\mathbf{X}}_t} & \mathbf{0}_{15,6} \\ \mathbf{0}_{6,15} & \mathbf{I}_6 \end{bmatrix} \mathrm{Cov}(\mathbf{w}_t) \begin{bmatrix} \mathrm{Ad}_{\bar{\mathbf{X}}_t} & \mathbf{0}_{15,6} \\ \mathbf{0}_{6,15} & \mathbf{I}_6 \end{bmatrix}^\mathsf{T}$$

Log-Linear Left-Invariant Dynamics

$$\mathbf{A}_t^l = \begin{bmatrix} -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ -\left(\bar{\mathbf{a}}_t\right)_\times & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{I} & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\hat{\mathbf{Q}}_t^l = \mathrm{Cov}(\mathbf{w}_t)$$

Measurement	Observation Matrix, $\mathbf{H}$	Observation Type
Forward Kinematic	$\begin{bmatrix}\mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Right-Invariant
Relative Landmark Position	$\begin{bmatrix}\mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Right-Invariant
Absolute Landmark Position	$\begin{bmatrix}(\mathbf{l})_\times & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Right-Invariant
Magnetometer	$\begin{bmatrix}(\mathbf{m})_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Right-Invariant
GPS Position	$\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Left-Invariant

Table 4.3: Summary of Robo-centric State Estimator

State Definition

$$\mathbf{X}_t \triangleq \begin{bmatrix} \mathbf{R}_{\mathrm{BW}} & -{}_{\mathrm{B}}\mathbf{v}_{\mathrm{B}} & {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BW}} & {}_{\mathrm{B}}\mathbf{p}_{\mathrm{CW}} & {}_{\mathrm{B}}\mathbf{p}_{\mathrm{LW}} \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 1 \end{bmatrix}$$

Deterministic Nonlinear Dynamics

$$f_{\mathbf{u}_t}(\bar{\mathbf{X}}_t, \bar{\boldsymbol{\theta}}_t) = \begin{bmatrix} -\left(\bar{\boldsymbol{\omega}}_t\right)_\times \mathbf{R}_t & -\bar{\mathbf{a}}_t - \mathbf{R}_t \mathbf{g} - \left(\bar{\boldsymbol{\omega}}_t\right)_\times \mathbf{v}_t & \mathbf{v}_t - \left(\bar{\boldsymbol{\omega}}_t\right)_\times \mathbf{p}_t & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times \mathbf{d}_t & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times \mathbf{l}_t \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 0 & 0 \end{bmatrix}$$

Log-Linear Right-Invariant Dynamics

$$\mathbf{A}_t^r = \begin{bmatrix} -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\left(\bar{\mathbf{a}}_t\right)_\times & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\left(\bar{\boldsymbol{\omega}}_t\right)_\times & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\hat{\mathbf{Q}}_t^r = \mathrm{Cov}(\mathbf{w}_t)$$

Log-Linear Left-Invariant Dynamics

$$\mathbf{A}_t^l = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \bar{\mathbf{R}}_t & \mathbf{0} \\ (\mathbf{g})_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \left(\bar{\mathbf{v}}_t\right)_\times \bar{\mathbf{R}}_t & \bar{\mathbf{R}}_t \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \left(\bar{\mathbf{p}}_t\right)_\times \bar{\mathbf{R}}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \left(\bar{\mathbf{d}}_t\right)_\times \bar{\mathbf{R}}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \left(\bar{\mathbf{l}}_t\right)_\times \bar{\mathbf{R}}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\hat{\mathbf{Q}}_t^l = \begin{bmatrix} \mathrm{Ad}_{\bar{\mathbf{X}}_t^{-1}} & \mathbf{0}_{15,6} \\ \mathbf{0}_{6,15} & \mathbf{I}_6 \end{bmatrix} \mathrm{Cov}(\mathbf{w}_t) \begin{bmatrix} \mathrm{Ad}_{\bar{\mathbf{X}}_t^{-1}} & \mathbf{0}_{15,6} \\ \mathbf{0}_{6,15} & \mathbf{I}_6 \end{bmatrix}^\mathsf{T}$$

Measurement	Observation Matrix, $\mathbf{H}$	Observation Type
Forward Kinematic	$\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Left-Invariant
Relative Landmark Position	$\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Left-Invariant
Absolute Landmark Position	$\begin{bmatrix}-(\mathbf{l})_\times & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Left-Invariant
Magnetometer	$\begin{bmatrix}-(\mathbf{m})_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Left-Invariant
GPS Position	$\begin{bmatrix}\mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}$	Right-Invariant

4.10 Discretization of Filter Equations

In the preceding sections, the filters equations were in continuous time. However, in order to implement these filters using software and physical sensors, these equations need to be discretized. For our implementation, we assumed a zero-order hold on the inertial measurements, and performed analytical integration [74, 129]. In particular, analytical integration was important for the resulting error dynamics to satisfy Theorem 2.

4.10.1 Discrete World-centric Dynamics

This section demonstrates how to derive the deterministic, discrete time (world-centric) dynamics through analytical integration of the continuous state (4.45) and bias (4.46) dynamics. The contact, landmark, and bias dynamics are simply gaussian noise. Therefore, the discrete, deterministic dynamics are simply:

$$\bar{\mathbf{d}}_{t_{k+1}} = \bar{\mathbf{d}}_{t_k}, \qquad \bar{\mathbf{b}}^g_{t_{k+1}} = \bar{\mathbf{b}}^g_{t_k}, \qquad \bar{\mathbf{b}}^a_{t_{k+1}} = \bar{\mathbf{b}}^a_{t_k}. \tag{4.81}$$

Assuming a zero-order hold on the incoming IMU measurements between times t_k and t_{k+1}, the orientation can be updated using the exponential map of SO(3):

$$\bar{\mathbf{R}}_{t_{k+1}} = \int_{t_k}^{t_{k+1}} \bar{\mathbf{R}}_{t_k} \left(\bar{\boldsymbol{\omega}}_t\right)_\times \, dt = \bar{\mathbf{R}}_{t_k} \mathrm{Exp}\left(\bar{\boldsymbol{\omega}}_{t_k} \Delta t\right) \tag{4.82}$$

where $\Delta t \triangleq t_{k+1} - t_k$. Integrating the velocity dynamics yields an equation that involves the integral of the exponential map:

$$\bar{\mathbf{v}}_{t_{k+1}} = \bar{\mathbf{v}}_{t_k} + \int_{t_k}^{t_{k+1}} \bar{\mathbf{R}}_t \bar{\mathbf{a}}_t + \mathbf{g} \, dt = \bar{\mathbf{v}}_{t_k} + \mathbf{g}\Delta t + \bar{\mathbf{R}}_{t_k} \left(\int_{t_k}^{t_{k+1}} \mathrm{Exp}\left(\bar{\boldsymbol{\omega}}_{t_k} t\right) \, dt \right) \bar{\mathbf{a}}_{t_k}. \tag{4.83}$$

Likewise, analytically solving for the discrete position dynamics involves computing the double integral:

$$\bar{\mathbf{p}}_{t_{k+1}} = \bar{\mathbf{p}}_{t_k} + \bar{\mathbf{v}}_{t_k} \Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 + \bar{\mathbf{R}}_{t_k} \left(\int_{t_k}^{t_{k+1}} \int_{t_k}^{\tau} \mathrm{Exp}\left(\bar{\boldsymbol{\omega}}_{t_k} t\right) \, dt \, d\tau \right) \bar{\mathbf{a}}_{t_k}. \tag{4.84}$$

To solve these integrals, is useful to define an auxiliary function [34]:

$$\boldsymbol{\Gamma}_m(\boldsymbol{\phi}) \triangleq \left(\sum_{n=0}^{\infty} \frac{1}{(n+m)!} \left(\boldsymbol{\phi}\right)^n_\times \right), \tag{4.85}$$

which allows integrals to be easily expressed and computed using the taylor series form of the SO(3) exponential map.

$$\begin{aligned}
&\int_{t_k}^{t_{k+1}} \text{Exp}\left(\boldsymbol{\omega} t\right) dt = \int_{t_k}^{t_{k+1}} \boldsymbol{\Gamma}_0(\bar{\boldsymbol{\omega}}_{t_k} t)\, dt = \left(\sum_{n=0}^{\infty} \frac{1}{(n+1)!} \left(\bar{\boldsymbol{\omega}}_{t_k} \Delta t\right)_{\times}^{n}\right) \Delta t = \boldsymbol{\Gamma}_1(\boldsymbol{\omega}\Delta t)\Delta t \\
&\int_{t_k}^{t_{k+1}} \int_{t_k}^{\tau} \text{Exp}\left(\bar{\boldsymbol{\omega}}_{t_k} t\right)\, dt\, d\tau = \int_{t_k}^{t_{k+1}} \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_{t_k} t)\, t\, dt = \left(\sum_{n=0}^{\infty} \frac{1}{(n+2)!} \left(\bar{\boldsymbol{\omega}}_{t_k} \Delta t\right)_{\times}^{n}\right) \Delta t^2 = \boldsymbol{\Gamma}_2(\boldsymbol{\omega}\Delta t)\Delta t^2
\end{aligned} \tag{4.86}$$

closed-form expressions also exist, allowing fast and easy computation of these quantities [207].

$$\begin{aligned}
\boldsymbol{\Gamma}_0(\boldsymbol{\phi}) &= \mathbf{I}_3 + \frac{\sin(||\boldsymbol{\phi}||)}{||\boldsymbol{\phi}||} (\boldsymbol{\phi})_{\times} + \frac{1-\cos(||\boldsymbol{\phi}||)}{||\boldsymbol{\phi}||^2} (\boldsymbol{\phi})_{\times}^2 \\
\boldsymbol{\Gamma}_1(\boldsymbol{\phi}) &= \mathbf{I}_3 + \frac{1-\cos(||\boldsymbol{\phi}||)}{||\boldsymbol{\phi}||^2} (\boldsymbol{\phi})_{\times} + \frac{||\boldsymbol{\phi}|| - \sin(||\boldsymbol{\phi}||)}{||\boldsymbol{\phi}||^3} (\boldsymbol{\phi})_{\times}^2 \\
\boldsymbol{\Gamma}_2(\boldsymbol{\phi}) &= \frac{1}{2}\mathbf{I}_3 + \frac{||\boldsymbol{\phi}|| - \sin(||\boldsymbol{\phi}||)}{||\boldsymbol{\phi}||^3} (\boldsymbol{\phi})_{\times} + \frac{||\boldsymbol{\phi}||^2 + 2\cos(||\boldsymbol{\phi}||) - 2}{2||\boldsymbol{\phi}||^4} (\boldsymbol{\phi})_{\times}^2
\end{aligned} \tag{4.87}$$

Remark 11. $\boldsymbol{\Gamma}_0(\boldsymbol{\phi})$ is simply the exponential map of SO(3), while $\boldsymbol{\Gamma}_1(\boldsymbol{\phi})$ is also known as the left Jacobian of SO(3) [53, 21].

Using these expressions, we can write down the discrete dynamics for the rotation, velocity, and positions states as:

$$\begin{aligned}
\bar{\mathbf{R}}_{k+1} &= \bar{\mathbf{R}}_k\, \boldsymbol{\Gamma}_0(\bar{\boldsymbol{\omega}}_k \Delta t) \\
\bar{\mathbf{v}}_{k+1} &= \bar{\mathbf{v}}_k + \bar{\mathbf{R}}_k \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\bar{\mathbf{a}}_k \Delta t + \mathbf{g}\Delta t \\
\bar{\mathbf{p}}_{k+1} &= \bar{\mathbf{p}}_k + \bar{\mathbf{v}}_k \Delta t + \bar{\mathbf{R}}_k \boldsymbol{\Gamma}_2(\bar{\boldsymbol{\omega}}_k \Delta t)\bar{\mathbf{a}}_k \Delta t^2 + \frac{1}{2}\mathbf{g}\Delta t^2,
\end{aligned} \tag{4.88}$$

where the t is dropped from the subscript for readability. These discrete dynamics are an exact integration of the continuous-time system under the assumption that the inertial measurement unit (IMU) measurements are constant over Δt.

4.10.2 Discrete Body-centric Dynamics

To compute the body-centric dynamics you simply need to inverse the discrete world-centric dynamics to obtain:

$$\begin{aligned}
\bar{\mathbf{R}}_{k+1} &= \boldsymbol{\Gamma}_0(-\bar{\boldsymbol{\omega}}_k \Delta t)\, \bar{\mathbf{R}}_k \\
\bar{\mathbf{v}}_{k+1} &= \boldsymbol{\Gamma}_0(-\bar{\boldsymbol{\omega}}_k \Delta t)\, \big(\bar{\mathbf{v}}_k - \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\bar{\mathbf{a}}_k \Delta t - \bar{\mathbf{R}}_k \mathbf{g} \Delta t\big) \\
\bar{\mathbf{p}}_{k+1} &= \boldsymbol{\Gamma}_0(-\bar{\boldsymbol{\omega}}_k \Delta t)\Big(\bar{\mathbf{p}}_k + \bar{\mathbf{v}}_k \Delta t \\
&\qquad - \boldsymbol{\Gamma}_2(\bar{\boldsymbol{\omega}}_k \Delta t)\bar{\mathbf{a}}_k \Delta t^2 - \frac{1}{2}\bar{\mathbf{R}}_k \mathbf{g} \Delta t^2\Big).
\end{aligned} \tag{4.89}$$

4.10.3 Discrete Covariance Propagation

In order to propagate the covariance, a continuous-time Riccati equation needs to be solved.

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{P}_t = \mathbf{A}_t \mathbf{P}_t + \mathbf{P}_t \mathbf{A}_t^{\mathsf{T}} + \bar{\mathbf{Q}}_t \tag{4.90}$$

The analytical solution to the differential equation above is given by [165]:

$$\mathbf{P}_{t_{k+1}} = \boldsymbol{\Phi}(t_{k+1}, t_k)\mathbf{P}_{t_k}\boldsymbol{\Phi}(t_{k+1}, t_k)^{\mathsf{T}} + \bar{\mathbf{Q}}_d, \tag{4.91}$$

where the discrete noise covariance matrix is computed by

$$\bar{\mathbf{Q}}_d = \int_{t_k}^{t_{k+1}} \boldsymbol{\Phi}(t, t_k)\bar{\mathbf{Q}}_t \boldsymbol{\Phi}(t, t_k)^{\mathsf{T}} dt, \tag{4.92}$$

and the state transition matrix, $\boldsymbol{\Phi}(t_{k+1}, t_k)$, satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\Phi}(t, t_k) = \mathbf{A}_t \boldsymbol{\Phi}(t, t_k) \quad \text{with} \quad \boldsymbol{\Phi}(t_k, t_k) = \mathbf{I}. \tag{4.93}$$

The (world-centric) left-invariant error dynamics matrix only depends on the IMU inputs and the estimated bias terms, see Table 2. Since both are assumed to be constant between times t_k and t_{k+1}, the state transition matrix can be simply computed from the matrix exponential.

$$\boldsymbol{\Phi}^l(t_{k+1}, t_k) = \mathrm{Exp}_m(\mathbf{A}_t^l \Delta t) \tag{4.94}$$

This state transition matrix also has an analytical solution of the form:

$$\mathbf{\Phi}^l(t_{k+1}, t_k) = \begin{bmatrix} \mathbf{\Phi}^l_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Phi}^l_{15} & \mathbf{0} \\ \mathbf{\Phi}^l_{21} & \mathbf{\Phi}^l_{22} & \mathbf{0} & \mathbf{0} & \mathbf{\Phi}^l_{25} & \mathbf{\Phi}^l_{26} \\ \mathbf{\Phi}^l_{31} & \mathbf{\Phi}^l_{32} & \mathbf{\Phi}^l_{33} & \mathbf{0} & \mathbf{\Phi}^l_{35} & \mathbf{\Phi}^l_{36} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Phi}^l_{44} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{4.95}$$

where the individual terms are

$$\begin{aligned}
\mathbf{\Phi}^l_{11} &= \mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t) \\
\mathbf{\Phi}^l_{21} &= -\mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t)\left(\mathbf{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\bar{\mathbf{a}}_k\right)_\times \Delta t \\
\mathbf{\Phi}^l_{31} &= -\mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t)\left(\mathbf{\Gamma}_2(\bar{\boldsymbol{\omega}}_k \Delta t)\bar{\mathbf{a}}_k\right)_\times \Delta t^2 \\
\mathbf{\Phi}^l_{22} &= \mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t) \\
\mathbf{\Phi}^l_{32} &= \mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t \\
\mathbf{\Phi}^l_{33} &= \mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t) \\
\mathbf{\Phi}^l_{44} &= \mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t) \\
\mathbf{\Phi}^l_{15} &= -\mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t)\mathbf{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t \\
\mathbf{\Phi}^l_{25} &= \mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t)\mathbf{\Psi_1} \\
\mathbf{\Phi}^l_{35} &= \mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t)\mathbf{\Psi_2} \\
\mathbf{\Phi}^l_{26} &= -\mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t)\mathbf{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t \\
\mathbf{\Phi}^l_{36} &= -\mathbf{\Gamma}_0^\mathsf{T}(\bar{\boldsymbol{\omega}}_k \Delta t)\mathbf{\Gamma}_2(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t^2.
\end{aligned}$$

The matrices $\mathbf{\Psi_1}$ and $\mathbf{\Psi_2}$ involve computing the solution to a more complicated integral. However, these integrals still have analytical solutions which can be expressed easier after

defining $\phi \triangleq ||\bar{\boldsymbol{\omega}}_k||$ and $\theta \triangleq \phi \Delta t$.

$$
\begin{aligned}
\boldsymbol{\Psi_1} &\triangleq \int_{t_k}^{t_{k+1}} (\boldsymbol{\Gamma}_0(\bar{\boldsymbol{\omega}}_k t)\bar{\mathbf{a}}_k)_\times \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k t)\, t\, dt \\
&= (\bar{\mathbf{a}}_k)_\times \boldsymbol{\Gamma}_2(-\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t^2 \Big(\frac{\sin(\theta) - \theta\cos(\theta)}{\phi^3} (\bar{\boldsymbol{\omega}}_k)_\times (\bar{\mathbf{a}}_k)_\times \\
&\quad + \frac{\cos(2\theta) - 4\cos(\theta) + 3}{4\phi^4} (\bar{\boldsymbol{\omega}}_k)_\times (\bar{\mathbf{a}}_k)_\times (\bar{\boldsymbol{\omega}}_k)_\times \\
&\quad + \frac{4\sin(\theta) + \sin(2\theta) - 4\theta\cos(\theta) - 2\theta}{4\phi^5} (\bar{\boldsymbol{\omega}}_k)_\times (\bar{\mathbf{a}}_k)_\times (\bar{\boldsymbol{\omega}}_k)_\times^2 \\
&\quad + \frac{\theta^2 - 2\theta\sin(\theta) - 2\cos(\theta) + 2}{2\phi^4} (\bar{\boldsymbol{\omega}}_k)_\times^2 (\bar{\mathbf{a}}_k)_\times \\
&\quad + \frac{6\theta - 8\sin(\theta) + \sin(2\theta)}{4\phi^5} (\bar{\boldsymbol{\omega}}_k)_\times^2 (\bar{\mathbf{a}}_k)_\times (\bar{\boldsymbol{\omega}}_k)_\times \\
&\quad + \frac{2\theta^2 - 4\theta\sin(\theta) - \cos(2\theta) + 1}{4\phi^6} (\bar{\boldsymbol{\omega}}_k)_\times^2 (\bar{\mathbf{a}}_k)_\times (\bar{\boldsymbol{\omega}}_k)_\times^2 \Big)
\end{aligned} \tag{4.96}
$$

$$
\begin{aligned}
\boldsymbol{\Psi_2} &\triangleq \int_{t_k}^{t_{k+1}} \boldsymbol{\Gamma}_0(\bar{\boldsymbol{\omega}}_k t)\boldsymbol{\Phi}_{25}^l(t, t_k)\, dt \\
&= (\bar{\mathbf{a}}_k)_\times \boldsymbol{\Gamma}_3(-\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t^3 \Big(\frac{\theta\sin(\theta) + 2\cos(\theta) - 2}{\phi^4} (\bar{\boldsymbol{\omega}}_k)_\times (\bar{\mathbf{a}}_k)_\times \\
&+ \frac{6\theta - 8\sin(\theta) + sin(2\theta)}{8\phi^5} (\bar{\boldsymbol{\omega}}_k)_\times (\bar{\mathbf{a}}_k)_\times (\bar{\boldsymbol{\omega}}_k)_\times \\
&+ \frac{2\theta^2 + 8\theta\sin(\theta) + 16\cos(\theta) + \cos(2\theta) - 17}{8\phi^6} (\bar{\boldsymbol{\omega}}_k)_\times (\bar{\mathbf{a}}_k)_\times (\bar{\boldsymbol{\omega}}_k)_\times^2 \\
&+ \frac{\theta^3 + 6\theta - 12\sin(\theta) + 6\theta\cos(\theta)}{6\phi^5} (\bar{\boldsymbol{\omega}}_k)_\times^2 (\bar{\mathbf{a}}_k)_\times \\
&+ \frac{6\theta^2 + 16\cos(\theta) - \cos(2\theta) - 15}{8\phi^6} (\bar{\boldsymbol{\omega}}_k)_\times^2 (\bar{\mathbf{a}}_k)_\times (\bar{\boldsymbol{\omega}}_k)_\times \\
&+ \frac{4\theta^3 + 6\theta - 24\sin(\theta) - 3\sin(2\theta) + 24\theta\cos(\theta)}{24\phi^7} (\bar{\boldsymbol{\omega}}_k)_\times^2 (\bar{\mathbf{a}}_k)_\times (\bar{\boldsymbol{\omega}}_k)_\times^2 \Big)
\end{aligned} \tag{4.97}
$$

The (world-centric) right-invariant error dynamics matrix depends on the the state estimates, $\bar{\mathbf{R}}_t$, $\bar{\mathbf{v}}_t$, and $\bar{\mathbf{p}}_t$, which will change between times t_k and t_{k+1}, see Table 2. Therefore, the state transition matrix will not simply be the matrix exponential, as in the left-invariant

case. Solving (4.93) yields a state transition matrix of the form:

$$\boldsymbol{\Phi}^r(t_{k+1}, t_k) = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}^r_{15} & \mathbf{0} \\ \boldsymbol{\Phi}^r_{21} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}^r_{25} & \boldsymbol{\Phi}^r_{26} \\ \boldsymbol{\Phi}^r_{31} & \boldsymbol{\Phi}^r_{32} & \mathbf{I} & \mathbf{0} & \boldsymbol{\Phi}^r_{35} & \boldsymbol{\Phi}^r_{36} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \boldsymbol{\Phi}^r_{45} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{4.98}$$

where the individual terms can be analytically computed as

$$\begin{aligned}
\boldsymbol{\Phi}^r_{21} &= (\mathbf{g})_\times \Delta t \\
\boldsymbol{\Phi}^r_{31} &= \frac{1}{2} (\mathbf{g})_\times \Delta t^2 \\
\boldsymbol{\Phi}^r_{32} &= \mathbf{I}\Delta t \\
\boldsymbol{\Phi}^r_{15} &= -\bar{\mathbf{R}}_k \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t \\
\boldsymbol{\Phi}^r_{25} &= -\left(\bar{\mathbf{v}}_{k+1}\right)_\times \bar{\mathbf{R}}_k \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t + \bar{\mathbf{R}}_k \boldsymbol{\Psi}_\mathbf{1} \\
\boldsymbol{\Phi}^r_{35} &= -\left(\bar{\mathbf{p}}_{k+1}\right)_\times \bar{\mathbf{R}}_k \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t + \bar{\mathbf{R}}_k \boldsymbol{\Psi}_\mathbf{2} \\
\boldsymbol{\Phi}^r_{45} &= -\left(\bar{\mathbf{d}}_{k+1}\right)_\times \bar{\mathbf{R}}_k \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t \\
\boldsymbol{\Phi}^r_{26} &= -\bar{\mathbf{R}}_k \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t \\
\boldsymbol{\Phi}^r_{36} &= -\bar{\mathbf{R}}_k \boldsymbol{\Gamma}_2(\bar{\boldsymbol{\omega}}_k \Delta t)\Delta t^2.
\end{aligned}$$

Since the left/right-invariant errors are related through the adjoint, the two state transition and discrete noise matrices also satisfy the following relations [23].

$$\begin{aligned}
\boldsymbol{\Phi}^r &= \mathrm{Ad}_{\bar{\mathbf{X}}_{k+1}} \boldsymbol{\Phi}^l \mathrm{Ad}_{\bar{\mathbf{X}}_k^{-1}} \\
\bar{\mathbf{Q}}^r_d &= \mathrm{Ad}_{\bar{\mathbf{X}}_{k+1}} \bar{\mathbf{Q}}^l_d \mathrm{Ad}^\mathsf{T}_{\bar{\mathbf{X}}_{k+1}}
\end{aligned} \tag{4.99}$$

Therefore, the right-invariant state transition matrix can alternatively be computed using:

$$\boldsymbol{\Phi}^r(t_{k+1}, t_k) = \mathrm{Ad}_{\bar{\mathbf{X}}_{k+1}} \mathrm{Exp}_m(\mathbf{A}^l_t \Delta t)\, \mathrm{Ad}_{\bar{\mathbf{X}}_k^{-1}}, \tag{4.100}$$

which can simplify implementation since many software libraries already contain efficient methods to compute the matrix exponential.

Similar to the state transition matrices, the discrete noise covariance matrix (4.92) also

has an analytical solution. In practice, this matrix is often approximated as:

$$\bar{\mathbf{Q}}_d \approx \mathbf{\Phi}\bar{\mathbf{Q}}_k\mathbf{\Phi}^\mathsf{T}\Delta t. \tag{4.101}$$

This approximated discrete noise matrix was used for all results in this chapter.

4.11 Error-state Conversions

Throughout this chapter, several versions of error states are used. These include the left/right invariant error (4.2), the quaternion-based extended Kalman filter (QEKF) error states (4.38), and the "Euclidean orientation error" used for plotting the covariance hull (4.60). It is often necessary to covert between these error states for plotting or when initializing the filters to provide fair comparisons. For example, the initial if the QEKF and the invariant extended Kalman filter (InEKF) are initialized with identical covariance matrices, the underlying distribution that they represent may be substantially different. This section provides details on how to convert between these error states up to a first-order approximation.

When designing a QEKF, the orientation error can be defined in either the local or global frame [207]. These errors are equivalent to the left- and right-invariant errors for SO(3). In this chapter, the orientation error in the QEKF was chosen to be the error defined in the local frame (left-invariant error). Since the invariant errors are related through the group's adjoint, the exact relation between right-invariant and QEKF orientation errors is

$$\mathrm{Exp}(\boldsymbol{\xi}_t^R) = \mathrm{Exp}(\bar{\mathbf{R}}_t\delta\boldsymbol{\theta}_t). \tag{4.102}$$

When plotting the individual axes of the orientation error covariance hull, a "Euclidean orientation error" should be used. Let $\delta\boldsymbol{\phi}_t \triangleq \boldsymbol{\phi}_t - \bar{\boldsymbol{\phi}}_t$ be this Euclidean error where $\mathrm{Exp}(\boldsymbol{\phi}) \triangleq \mathbf{R}$ is the exponential coordinate representation of a particular orientation. When the errors are small, a first-order approximation can be used to find a mapping between the right-invariant orientation error and this "Euclidean orientation error".

$$\begin{aligned}\mathrm{Exp}(\boldsymbol{\xi}_t^R) = \bar{\mathbf{R}}_t\mathbf{R}_t^\mathsf{T} &= \mathrm{Exp}(\bar{\boldsymbol{\phi}}_t)\mathrm{Exp}(-\bar{\boldsymbol{\phi}}_t - \delta\boldsymbol{\phi}) \\ &\approx \mathrm{Exp}(-\mathbf{\Gamma}_1(\bar{\boldsymbol{\phi}}_t)\delta\boldsymbol{\phi})\end{aligned} \tag{4.103}$$

A similar first-order approximation can be used to find the relation between the right-

invariant error and the QEKF velocity errors.

$$\begin{aligned}\boldsymbol{\eta}_t^v &= \bar{\mathbf{v}}_t - \bar{\mathbf{R}}_t \mathbf{R}_t^{\mathsf{T}} \mathbf{v}_t = \bar{\mathbf{v}}_t - \mathrm{Exp}(\boldsymbol{\xi}_t^R)\mathbf{v}_t \\ &\approx \bar{\mathbf{v}}_t - \mathbf{v}_t - \left(\boldsymbol{\xi}_t^R\right)_\times \mathbf{v}_t = -\delta\mathbf{v}_t + (\mathbf{v}_t)_\times \boldsymbol{\xi}_t^R \\ \implies \boldsymbol{\xi}_t^v &\approx -\delta\mathbf{v}_t + (\bar{\mathbf{v}}_t)_\times \boldsymbol{\xi}_t^R\end{aligned} \tag{4.104}$$

The same process can be repeated for the position states.

$$\begin{aligned}\boldsymbol{\xi}_t^p &\approx -\delta\mathbf{p}_t + (\bar{\mathbf{p}}_t)_\times \boldsymbol{\xi}_t^R \\ \boldsymbol{\xi}_t^d &\approx -\delta\mathbf{d}_t + \left(\bar{\mathbf{d}}_t\right)_\times \boldsymbol{\xi}_t^R\end{aligned} \tag{4.105}$$

CHAPTER 5

Contact-Aided Smoothing using Factor Graphs

5.1 Overview

The core content in this chapter was previously published in [110, 108]. Co-authors for [110] were Josh Mangelson, Lu Gan, Maani Ghaffari Jadidi, Jeffrey M. Walls, Ryan M. Eustice, and Jessy W Grizzle. Co-authors for [108] included Maani Ghaffari Jadidi, Lu Gan, Jiunn-Kai Huang, Jessy W Grizzle, and Ryan M. Eustice. Several extensions are made, however, much of the text and results remain unchanged.

5.1.1 Motivation and Objective

Long-term state estimation and mapping for legged robots requires a flexible sensor fusion framework that allows for reducing the drift and correcting past estimates as the robot perceives new information. During long-term missions, odometry systems can drift substantially since the absolute position and yaw (rotation about gravity) are unobservable [35] (also see Section 4.2.4), leading to an unbounded growth in the covariance of the estimate and an undesirable expansion of the search space for data association tasks. The factor graph smoothing framework [137, 65, 47, 85] offers suitable machineries for building such systems in which real-time performance is achieved by exploiting the sparse structure of the simultaneous localization and mapping (SLAM) problem [218, 80]. In addition, the incorporation of loop-closures [80] into the graph, upon availability, is convenient.

Legged robot perception often involves fusing leg odometry, inertial measurement unit (IMU) data, and visual/depth measurements to infer the robot trajectory, controller inputs such as velocity, and calibration parameters [196, 151]. The challenge in such perception problems is the rigorous real-time performance requirements in legged robots arising from their direct and switching contact with the environment [35, 82, 33, 81]. Furthermore,

leg odometry involves estimating relative transformations and velocity using kinematic and contact information, which can be noisy due to the encoder noise and foot slip [195].

Towards building a perception system for legged robots suitable for long-term state estimation and mapping, we develop two novel factors that integrate the use of multi-link forward kinematics and the notion of contact between the robotic system and the environment into the factor graph framework. The *forward kinematic factors* relate the base frame to a contact frame through noisy joint encoder measurements. The *hybrid preintegration contact factors* provide odometry measurements of this contact frame over time based on a contact sensor. This factor is formulated to be able to handle an arbitrary number of contact switches between added nodes in the graph. Unlike the originally developed contact factor [110], the proposed hybrid modeling approach reduces the number of required variables in the nonlinear optimization problem by only requiring new states to be added alongside camera or selected keyframes.

5.1.2 State Representation

As with typical inertial navigation [160, 85, 240] and our contact-aided invariant extended Kalman filter (InEKF), developed in Chapter 4, we are interested in estimating the robot's base pose, $\mathbf{H}_{\mathrm{WB}} \in \mathrm{SE}(3)$, and velocity, ${}_{\mathrm{W}}\mathbf{v}_{\mathrm{B}} \in \mathbb{R}^3$, measured in the world frame. In order to later formulate the forward kinematicand contact factors, we also want to estimate the pose of a contact frame relative to the world frame, denoted $\mathbf{H}_{\mathrm{WC}} \in \mathrm{SE}(3)$. In addition, to enable practical applications with imperfect IMUs, we include the parameter vector, $\mathbf{b} \triangleq \mathrm{vec}\left(\mathbf{b}^a, \mathbf{b}^g\right) \in \mathbb{R}^6$, in our state, where $\mathbf{b}^a \in \mathbb{R}^3$ and $\mathbf{b}^g(t) \in \mathbb{R}^3$ are the accelerometer and gyroscope biases, respectively. All together, the state at any time t_i is a tuple as follows:

$$\mathcal{T}_i \triangleq \left(\mathbf{H}_{\mathrm{WB}}(t_i), {}_{\mathrm{W}}\mathbf{v}_{\mathrm{WB}}(t_i), \mathbf{H}_{\mathrm{WC}}(t_i), \mathbf{b}(t_i)\right) \triangleq \left(\mathbf{X}_i, \mathbf{v}_i, \mathbf{C}_i, \mathbf{b}_i\right). \tag{5.1}$$

Further, it is convenient to denote the discrete trajectory of the state variables up to time step k by $\mathcal{X}_k \triangleq \bigcup_{i=1}^{k} \mathcal{T}_i$. Unlike filtering methods, which typically estimate only the current state, smoothing methods often aim to estimate the complete state trajectory. Therefore, the main goal in this chapter is to estimate $\mathcal{X}_k$.

5.1.3 Factor Graph Formulation

Overall, we aim to incorporate forward kinematic and contact measurements into existing factor graphs that may already include visual and inertial measurements. A illustration of this factor graph is shown in Figure 5.1. Let $\mathcal{K}_k$ be the index set of time steps (also called

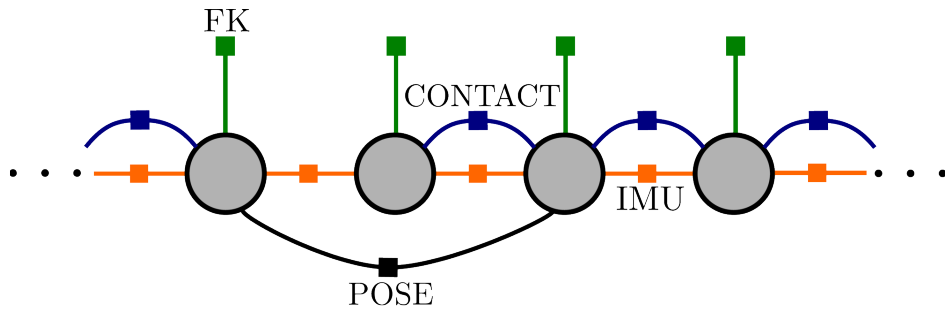


Figure 5.1: An example factor graph for the proposed system. Forward kinematic factors are added at each node and constrain the pose of the contact frames on the feet of the robot with respect to the robot base. Contact factors are added to the graph over time steps where at least one (potentially switching) contact frame remains in contact with the environment. This framework enables the system to handle failures of the visual tracking or loop closure system (denoted here by general pose constraints).

nodes or *keyframes*) up to time step k. The rate at which keyframes are added to the factor graph is a design parameter, however, it is typically chosen based on the rate of the slowest sensor.

Let $\mathcal{L}_{ij} \in \mathrm{SE}(3)$ be a relative odometry or loop-closure measurement relating poses at time steps i and j ($j > i$) computed from an independent perceptual sensor. Examples of this type of measurement include the output of point cloud matching algorithms, such as iterative closest point (ICP) [29], or the relative transformations computed from visual odometry algorithms [86]. Most IMUs provide measurements at a higher frequency than the keyframe frequency. Therefore, between any two time steps i and j, we denote the set of all IMUs (inertial) measurements by $\mathcal{I}_{ij}$. The forward kinematicmeasurement at time step i is denoted by $\mathcal{F}_i$. This measurement originates from high-frequency joint encoders, however, these measurements are only needed at the slower keyframe frequency. Lastly, we denote the set of high-frequency contact measurements between any two time steps i and j, by $\mathcal{C}_{ij}$.

We denote the set of all measurements (loop-closure, inertial, forward-kinematic, and contact) up to time step k by $\mathcal{Z}_k \triangleq \{\mathcal{L}_{ij}, \mathcal{I}_{ij}, \mathcal{F}_i, \mathcal{C}_{ij}\}_{i,j\in\mathcal{K}_k}$. By assuming the measurements are conditionally independent and are corrupted by additive zero mean white Gaussian noise, the posterior probability of the *full SLAM* problem can be written as $p(\mathcal{X}_k|\mathcal{Z}_k) \propto p(\mathcal{X}_0)p(\mathcal{Z}_k|\mathcal{X}_k)$, where

$$p(\mathcal{Z}_k|\mathcal{X}_k) = \prod_{i,j\in\mathcal{K}_k} p(\mathcal{L}_{ij}|\mathcal{X}_j)p(\mathcal{I}_{ij}|\mathcal{X}_j)p(\mathcal{F}_i|\mathcal{X}_i)p(\mathcal{C}_{ij}|\mathcal{X}_j). \tag{5.2}$$

The *Maximum-A-Posteriori* (MAP) estimate of $\mathcal{X}_k$ can be computed by solving the following optimization problem:

$$\underset{\mathcal{X}_k}{\text{minimize}} \; -\log p(\mathcal{X}_k|\mathcal{Z}_k). \tag{5.3}$$

Due to the Gaussian noise assumption mentioned earlier is this is equivalent to solving the nonlinear least-squares problem,

$$\underset{\mathcal{X}_k}{\text{minimize}} \|\mathbf{r}_0\|^2_{\mathbf{\Sigma}_0} + \sum_{i,j\in\mathcal{K}_k} \|\mathbf{r}_{\mathcal{L}_{ij}}\|^2_{\mathbf{\Sigma}_{\mathcal{L}_{ij}}} + \sum_{i,j\in\mathcal{K}_k} \|\mathbf{r}_{\mathcal{I}_{ij}}\|^2_{\mathbf{\Sigma}_{\mathcal{I}_{ij}}} + \sum_{i\in\mathcal{K}_k} \|\mathbf{r}_{\mathcal{F}_i}\|^2_{\mathbf{\Sigma}_{\mathcal{F}_i}} + \sum_{i,j\in\mathcal{K}_k} \|\mathbf{r}_{\mathcal{C}_{ij}}\|^2_{\mathbf{\Sigma}_{\mathcal{C}_{ij}}}, \quad (5.4)$$

where $\mathbf{r}_0$ and $\mathbf{\Sigma}_0$ represents the prior over the initial state and serves to anchor the graph, $\mathbf{r}_{\mathcal{L}_{ij}}$, $\mathbf{r}_{\mathcal{I}_{ij}}$, $\mathbf{r}_{\mathcal{F}_i}$, $\mathbf{r}_{\mathcal{C}_{ij}}$ are the residual terms associated with the loop closure, IMU, forward-kinematic, and contact measurements respectively, i.e. the error between the measured and predicted values given the state, and $\mathbf{\Sigma}_{\mathcal{L}_{ij}}$, $\mathbf{\Sigma}_{\mathcal{I}_{ij}}$, $\mathbf{\Sigma}_{\mathcal{F}_i}$, $\mathbf{\Sigma}_{\mathcal{C}_{ij}}$ are the corresponding covariance matrices. More information about the basic factor graph formulation is given in the background, Section 3.3

The calculation of the the loop-closure (and/or visual odometry) residuals and covariance, $\mathbf{r}_{\mathcal{L}_{ij}}$ and $\mathbf{\Sigma}_{\mathcal{L}_{ij}}$, will depend on the specific type of measurement, and are out of the scope of this work. The preintegrated IMU residuals and covariance, $\mathbf{r}_{\mathcal{I}_{ij}}$ and $\mathbf{\Sigma}_{\mathcal{I}_{ij}}$, can be found in [85]. The goal of the following sections will be to formulate the residuals and covariances for the forward kinematicand contact measurements, $\mathbf{r}_{\mathcal{F}_i}$, $\mathbf{r}_{\mathcal{C}_{ij}}$, and $\mathbf{\Sigma}_{\mathcal{F}_i}$, $\mathbf{\Sigma}_{\mathcal{C}_{ij}}$.

5.2 Forward Kinematic Pose Factor

In this section, we derive a *forward kinematic pose factor* that relates the base frame to a contact frame through noisy joint encoder measurements.

5.2.1 Frames and Definitions

Forward kinematics refers to the process of computing the relative pose transformation between two frames of a multi-link system. Each individual joint displacement describes how the child link moves with respect to the parent one. This joint displacement can either be an angle (revolute joints) or a distance (prismatic joints). Let $\boldsymbol{\alpha} \in \mathbb{R}^N$ denote the vector of joint displacements for a general robot. More information about forward kinematics can be found in the background, Section 3.5.

Without loss of generality, we define a *base frame* on the robot, denoted B, that is assumed to be collocated with both the IMU and the camera frames. When the robot is in contact with a static environment, we can also define a *contact frame*, denoted C, on the robot at the point of contact. The homogeneous transformation between the base frame and

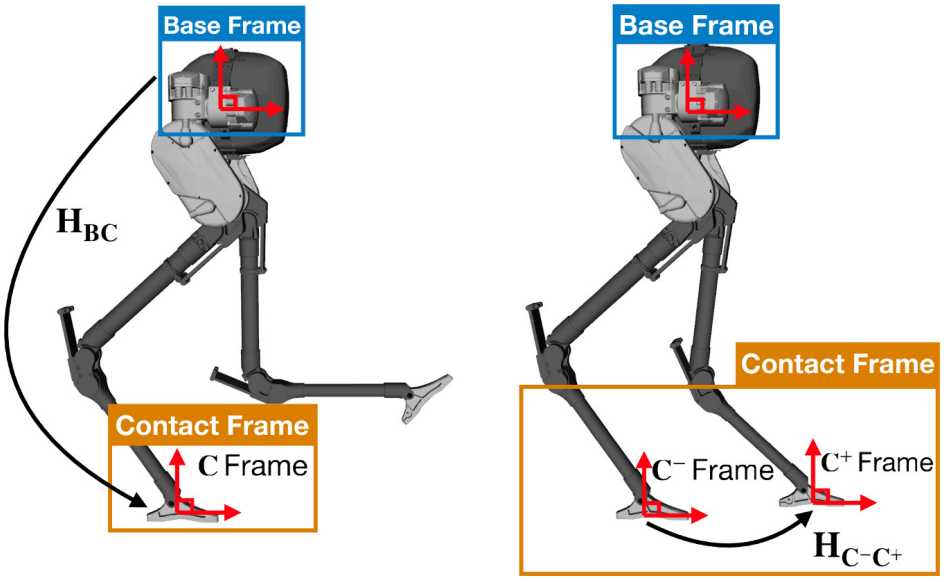


Figure 5.2: In this chapter, we refer to two separate forward kinematics functions. The pose of the current contact frame relative to the base frame is denoted by $\mathbf{H}_{\mathrm{BC}}$. When the robot has multiple points of contact with the environment, it is possible to transfer this contact from from one frame to another. This transfer of contact is captured by the homogeneous transform $\mathbf{H}_{\mathrm{C^-C^+}}$.

the contact frame is defined by

$$\mathbf{H}_{\mathrm{BC}}(\boldsymbol{\alpha}) \triangleq \begin{bmatrix} \mathbf{R}_{\mathrm{BC}}(\boldsymbol{\alpha}) & {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\boldsymbol{\alpha}) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{5.5}$$

where $\mathbf{R}_{\mathrm{BC}}(\boldsymbol{\alpha})$ and ${}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\boldsymbol{\alpha})$ denote the relative orientation and position of the contact frame with respect to the base frame.

When there are two points in contact with the static environment, it is possible to "transfer" the contact frame from one point to the other (shown in Figure 5.2). This will later be useful when preintegrating contact measurements through contact switches. Let C^- denote the old contact frame and C^+ denote the new contact frame. Then, the homogeneous transformation between the old frame and the new frame is defined by

$$\mathbf{H}_{\mathrm{C^-C^+}}(\boldsymbol{\alpha}) \triangleq \begin{bmatrix} \mathbf{R}_{\mathrm{C^-C^+}}(\boldsymbol{\alpha})(\boldsymbol{\alpha}) & {}_{\mathrm{C^-}}\mathbf{p}_{\mathrm{C^-C^+}}(\boldsymbol{\alpha}) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{5.6}$$

where $\mathbf{R}_{\mathrm{C^-C^+}}(\boldsymbol{\alpha})$ and ${}_{\mathrm{C^-}}\mathbf{p}_{\mathrm{C^-C^+}}(\boldsymbol{\alpha})$ denote the relative orientation and position of the new contact frame with respect to the old contact frame.

5.2.2 Noisy Encoder Measurements

We assume that every joint on the robot is equipped with a set of joint encoders that can measure the joint displacement. These encoder measurements, $\tilde{\boldsymbol{\alpha}}$, are assumed to be corrupted with additive Gaussian white noise. This is an explicit measurement coming from physical sensors located on the robot;

$$\tilde{\boldsymbol{\alpha}}(t) = \boldsymbol{\alpha}(t) + \mathbf{w}^{\boldsymbol{\alpha}}(t) \qquad \mathbf{w}^{\boldsymbol{\alpha}}(t) \sim \mathcal{N}(\mathbf{0}_{N\times 1}, \boldsymbol{\Sigma}^{\boldsymbol{\alpha}}(t)). \tag{5.7}$$

The manipulator (or geometric) Jacobian, denoted $\mathbf{J}(\boldsymbol{\alpha})$, provides a method for computing the angular and linear velocity of an end-effector given the vector of joint velocities [171]. In a similar manner, we can use the Jacobian to map incremental angle changes to incremental changes in the end-effector pose. Let ${}_{\mathrm{C}}\mathbf{J}_{\mathrm{BC}}(\boldsymbol{\alpha})$ denote the body manipulator Jacobian of the forward kinematics function (5.5). Then, this relationship can be expressed as

$${}_{\mathrm{C}}\boldsymbol{\xi}_{\mathrm{BC}}\,\delta t = {}_{\mathrm{C}}\mathbf{J}_{\mathrm{BC}}(\boldsymbol{\alpha})\,\delta\boldsymbol{\alpha}, \tag{5.8}$$

where $\delta\boldsymbol{\alpha}$ and $\delta\mathbf{t}$ are incremental encoder and time quantities and ${}_{\mathrm{C}}\boldsymbol{\xi}_{\mathrm{BC}}$ denotes the vector of angular and linear velocities (the twist) of the contact frame due to $\delta\boldsymbol{\alpha}$ (measured in the contact frame). We perform Euler integration using (5.8) to provide a method for factoring out the noise from the forward kinematics equations. Up to a first order approximation, (5.5) can be factored as

$$\mathbf{H}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}(t) - \mathbf{w}^{\boldsymbol{\alpha}}(t)) \approx \mathbf{H}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}(t))\,\mathrm{Exp}\left(-{}_{\mathrm{C}}\mathbf{J}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}(t))\mathbf{w}^{\boldsymbol{\alpha}}(t)\right). \tag{5.9}$$

A similar approximation can be found for (5.6), albeit with a different Jacobian;

$$\mathbf{H}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}(t) - \mathbf{w}^{\boldsymbol{\alpha}}(t)) \approx \mathbf{H}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}(t))\,\mathrm{Exp}\left(-{}_{\mathrm{C^+}}\mathbf{J}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}(t))\mathbf{w}^{\boldsymbol{\alpha}}(t)\right). \tag{5.10}$$

Remark 12. In general, the manipulator Jacobian can be derived as spatial or body manipulator Jacobian [171], and based on this choice, the noise term will appear on the left or right side of the rotation/rigid body transformation, respectively. See Section 3.5 for more information about these Jacobians.

5.2.3 Forward Kinematic Factor

The goal of this section is to derive a general *forward kinematicfactor* that can be used in the factor graph framework. This will be a unary factor that relates two poses (at a single time step) through the forward kinematics equations while accounting for encoder noise. The derivation here assumes that we are relating the base and contact frames.

The pose of the contact frame with respect to the world frame is given by

$$\mathbf{H}_{\mathrm{WC}}(t) = \mathbf{H}_{\mathrm{WB}}(t)\ \mathbf{H}_{\mathrm{BC}}(\boldsymbol{\alpha}(t)). \tag{5.11}$$

Substituting in the state variables at time t_i (5.1), the forward kinematic equations (5.5) yields

$$\mathbf{C}_i = \mathbf{X}_i\ \mathbf{H}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i - \mathbf{w}_i^{\boldsymbol{\alpha}}). \tag{5.12}$$

We can now use the first order approximation (5.9) to factor out the encoder noise to give the following expressions:

$$\mathbf{C}_i = \mathbf{X}_i\ \mathbf{H}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)\ \mathrm{Exp}\left(-{}_{\mathrm{C}}\mathbf{J}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)\mathbf{w}_i^{\boldsymbol{\alpha}}\right) \tag{5.13}$$

Defining the zero-mean white Gaussian forward kinematic noise term, $\delta\mathbf{c}_i \triangleq {}_{\mathrm{C}}\mathbf{J}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)\mathbf{w}_i^{\boldsymbol{\alpha}}$, allows us to write out the *forward kinematicmeasurement model*,

$$\mathbf{H}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i) = \mathbf{X}_i^{-1}\ \mathbf{C}_i\ \mathrm{Exp}(\delta\mathbf{c}_i), \tag{5.14}$$

where the forward kinematics noise is characterized by $\delta\mathbf{c}_i \sim \mathcal{N}(\mathbf{0}_{6\times 1}, \boldsymbol{\Sigma}_{\mathcal{F}_i})$. The residual errors are defined in the tangent space and can be written as

$$\mathbf{r}_{\mathcal{F}i} = \mathrm{Log}\left(\mathbf{C}_i^{-1}\ \mathbf{X}_i\ \mathbf{H}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)\right). \tag{5.15}$$

The covariance is computed through the linear transformation

$$\boldsymbol{\Sigma}_{\mathcal{F}_i} = {}_{\mathrm{C}}\mathbf{J}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)\ \boldsymbol{\Sigma}_i^{\boldsymbol{\alpha}}\ {}_{\mathrm{C}}\mathbf{J}_{\mathrm{BC}}^{\mathsf{T}}(\tilde{\boldsymbol{\alpha}}_i), \tag{5.16}$$

where ${}_{\mathrm{C}}\mathbf{J}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)$ is the body manipulator Jacobian evaluated at the current encoder measurements and $\boldsymbol{\Sigma}_i^{\boldsymbol{\alpha}}$ denotes the encoder covariance matrix at time t_i. This residual and covariance can now be used to insert the *forward kinematic factors* into the factor graph optimization problem (5.4).

5.3 Hybrid Preintegrated Rigid Contact Factor

This section derives the *hybrid preintegration contact factor* that provides contact frame odometry based on measurements from a contact sensor. This factor is formulated to be able to handle an arbitrary number of contact switches between graph nodes.

5.3.1 Contact Dynamics as a Hybrid System

As a legged robot moves through the environment, contact is made and broken numerous times. Therefore, between any two keyframes, the definition of the contact frame may switch (between the left and right foot for example) an arbitrary number of times. This switching process can be captured by modeling the contact dynamics as a *hybrid dynamical system*.

Remark 13. If we assume that the contact frame does not switch between keyframes, then the hybrid nature can be ignored, and the contact dynamics are simplified. This is the approach we took in [110]. However, to enforce the non-switching assumption, keyframes have to be added at every time step where contact with the environment is made or broken. This often leads to large optimization problems that can be difficult to solve in real-time (due to the large number of keyframes). This problem is only exacerbated when dealing with robot's with more legs (imagine how often contact switches occur for a hexapod).

A continuous hybrid dynamical system, $\mathcal{H}$, can defined with a continuous dynamics function, $f(\cdot)$, a discrete transition map, $\Delta(\cdot)$, and a switching surface, $\mathcal{S}$ [233]. The general form of this system can be expressed as

$$\mathcal{H} : \begin{cases} \dot{\mathbf{x}}(t) = f(\mathbf{x}, t) & (\mathbf{x}^-, t^-) \notin \mathcal{S} \\ \mathbf{x}^+ = \Delta(\mathbf{x}^-) & (\mathbf{x}^-, t^-) \in \mathcal{S}. \end{cases} \tag{5.17}$$

Trajectories of the hybrid dynamical system evolve according to the continuous dynamics, until the switching surface it hit. At those moments, the state gets mapped through the discrete transition map, after which the trajectory continues according to the continuous dynamics again.

In general, the contact pose dynamics can be modeled using the hybrid system

$$\mathcal{H} : \begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}\mathbf{H}_{\mathrm{WC}}(t) = \mathbf{H}_{\mathrm{WC}}(t)\ ({}_{\mathrm{C}}\boldsymbol{\xi}_{\mathrm{WC}}(t))^{\wedge} & t^- \notin \mathcal{S} \\ \mathbf{H}_{\mathrm{WC}^+} = \mathbf{H}_{\mathrm{WC}^-}\ \mathbf{H}_{\mathrm{C}^-\mathrm{C}^+} & t^- \in \mathcal{S}, \end{cases} \tag{5.18}$$

where ${}_{\mathrm{C}}\boldsymbol{\xi}_{\mathrm{WC}}(t)$, denotes the twist (angular and linear velocity) of the contact frame, and the switching surface, $\mathcal{S}$ is simply modeled as the set of all times where contact is switched from one frame to another. The matrix $\mathbf{H}_{\mathrm{C^-C^+}}$ denotes the pose of the new contact frame relative to the old contact frame (5.6).

Since the sensor measurements are coming in at discrete time steps, we perform Euler integration from time t to $t+\Delta t$ to discretize the continuous hybrid contact dynamics (5.18) to form the discrete hybrid system

$$\mathcal{H}: \begin{cases} \mathbf{H}_{\mathrm{WC}}(t+\Delta t) = \mathbf{H}_{\mathrm{WC}}(t)\ \mathrm{Exp}\left({}_{\mathrm{C}}\boldsymbol{\xi}_{\mathrm{WC}}(t)\Delta t\right) & t^- \notin \mathcal{S} \\ \mathbf{H}_{\mathrm{WC^+}} = \mathbf{H}_{\mathrm{WC^-}}\ \mathbf{H}_{\mathrm{C^-C^+}} & t^- \in \mathcal{S}, \end{cases} \tag{5.19}$$

where the twist assumed to be constant over Δt. Physically, the continuous dynamics function, f, describes how a single contact frame moves over time while contact is maintained. When a new contact frame is detected, the new contact pose can be computed by applying the transition map, $\Delta(\cdot)$, which utilizes the homogeneous transformation between the old and new contact frames.

5.3.2 Noisy Contact Twist Measurements

The angular and linear velocities of the contact frame are implicit measurements that are *inferred* through a binary contact sensor; specifically, when this sensor indicates contact, the pose of the contact frame is *assumed to remain fixed in the world frame*, i.e. the measured twist is zero. In order to accommodate potential contact slip, this measurement is assumed to be corrupted with additive white Gaussian noise, namely

$${}_{\mathrm{C}}\tilde{\boldsymbol{\xi}}_{\mathrm{WC}}(t) = \mathbf{0}_{6\times 1} = {}_{\mathrm{C}}\boldsymbol{\xi}_{\mathrm{WC}}(t) + \mathbf{w}^{\boldsymbol{\xi}}(t), \quad \mathbf{w}^{\boldsymbol{\xi}}(t) \sim \mathcal{N}(\mathbf{0}_{6\times 1}, \boldsymbol{\Sigma}^{\boldsymbol{\xi}}(t)). \tag{5.20}$$

Using the state variables (5.1), forward kinematics definition (5.6), encoder measurements (5.7), and contact measurements (5.20), the hybrid contact dynamics (5.19) can be written as

$$\mathcal{H}: \begin{cases} \mathbf{C}_{k+1} = \mathbf{C}_k\ \mathrm{Exp}\left(-\mathbf{w}_k^{\boldsymbol{\xi}^d}\Delta t\right) & t_k^- \notin \mathcal{S} \\ \mathbf{C}^+ = \mathbf{C}^-\ \mathbf{H}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_k - \mathbf{w}_k^{\boldsymbol{\alpha}}) & t_k^- \in \mathcal{S} \end{cases}, \tag{5.21}$$

where $\mathbf{w}_k^{\boldsymbol{\xi}^d}$, the discrete time contact noise, is computed using the sample time,

$$\mathrm{Cov}(\mathbf{w}^d(t)) = \frac{1}{\Delta t}\mathrm{Cov}(\mathbf{w}(t)). \tag{5.22}$$

This results in a *rigid* contact model, because without noise, the contact frame's position and orientation are assumed to rigid with respect to the world frame. This is a strict assumption, relaxed only slightly by the addition of velocity noise, that should be validated with the particular robot's gait. However, this method can be applied to many humanoid robots that walk flat-footed. This contact factor is re-derived with a point contact assumption later in Section 5.5.

Remark 14. The equations defining the forward kinematics function, $\mathbf{H}_{\mathrm{C}^-\mathrm{C}^+}(\boldsymbol{\alpha}_k)$, will depend on the specific contact frames, C^- and C^+, at time t_k. For example, the forward kinematics function to switch from left foot contact to a right foot contact will be different than the one used to switch from right foot to left foot. Therefore, when implementing the derived factor, the user needs to determine the specific frames involved in the contact transfer in order to select the appropriate kinematics function.

5.3.3 Preintegrating Contact Pose

The goal of this section is to formulate a general *hybrid preintegrated rigid contact factor* that can be used within an existing factor graph framework. This factor is a binary factor that relates the contact pose at time t_i to the contact pose at time t_j. When combined with the forward kinematicfactor, derived in Section 5.2, the motion of the robot's base frame is further constrained, leading to improved Maximum-A-Posteriori (MAP) state estimation. The hybrid nature of this factor comes from the potential switching of contact that occurs naturally in legged locomotion described in Section 5.3.1. Since contact measurements are typically obtained at a much higher frequency than the keyframes, all measurements between any two keyframes need to be integrated into a single factor. In order to avoid re-integration of these contact measurements every time the state estimate is updated, we employ ideas from inertial measurement unit (IMU) preintegration theory [160, 85, 74] to prevent unnecessary computation allowing efficient implementation of the factor.

Let $\mathcal{S}$ denote the sequence of all time indices associated with contact switches between times t_i and t_j, so that each $s_i \in \mathcal{S}$ represents a single time index where a contact switch occurs. To compute the contact pose at time t_j, we need to integrate the hybrid contact dynamics model (5.21) (starting with the pose at time t_i) through this set of contact switches. This involves integrating the measurements up to the first contact switch time t_{s_i}, applying the transition map $\Delta(\cdot)$, integrating until the next contact switch, applying the next transition map, and so on until time t_j.

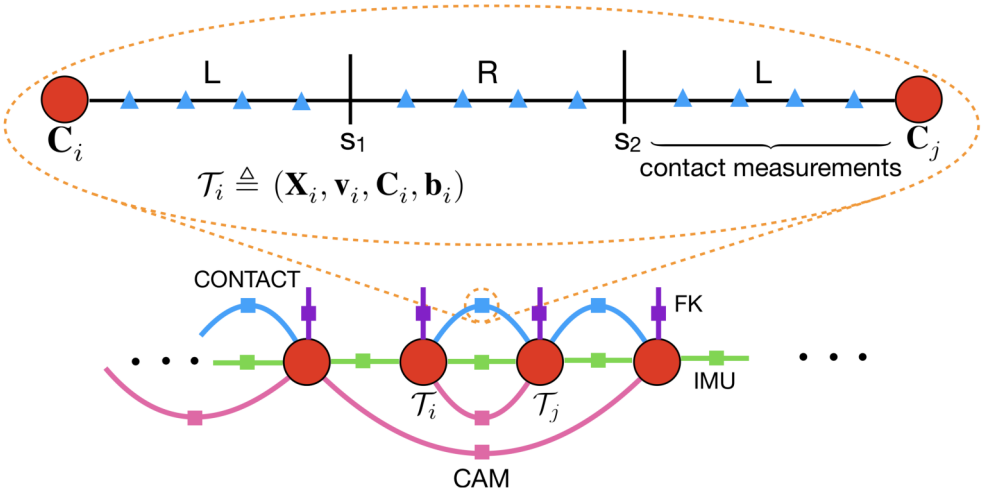


Figure 5.3: In the factor graph framework, we are estimating the robot's state along a discretized trajectory (denoted by red circles). Each independent sensor measurement can provide a "factor" (denoted by lines) that constraints the state at separate time steps. The proposed hybrid contact factor (shown on top) allows preintegration of high-frequency contact data through an arbitrary number of contact switches. In this example, there are two contact switches, where the robot moves from left-stance (L) to right-stance (R), then back to left stance.

For the following derivations, a concrete example (depicted in Figure 5.3) is used to make it easier for the reader to understand. We later use this example to extend the derivation to an arbitrary number of contact switches. For the example shown in Figure 5.3, there are two contact switches between times t_i and t_j (i.e. $|\mathcal{S}| = 2$). Integrating the discrete hybrid contact model (5.21) from t_i to t_j yields

$$\mathbf{C}_j = \mathbf{C}_i \left(\prod_{k=i}^{s_1-1} \mathrm{Exp}(-\mathbf{w}_k^{\boldsymbol{\xi}d} \Delta t) \right) \mathbf{H}_{\mathrm{C}\text{-}\mathrm{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_1} - \mathbf{w}_{s_1}^{\boldsymbol{\alpha}}) \left(\prod_{k=s_1}^{s_2-1} \mathrm{Exp}(-\mathbf{w}_k^{\boldsymbol{\xi}d} \Delta t) \right) \mathbf{H}_{\mathrm{C}\text{-}\mathrm{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_2} - \mathbf{w}_{s_2}^{\boldsymbol{\alpha}}) \left(\prod_{k=s_2}^{j-1} \mathrm{Exp}(-\mathbf{w}_k^{\boldsymbol{\xi}d} \Delta t) \right). \tag{5.23}$$

We then use the manipulator Jacobian (5.10) to factor the noise from the forward kinematics terms along with the adjoint map (4.19) to shift the measured kinematics to the left, resulting in the following expression:

$$\begin{aligned} \mathbf{C}_j = & \mathbf{C}_i \, \mathbf{H}_{\mathrm{C}\text{-}\mathrm{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_1}) \, \mathbf{H}_{\mathrm{C}\text{-}\mathrm{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_2}) \left(\prod_{k=i}^{s_1-1} \mathrm{Exp}(-\mathrm{Ad}_{\mathbf{H}_{\mathrm{C}\text{-}\mathrm{C}^+}^{-1}(\tilde{\boldsymbol{\alpha}}_{s_2})} \mathrm{Ad}_{\mathbf{H}_{\mathrm{C}\text{-}\mathrm{C}^+}^{-1}(\tilde{\boldsymbol{\alpha}}_{s_1})} \mathbf{w}_k^{\boldsymbol{\xi}d} \Delta t) \right) \\ & \mathrm{Exp}(-\mathrm{Ad}_{\mathbf{H}_{\mathrm{C}\text{-}\mathrm{C}^+}^{-1}(\tilde{\boldsymbol{\alpha}}_{s_2})} {}_{\mathrm{C}^+}\mathbf{J}_{\mathrm{C}\text{-}\mathrm{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_1}) \mathbf{w}_{s_1}^{\boldsymbol{\alpha}}) \left(\prod_{k=s_1}^{s_2-1} \mathrm{Exp}(-\mathrm{Ad}_{\mathbf{H}_{\mathrm{C}\text{-}\mathrm{C}^+}^{-1}(\tilde{\boldsymbol{\alpha}}_{s_2})} \mathbf{w}_k^{\boldsymbol{\xi}d} \Delta t) \right) \\ & \mathrm{Exp}(-{}_{\mathrm{C}^+}\mathbf{J}_{\mathrm{C}\text{-}\mathrm{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_2}) \mathbf{w}_{s_2}^{\boldsymbol{\alpha}}) \left(\prod_{k=s_2}^{j-1} \mathrm{Exp}(-\mathbf{w}_k^{\boldsymbol{\xi}d} \Delta t) \right). \end{aligned} \tag{5.24}$$

In the above expression, anytime $\mathbf{C}_i$ changes, we would need to re-integrate to obtain $\mathbf{C}_j$. To prevent this, we multiply both sides by $\mathbf{C}_i^{-1}$, finally arriving at a relative contact pose expression that is independent of states $\mathcal{T}_i$ and $\mathcal{T}_j$,

$$\Delta\mathbf{C}_{ij} \triangleq \mathbf{C}_i^{-1}\mathbf{C}_j = \Delta\tilde{\mathbf{C}}_{ij}\text{Exp}(-\delta\mathbf{c}_{ij}), \tag{5.25}$$

where $\Delta\tilde{\mathbf{C}}_{ij} \triangleq \prod_{k\in\mathcal{S}} \mathbf{H}_{\text{C-C}^+}(\tilde{\boldsymbol{\alpha}}_k)$ represents the *hybrid preintegrated contact measurement*, and $\text{Exp}(-\delta\mathbf{c}_{ij})$ groups all of the noise terms together. Physically, $\Delta\tilde{\mathbf{C}}_{ij}$ corresponds to the measured change in contact pose (from t_i to t_j) as measured by the encoders. The noise term, $\delta\mathbf{c}_{ij}$, corresponds to the cumulative noise coming from both contact slip and the encoders.

We introduce the following Lemma, which allows a product of small group elements to be approximated by a summation in the tangent space. A similar results for 3D rotation group, SO(3), is worked out in [85], however, we state this result for general matrix Lie groups.

Lemma 1 (Approximate Accumulation of Product of Small Group Elements)**.** *Let $\mathcal{G}$ be a matrix Lie group and $\mathfrak{g}$ be its associated Lie Algebra. The product of small group elements (near the identity) can be approximated by a summation of the corresponding elements in the Lie Algebra $\mathfrak{g}$.*

Proof. Let $\text{Exp}(\delta\mathbf{x}_k) \in \mathcal{G}$ be a group element near the identity where $\delta\mathbf{x}_k \in \mathbb{R}^{\dim\mathfrak{g}}$. This result can be seen by first separating one element from the product to yield

$$\begin{aligned}\text{Log}\left(\prod_{k=i}^{j}\text{Exp}(\delta\mathbf{x}_k)\right) &= \text{Log}\left(\left(\prod_{k=i}^{j-1}\text{Exp}(\delta\mathbf{x}_k)\right)\text{Exp}(\delta\mathbf{x}_j)\right)\\ &= \text{Log}\left(\text{Exp}\left(\text{Log}\left(\left(\prod_{k=i}^{j-1}\text{Exp}(\delta\mathbf{x}_k)\right)\right)\right)\text{Exp}(\delta\mathbf{x}_j)\right),\end{aligned} \tag{5.26}$$

which can be approximated using the inverse of the right Jacobian of $\mathcal{G}$, defined in Section 3.4.4,

$$\text{Log}\left(\prod_{k=i}^{j}\text{Exp}(\delta\mathbf{x}_k)\right) \approx \text{Log}\left(\prod_{k=i}^{j-1}\text{Exp}(\delta\mathbf{x}_k)\right) + \mathbf{J}_r^{-1}\left(\text{Log}\left(\prod_{k=i}^{j-1}\text{Exp}(\delta\mathbf{x}_k)\right)\right)\delta\mathbf{x}_j. \tag{5.27}$$

The right Jacobian (and its inverse) can be approximated by the identity matrix since the argument is small. Therefore,

$$\text{Log}\left(\prod_{k=i}^{j}\text{Exp}(\delta\mathbf{x}_k)\right) \approx \text{Log}\left(\prod_{k=i}^{j-1}\text{Exp}(\delta\mathbf{x}_k)\right) + \delta\mathbf{x}_j, \tag{5.28}$$

and the process can be repeated incrementally to show

$$\text{Log}\left(\prod_{k=i}^{j}\text{Exp}(\delta\mathbf{x}_k)\right) \approx \sum_{k=i}^{j}\delta\mathbf{x}_k. \tag{5.29}$$

□

The collected noise term, $\text{Exp}(-\delta\mathbf{c}_{ij})$, is a product of multiple small rigid body transformations, therefore, we can use the results from Lemma 1 to approximate the noise as a summation;

$$\begin{aligned}\delta\mathbf{c}_{ij} \approx \sum_{k=i}^{s_1-1} &\text{Ad}_{\mathbf{H}^{-1}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_2})}\text{Ad}_{\mathbf{H}^{-1}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_1})}\mathbf{w}_k^{\boldsymbol{\xi}^d}\Delta t \\ &+ \text{Ad}_{\mathbf{H}^{-1}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_2})\text{C}^+}\mathbf{J}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_1})\mathbf{w}_{s_1}^{\boldsymbol{\alpha}} + \sum_{k=s_1}^{s_2-1}\text{Ad}_{\mathbf{H}^{-1}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_2})}\mathbf{w}_k^{\boldsymbol{\xi}^d}\Delta t \\ &+ {}_{\text{C}^+}\mathbf{J}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_2})\mathbf{w}_{s_2}^{\boldsymbol{\alpha}} + \sum_{k=s_2}^{j-1}\mathbf{w}_k^{\boldsymbol{\xi}^d}\Delta t.\end{aligned} \tag{5.30}$$

The same result can be obtained by repeatedly using a first-order approximation of the Baker-Campbell-Hausdorff (BCH) formula (3.45). The *hybrid preintegrated contact noise*, $\delta\mathbf{c}_{ij}$, is now a summation of zero-mean Gaussian terms, and is therefore also zero-mean and Gaussian. It is possible to generalize this noise expression to an arbitrary number of contact switches, however, it becomes much simpler to do so when looking at the iterative propagation form in the following section.

5.3.4 Iterative Propagation of Contact Measurements and Noise

It is possible to write both the preintegrated contact measurements, $\Delta\tilde{\mathbf{C}}_{ij}$, and the preintegrated contact noise, $\delta\mathbf{c}_{ij}$, in iterative update forms. This allows the terms to be updated as contact and encoder measurements are coming in. In addition, this form simplifies the expressions and allows for the covariance to be easily computed. This result is captured in the following proposition.

Proposition 3 (Iterative Propagation of Hybrid Contact Process)**.** *Between any two time steps t_i and t_j such that $j > i$, starting with $\Delta\tilde{\mathbf{C}}_{ii} = \mathbf{I}_4$ and $\delta\mathbf{c}_{ii} = \mathbf{0}_{6\times1}$, the hybrid preintegrated contact measurement, $\Delta\tilde{\mathbf{C}}_{ij}$, and noise, $\delta\mathbf{c}_{ij}$, for an arbitrary number of contact switches can be computed iteratively using the following hybrid systems:*

$$\tilde{\mathcal{H}} : \begin{cases} \Delta\tilde{\mathbf{C}}_{ik+1} = \Delta\tilde{\mathbf{C}}_{ik} & t_k^- \notin \mathcal{S} \\ \Delta\tilde{\mathbf{C}}_{ik}^+ = \Delta\tilde{\mathbf{C}}_{ik}^- \, \mathbf{H}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_k) & t_k^- \in \mathcal{S} \end{cases} \tag{5.31}$$

$$\delta\mathcal{H} : \begin{cases} \delta\mathbf{c}_{ik+1} = \delta\mathbf{c}_{ik} + \mathbf{w}_k^{\boldsymbol{\xi}^d}\Delta t & t_k^- \notin \mathcal{S} \\ \delta\mathbf{c}_{ik}^+ = \text{Ad}_{\mathbf{H}_{\text{C}^-\text{C}^+}^{-1}(\tilde{\boldsymbol{\alpha}}_k)}\delta\mathbf{c}_{ik}^- + {}_{\text{C}^+}\mathbf{J}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}_k^{\boldsymbol{\alpha}} & t_k^- \in \mathcal{S}. \end{cases} \tag{5.32}$$

Therefore, when the contact frame remains fixed, the measured relative pose, $\Delta\tilde{\mathbf{C}}_{ik}$, remains constant. When a contact switch occurs, the new contact pose is computed by applying the measured homogeneous transformation, $\mathbf{H}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_k)$, which corresponds to the forward kinematics of the new contact frame relative to the old contact frame. A similar intuitive description of the noise model can be given. When the contact frame remains fixed, the contact noise accumulates additive white noise coming from potential contact slip. When a contact switch occurs, the new noise, $\delta\mathbf{c}_{ik+1}$, is computed by transforming the old noise from the tangent space about $\mathbf{C}^-$ to the tangent space about $\mathbf{C}^+$ using the adjoint map. In addition, an extra noise term, ${}_{\text{C}^+}\mathbf{J}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}_k^{\boldsymbol{\alpha}}$, needs to be added to account for forward kinematics noise. An abstract representation of these hybrid systems is shown in Figure 5.4. The proof for this proposition is developed over the following two sections.

5.3.4.1 Contact Pose

We begin with the preintegrated pose measurement, $\Delta\tilde{\mathbf{C}}_{ij} \triangleq \prod_{k\in\mathcal{S}} \mathbf{H}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_k)$. This relative measurement is simply the product of multiple homogeneous transformations, which is only updated when a contact switch occurs. This is the integration of the following hybrid system that iteratively updates the contact pose measurement, starting with $\Delta\tilde{\mathbf{C}}_{ii} = \mathbf{I}_4$:

$$\begin{cases} \Delta\tilde{\mathbf{C}}_{ik+1} = \Delta\tilde{\mathbf{C}}_{ik} & t_k^- \notin \mathcal{S} \\ \Delta\tilde{\mathbf{C}}_{ik}^+ = \Delta\tilde{\mathbf{C}}_{ik}^- \, \mathbf{H}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_k) & t_k^- \in \mathcal{S}. \end{cases} \tag{5.33}$$

When contact persists, the relative measurement remains the same. When a contact switch occurs, the relative pose is updated by multiplying by a homogeneous transform that represents the new contact frame relative to the old frame.

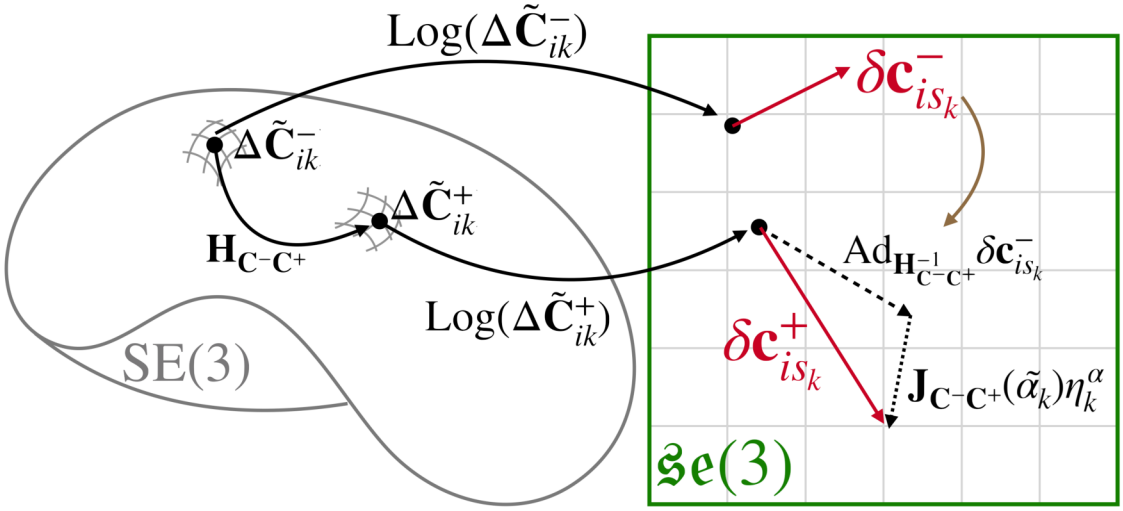


Figure 5.4: When a contact switch occurs, the relative contact pose, $\Delta\tilde{\mathbf{C}}^-_{ik}$, gets mapped from one point in SE(3) to another point, $\Delta\tilde{\mathbf{C}}^+_{ik}$, on the manifold. The contact noise, $\delta\mathbf{c}^-_{ik}$, is represented in the tangent space, $\mathfrak{se}(3)$, and is mapped from the tangent space of $\Delta\tilde{\mathbf{C}}^-_{ik}$ to the tangent space of $\Delta\tilde{\mathbf{C}}^+_{ik}$ through the use of the adjoint map of the forward kinematics transformation, $\mathbf{H}_{\text{C}^-\text{C}^+}$. However, due to noisy encoders, an addition noise term (computed using the manipulator Jacobian) has to be added to compute $\delta\mathbf{c}^+_{ik}$.

5.3.4.2 Contact Noise

In similar fashion, the iterative form of the noise can be derived. We first reorder the approximate noise expression from (5.30). This expression is only for the two contact switch example depicted in Figure 5.3.

$$\delta\mathbf{c}_{ij} \approx \underbrace{\text{Ad}_{\mathbf{H}^{-1}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_2})}\underbrace{\left(\underbrace{\text{Ad}_{\mathbf{H}^{-1}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_1})}\underbrace{\left(\sum_{k=i}^{s_1-1}\mathbf{w}^{\boldsymbol{\xi}d}_k\Delta t\right)}_{\delta\mathbf{c}^-_{is_1}} + {}_{\text{C}^+}\mathbf{J}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_1})\mathbf{w}^{\boldsymbol{\alpha}}_{s_1} + \sum_{k=s_1}^{s_2-1}\mathbf{w}^{\boldsymbol{\xi}d}_k\Delta t}_{\delta\mathbf{c}^+_{is_1}}\right)}_{\delta\mathbf{c}^-_{is_2}} + {}_{\text{C}^+}\mathbf{J}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_{s_2})\mathbf{w}^{\boldsymbol{\alpha}}_{s_2} + \textstyle\sum_{k=s_2}^{j-1}\mathbf{w}^{\boldsymbol{\xi}d}_k\Delta t}_{\delta\mathbf{c}^+_{is_2}} \tag{5.34}$$

Upon close inspection, one can recognize that this is the integration of the following hybrid system from t_i to t_j:

$$\delta\mathcal{H}: \begin{cases} \delta\mathbf{c}_{ik+1} = \delta\mathbf{c}_{ik} + \mathbf{w}^{\boldsymbol{\xi}d}_k\Delta t & t^-_k \notin \mathcal{S} \\ \delta\mathbf{c}^+_{ik} = \text{Ad}_{\mathbf{H}^{-1}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_k)}\delta\mathbf{c}^-_{ik} + {}_{\text{C}^+}\mathbf{J}_{\text{C}^-\text{C}^+}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}^{\boldsymbol{\alpha}}_k & t^-_k \in \mathcal{S}. \end{cases} \tag{5.35}$$

The above expression is now general and can be integrated for an arbitrary number of contact switches to obtain $\delta\mathbf{c}_{ij}$.

5.3.5 Rigid Contact Factor

The relative contact pose expression (5.25) can be used to define the *preintegrated contact measurement model*,

$$\Delta\tilde{\mathbf{C}}_{ij} = \mathbf{C}_i^{-1}\mathbf{C}_j\mathrm{Exp}(\delta\mathbf{c}_{ij}) \qquad \text{with} \qquad \delta\mathbf{c}_{ij} \sim \mathcal{N}(\mathbf{0}_{6\times1}, \boldsymbol{\Sigma}_{\mathcal{C}_{ij}}), \tag{5.36}$$

where the zero-mean Gaussian preintegrated contact noise is characterized by $\delta\mathbf{c}_{ij}$. In the factor graph framework, the preintegrated contact model represents a binary factor that relates the contact frame pose over consecutive time steps. The residual error is defined in the tangent space, and can be written as

$$\mathbf{r}_{\mathcal{C}_{ij}} = \mathrm{Log}\left(\mathbf{C}_j^{-1}\,\mathbf{C}_i\,\Delta\tilde{\mathbf{C}}_{ij}\right). \tag{5.37}$$

The covariance is computed using using the hybrid contact noise model (5.35), starting with $\boldsymbol{\Sigma}_{\mathcal{C}_{ii}} = \mathbf{0}_{6\times6}$;

$$\begin{cases} \boldsymbol{\Sigma}_{\mathcal{C}_{ik+1}} = \boldsymbol{\Sigma}_{\mathcal{C}_{ik}} + \boldsymbol{\Sigma}_k^{\boldsymbol{\xi}}\,\Delta t & t_k^- \notin \mathcal{S} \\ \boldsymbol{\Sigma}_{\mathcal{C}_{ik}}^+ = \mathrm{Ad}_{\mathbf{H}_{\mathrm{C^-C^+}}^{-1}(\tilde{\boldsymbol{\alpha}}_k)}\boldsymbol{\Sigma}_{\mathcal{C}_{ik}}^- \mathrm{Ad}^{\mathsf{T}}_{\mathbf{H}_{\mathrm{C^-C^+}}^{-1}(\tilde{\boldsymbol{\alpha}}_k)} + {}_{\mathrm{C^+}}\mathbf{J}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\,\boldsymbol{\Sigma}_k^{\boldsymbol{\alpha}}\,{}_{\mathrm{C^+}}\mathbf{J}^{\mathsf{T}}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k) & t_k^- \in \mathcal{S}, \end{cases} \tag{5.38}$$

where $\boldsymbol{\Sigma}_k^{\boldsymbol{\xi}}$ denotes the covariance of the continuous contact slip noise and $\boldsymbol{\Sigma}_k^{\boldsymbol{\alpha}}$ denotes the covariance of the encoder noise. This residual and covariance can now be used to insert the *hybrid preintegrated contact factors* into the factor graph optimization problem (5.4).

5.4 Experimental Results on Cassie-series Robot

We now present experimental evaluations of the proposed factors. In the first experiment, we compare three odometry systems composed by visual-inertial-contact (VIC), inertial-contact (IC), and visual-inertial (VI) factors. Since the proposed forward kinematic model is a unary factor, whenever the contact factor is used, it is assumed that forward kinematic factor is also available; therefore, it is not explicitly mentioned for brevity. In the second experiment, we study the effect of losing visual data for a period to see how the contact factor can constrain the graph in the absence of a reliable vision system. In the third experiment, we will evaluate a novel notion called *terrain factor*, where loop-closures can be added to the

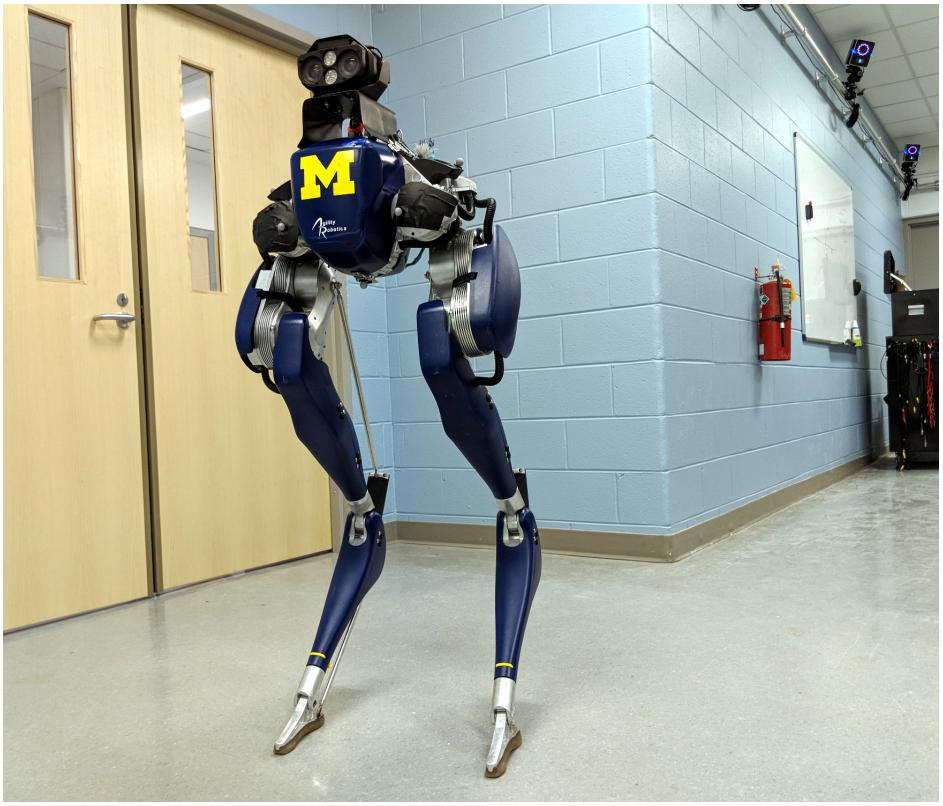


Figure 5.5: Experiments were conducted on a Cassie-series robot designed by Agility Robotics in an indoor laboratory environment. The motion capture system is used to record the robot trajectory as a proxy for ground truth data. The Cassie-series robot has 20 degrees of freedom, 10 actuators, joint encoders, an IMU, and mounted with a Multisense S7 stereo camera.

graph through contact frame poses.

5.4.1 Experimental Setup

All experiments are done on a Cassie-series robot designed by Agility Robotics, which has 20 degrees of freedom, 10 actuators, joint encoders, an IMU, along with a Multisense S7 stereo camera mounted on the top of the Cassie robot, containing another IMU, as shown in Fig. 5.5. Cassie also has four springs (two on each leg) that can be used as a binary contact sensor by thresholding the spring deflection measurements. The Cassie robot has two computers: a MATLAB Simulink Real-Time and a Linux-based system computer. We use the Robot Operating System (ROS) [188] with the User Datagram Protocol (UDP) to communicate sensor data between the two computers. We also integrate the time synchronization algorithm in [178] into our system to ensure all sensory data are synchronized. More information about the Cassie robot can be found in Section 3.1.3.

The motion capture system developed by VICON is used as a proxy for ground truth tra-

jectories. The setup consists of 17 motion capture cameras where four markers are attached to Cassie robot to represent its pelvis as well as two markers to represent the orientation of the IMU on Cassie where the base frame is located. The VICON contains the stereo images (20 Hz) and IMU data (750 Hz) from the Multisense S7 camera as well as the joint encoders and IMU data from the Cassie robot (at 400 Hz each).

The proposed factors are implemented in GTSAM [65], and we used the IMU factor built into GTSAM 4.0 [85] with an incremental solver iSAM2 [137]. We used a semi-direct visual odometry library SVO 2.0 [86] with the Multisense S7 camera. The camera recorded synchronized stereo images at 20 Hz and IMU measurements in about 750 Hz. SVO processes those measurements in real-time and outputs 6 degrees of freedom poses of the left camera in a fixed world frame, $\mathbf{X}_i$, for the current time step i. The relative transformation of the camera from time step i to j can be obtained using $\Delta\mathbf{X}_{ij} = \mathbf{X}_i^{-1}\mathbf{X}_j$. We selected keyframes approximately every 0.25 seconds and added a pose factor for connecting any two corresponding successive keyframes in the graph.

5.4.2 Odometry Comparison

In this experiment, we had Cassie stand in place for about 15 seconds, then slowly walk forwards and backwards along the length of the lab for approximately 45 seconds. The resulting data was used to compare the odometry performance (processed off-line) of different combinations of factors. The results are shown in Figure 5.6. The odometry estimate from VIC, as expected, outperforms all other combinations of factors. The Cumulative Distribution Function (CDF) of the relative position error is shown in Figure 5.7. The relative position CDF provides a method for analyzing the drift of an odometry estimate.

From Figure 5.7, it can be seen that VIC has the highest fraction of data corresponding to smaller relative position errors. This means VIC has lowest drift among all odometry systems. When the robot is walking, the hard impacts cause significant camera shake which leads to motion blur in the images. This effect, along with possibly the lab environment lacking numerous quality features, helps to explain the relatively poor VI odometry performance.

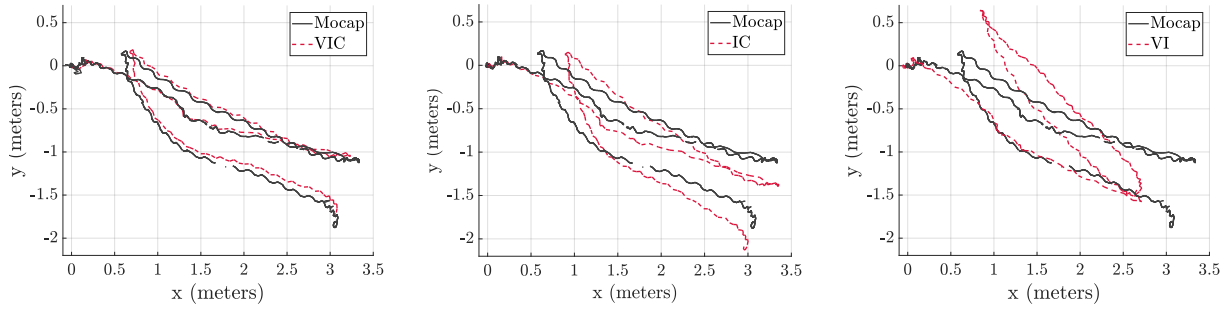


Figure 5.6: The odometry results from a 60 second walking experiment using a Cassie-series robot. The Visual-Inertial-Contact (VIC) odometry outperformed both the Inertial-Contact (IC), and the Visual-Inertial (VI) odometry. "Ground-truth" data was collected from a Vicon motion capture system. It is important to note that no loop-closures are being performed, which helps to explain the relatively poor odometry from VI. The video of this experiment is provided at https://youtu.be/WDPhdl5g2MQ.

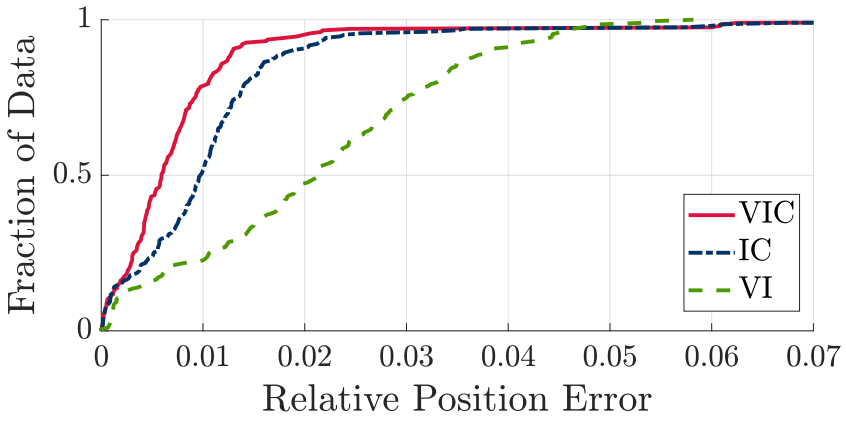


Figure 5.7: The Cumulative Distribution Function (CDF) of the relative position error provides a way to analyze the drift in the odometry estimates from various combinations of factors. The fraction of data corresponding to small relative position errors (low-drift odometry) is the larger for Visual-Inertial-Contact (VIC) odometry than for Inertial-Contact (IC) or Visual-Inertial (VI) odometry.

5.4.3 Vision Dropout

One of the main benefits from including forward kinematic and contact factors is that the state estimator can be more robust to failure of the vision system. In this experiment, we simulate the effects of "vision dropout" by simply ignoring SVO visual-odometry data for two 10-second periods of the experimental data described in the previous section. In other words, during a "vision dropout" period, VIC odometry reduces to IC odometry, and VI odometry reduces down to inertial (I) odometry. Figure 5.8 shows the log determinant of the base pose covariance for VIC and VI for these "vision dropout" experiments. The larger the log determinant, the more uncertain the estimator is about the robot's base pose. During

the "vision dropout" periods, uncertainty grows for VI odometry. This sharp covariance growth is due to the lack of additional sensor measurements to add into the factor graph. In contrast, the covariance growth for VIC is hardly affected over the same dropout periods.

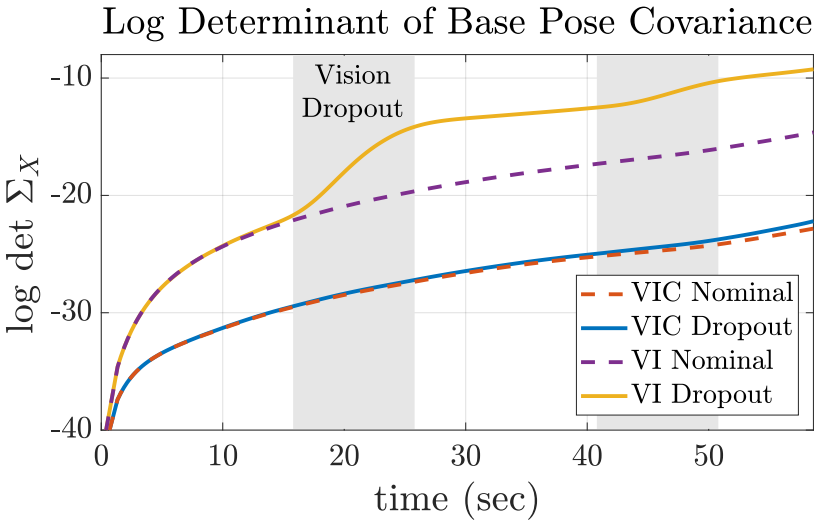


Figure 5.8: When vision data is lost, the covariance of the robot's base pose sharply grows for VI odometry due to the lack of additional measurements to constrain the graph. In contrast, during "vision dropout" periods, the additional contact factors allows the covariance estimate from VIC to remains close to the nominal case.

5.4.4 Terrain Factors as Loop-Closures

Another benefit of adding the proposed contact factors comes from the addition of the contact frame poses into the robot's state. With these new state variables, it becomes simple to to place additional constraints that relate the contact pose to a prior map. We test this idea on the collected experiment data by recognizing that the ground was relatively flat in the laboratory. This "zero-height" elevation data serves as our prior map. Figure 5.9 shows how adding this trivial constraint can reduce position drift in the z-direction. This experiment simply serves to illustrate the potential for "terrain factors", as the state estimate could be further improved if the robot was actually mapping out the terrain; there was actually a small downward slope in the lab (as shown in the motion capture data).

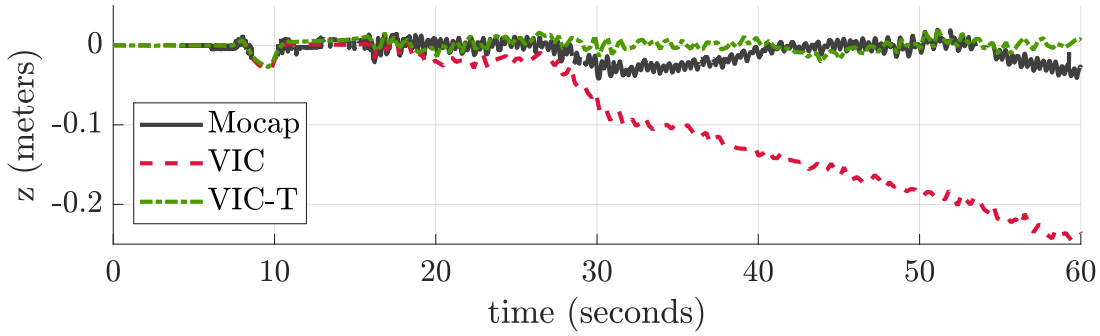


Figure 5.9: Since the contact pose is now part of the estimated state, it is possible to add "terrain factors" that relate this contact pose to a prior map. Adding the simple constraint that the contact frame z-translation is zero (VIC-T) improves the drift in the z-direction when compared to the nominal Visual-Inertial-Contact (VIC) case.

5.5 Hybrid Preintegrated Point Contact Factor

The preintegrated contact factor derived in Section 5.5 assumed that both the contact position and orientation remained fixed with respect to the world frame. For some robots, including Cassie and MARLO, this assumption may be violated. For example, when a robot has point feet, only the position remains fixed. The orientation of the contact frame can freely rotate during a step. Furthermore, since there are no encoders located between the contact frame and the ground, the orientation remains unobservable without the use of a gyroscope or visual sensor. This section aims to derive a hybrid preintegrated *point* contact factor that can be used when this point contact assumption is valid.

5.5.1 Point Contact Dynamics Model

Since the contact orientation is no longer constrained, we can remove it from the state and optimization framework. The state to be optimized now becomes

$$\mathcal{T}_i \triangleq \left(\mathbf{H}_{\mathrm{WB}}(t_i), {}_{\mathrm{W}}\mathbf{v}_{\mathrm{WB}}(t_i), {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}}(t_i), \mathbf{b}(t_i)\right) \triangleq \left(\mathbf{X}_i, \mathbf{v}_i, \mathbf{d}_i, \mathbf{b}_i\right), \tag{5.39}$$

where $\mathbf{d}_i \triangleq {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}}(t_i)$ denotes the contact frame position with respect to the world frame. The hybrid discrete contact dynamics can now be modeled as

$$\mathcal{H}: \begin{cases} {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}}(t+\Delta t) = {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}}(t) + \mathbf{R}_{\mathrm{WC}}(t)\ {}_{\mathrm{C}}\mathbf{v}_{\mathrm{WC}}(t)\Delta t & t^- \notin \mathcal{S} \\ {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}^+} = {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}^-} + \mathbf{R}_{\mathrm{WC}^-}\ {}_{\mathrm{C}^-}\mathbf{p}_{\mathrm{C}^-\mathrm{C}^+} & t^- \in \mathcal{S}. \end{cases} \tag{5.40}$$

The linear velocity of the contact frame is an implicit measurement that is *inferred* through a binary contact sensor; specifically, when this sensor indicates contact, the position of the

contact frame is *assumed to remain fixed in the world frame*, i.e. the measured velocity is zero. In order to accommodate potential contact slip, the measured velocity is assumed to be corrupted with additive white Gaussian noise, namely

$$ {}_{\mathrm{C}}\tilde{\mathbf{v}}_{\mathrm{C}}(t) = \mathbf{0}_{3\times 1} = {}_{\mathrm{C}}\mathbf{v}_{\mathrm{C}}(t) + \mathbf{w}^{v}(t), \quad \mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}_{3\times 1}, \boldsymbol{\Sigma}^{v}(t)), \tag{5.41} $$

where $\mathbf{w}(t)$ is the contact velocity noise represented in the contact frame.

The orientation of the contact frame is not part of the state anymore, so we express it in terms of base orientation, $\mathbf{R}_{\mathrm{WB}}$, and the relative orientation between the base and contact frame, $\mathbf{R}_{\mathrm{BC}}$, coming from forward kinematics. When contact switching occurs, the position of the new contact frame relative to the old contact frame can also be computed using forward kinematics. This leads to the following relations,

$$ \begin{aligned} \mathbf{R}_{\mathrm{WC}}(t) &= \mathbf{R}_{\mathrm{WB}}(t)\,\mathbf{R}_{\mathrm{BC}}(t) = \mathbf{R}_{t}\,\mathbf{R}_{\mathrm{BC}}(\boldsymbol{\alpha}_{t}) \\ \mathbf{R}_{\mathrm{WC^-}}(t)\,{}_{\mathrm{C^-}}\mathbf{p}_{\mathrm{C^-C^+}}(t) &= \mathbf{R}_{\mathrm{WB}}(t)\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}(t)) = \mathbf{R}_{t}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_{t}), \end{aligned} \tag{5.42} $$

where the base orientation, $\mathbf{R}_{t}$, can be extracted from the base pose $\mathbf{X}_{t}$. We can now rewrite the point contact dynamics (5.40) using the state variables (5.39), the velocity measurement (5.41), the encoder measurements (5.7), and the forward kinematicrelations (5.42);

$$ \mathcal{H} : \begin{cases} \mathbf{d}_{k+1} = \mathbf{d}_{k} - \mathbf{R}_{k}\,\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_{k} - \mathbf{w}^{\boldsymbol{\alpha}}_{k})\,\mathbf{w}^{vd}_{k}\Delta t \\ \mathbf{d}^{+}_{k} = \mathbf{d}^{-}_{k} + \mathbf{R}_{k}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_{k} - \mathbf{w}^{\boldsymbol{\alpha}}_{k}) \qquad k \in \mathcal{S}. \end{cases} \tag{5.43} $$

The above equation represents the hybrid point contact dynamics model.

5.5.2 Integrating Contact Position

Integrating the point contact dynamics (5.43) from time t_i to t_j gives,

$$ \mathbf{d}_{j} = \mathbf{d}_{i} + \sum_{k\in\mathcal{S}} \mathbf{R}_{k}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_{k} - \mathbf{w}^{\boldsymbol{\alpha}}_{k}) - \sum_{k=i}^{j-1} \mathbf{R}_{k}\,\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_{k} - \mathbf{w}^{\boldsymbol{\alpha}}_{k})\mathbf{w}^{vd}_{k}\Delta t, \tag{5.44} $$

where the terms arising from all contact switches are grouped together. To avoid having to re-integrate the measurements every time the state is updated, we rearrange terms to give an expression that is independent of the states $\mathcal{T}_i$ and $\mathcal{T}_j$;

$$ \mathbf{R}^{\mathsf{T}}_{i}(\mathbf{d}_{j} - \mathbf{d}_{i}) = \sum_{k\in\mathcal{S}} \Delta\mathbf{R}_{ik}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_{k} - \mathbf{w}^{\boldsymbol{\alpha}}_{k}) - \sum_{k=i}^{j-1} \Delta\mathbf{R}_{ik}\,\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_{k} - \mathbf{w}^{\boldsymbol{\alpha}}_{k})\mathbf{w}^{vd}_{k}\Delta t, \tag{5.45} $$

where $\Delta\mathbf{R}_{ik} \triangleq \mathbf{R}_i^\mathsf{T}\mathbf{R}_k$ denotes the relative change in base orientation from time t_i to time t_k.

For robots with point contact, the relative change in base orientation is unobservable using only forward kinematics and contact measurements. In other words, the robot is free to pivot about its point contact while maintaining constant joint displacements. Therefore, we use gyroscope measurements and inertial measurement unit (IMU) preintegration techniques from [160, 85, 74] to track the relative base orientation. Coming directly from Forster et al. [85], the relative base orientation is given by

$$\Delta\mathbf{R}_{ij} = \prod_{k=i}^{j-1} \mathrm{Exp}\left((\tilde{\boldsymbol{\omega}}_k - \bar{\mathbf{b}}_i^g - \mathbf{w}_k^{gd})\Delta t\right) \approx \Delta\tilde{\mathbf{R}}_{ij}\mathrm{Exp}(-\delta\boldsymbol{\phi}_{ij}), \tag{5.46}$$

where $\tilde{\boldsymbol{\omega}}_k$ denotes the measured angular velocity, $\mathbf{w}_k^{gd} \sim \mathcal{N}(\mathbf{0}_{3\times1}, \boldsymbol{\Sigma}_k^g)$ denotes the discrete gyroscope noise, and $\bar{\mathbf{b}}_i^g$ denotes the gyroscope bias estimate at the time of preintegration. Although this quantity still depends on the bias states, a first-order update can be used to correct these terms [85]. The *preintegrated rotation measurement* is given by

$$\Delta\tilde{\mathbf{R}}_{ij} = \prod_{k=i}^{j-1} \mathrm{Exp}((\tilde{\boldsymbol{\omega}}_k - \bar{\mathbf{b}}_i^g)\Delta t), \tag{5.47}$$

while the Gaussian white noise term is approximated as

$$\delta\boldsymbol{\phi}_{ij} \approx \sum_{k=i}^{j-1} \Delta\tilde{\mathbf{R}}_{k+1,j}^\mathsf{T}\, \mathbf{J}_r(\tilde{\boldsymbol{\omega}}_k - \bar{\mathbf{b}}_i^g)\, \mathbf{w}_k^{gd}\Delta t, \tag{5.48}$$

where $\mathbf{J}_r(\cdot)$ denotes the right Jacobian of SO(3); see Section 3.4.4 and equation (3.53) for more details regarding the right Jacobian.

We can use the manipulator Jacobians to approximate the forward kinematics functions, namely

$$\begin{aligned} \mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_k - \mathbf{w}_k^{\boldsymbol{\alpha}}) &\approx \mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_k)\,\mathrm{Exp}\left(-{}_{\mathrm{C}}\mathbf{J}^{\omega}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}_k^{\boldsymbol{\alpha}}\right) \\ {}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k - \mathbf{w}_k^{\boldsymbol{\alpha}}) &\approx {}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k) - {}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}_k^{\boldsymbol{\alpha}}, \end{aligned} \tag{5.49}$$

where $\mathbf{J}^{\omega}$ denotes the angular velocity component of the manipulator Jacobian, and $\mathbf{J}^{\dot{p}}$ denotes the analytical Jacobian of the forward kinematic position function[1]. Using this forward kinematics approximation (5.49) and the relative base orientation (5.46), we can

[1]This is different than the Jacobian that relates encoder rates to linear velocity. See Section 3.5 for more details.

expand (5.45) to give

$$\begin{aligned}\Delta\mathbf{d}_{ij} &\triangleq \mathbf{R}_i^{\mathsf{T}}(\mathbf{d}_j - \mathbf{d}_i) \\ &= \sum_{k\in\mathcal{S}} \Delta\tilde{\mathbf{R}}_{ik}\,\mathrm{Exp}(-\delta\boldsymbol{\phi}_{ik})\,\left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k) - {}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C\text{-}C^+}}\mathbf{w}^{\boldsymbol{\alpha}}_k\right) \\ &\quad - \sum_{k=i}^{j-1} \Delta\tilde{\mathbf{R}}_{ik}\,\mathrm{Exp}(-\delta\boldsymbol{\phi}_{ik})\,\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_k)\mathrm{Exp}(-{}_{\mathrm{C}}\mathbf{J}^{\omega}_{\mathrm{BC}}\mathbf{w}^{\boldsymbol{\alpha}}_k)\mathbf{w}^{vd}_k\Delta t,\end{aligned} \tag{5.50}$$

Using the first order approximation for the exponential map, $\mathrm{Exp}(\boldsymbol{\phi}) \approx (\mathbf{I} + (\boldsymbol{\phi})_\times)$, we can approximate the above equation as

$$\begin{aligned}\Delta\mathbf{d}_{ij} &\approx \sum_{k\in\mathcal{S}} \Delta\tilde{\mathbf{R}}_{ik}\left(\mathbf{I} - (\delta\boldsymbol{\phi}_{ik})_\times\right)\left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k) - {}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C\text{-}C^+}}\mathbf{w}^{\boldsymbol{\alpha}}_k\right) \\ &\quad - \sum_{k=i}^{j-1} \Delta\tilde{\mathbf{R}}_{ik}\left(\mathbf{I} - (\delta\boldsymbol{\phi}_{ik})_\times\right)\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_k)\left(\mathbf{I} - \left({}_{\mathrm{C}}\mathbf{J}^{\omega}_{\mathrm{BC}}\mathbf{w}^{\boldsymbol{\alpha}}_k\right)_\times\right)\mathbf{w}^{vd}_k\Delta t.\end{aligned} \tag{5.51}$$

Expanding (5.51) and dropping all higher order noise terms yields,

$$\begin{aligned}\Delta\mathbf{d}_{ij} &\approx \sum_{k\in\mathcal{S}} \Delta\tilde{\mathbf{R}}_{ik}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k) - \sum_{k=i}^{j-1} \Delta\tilde{\mathbf{R}}_{ik}\,\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_k)\,\mathbf{w}^{vd}_k\Delta t \\ &\quad + \sum_{k\in\mathcal{S}} \Delta\tilde{\mathbf{R}}_{ik}\left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\right)_\times \delta\boldsymbol{\phi}_{ik} - \sum_{k\in\mathcal{S}} \Delta\tilde{\mathbf{R}}_{ik}\,{}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}^{\boldsymbol{\alpha}}_k.\end{aligned} \tag{5.52}$$

Upon inspection, we can recognize that the above expression for $\Delta\mathbf{d}_{ij}$ is the sum of a measured term and a noise term. Therefore, we can express it as

$$\Delta\mathbf{d}_{ij} = \mathbf{R}_i^{\mathsf{T}}(\mathbf{d}_j - \mathbf{d}_i) \triangleq \Delta\tilde{\mathbf{d}}_{ij} - \delta\mathbf{d}_{ij}, \tag{5.53}$$

where $\Delta\tilde{\mathbf{d}}_{ij}$ represents the *hybrid preintegrated point contact position measurement*, and $\delta\mathbf{d}_{ij}$ groups all the noise terms together. Written explicitly, these terms are

$$\Delta\tilde{\mathbf{d}}_{ij} \triangleq \sum_{k\in\mathcal{S}} \Delta\tilde{\mathbf{R}}_{ik}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_{sk}), \quad \text{and}$$

$$\delta\mathbf{d}_{ij} \triangleq \sum_{k=i}^{j-1} \Delta\tilde{\mathbf{R}}_{ik}\,\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}^{vd}_k\Delta t + \sum_{k\in\mathcal{S}} \Delta\tilde{\mathbf{R}}_{ik}\left({}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}^{\boldsymbol{\alpha}}_k - \left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\right)_\times \delta\boldsymbol{\phi}_{ik}\right). \tag{5.54}$$

The noise is zero-mean and Gaussian since $\delta\mathbf{d}_{ij}$ is the sum of zero-mean Gaussians.

5.5.3 Iterative Propagation

Similar to the rigid contact case (Section 5.3.4), it is possible to derive iterative expressions for the hybrid preintegrated point contact position measurement and noise terms. This iterative form allows both the preintegrated measurement and the covariance to be easily and efficiently updated as new measurements are received.

The iterative update equations for the position measurement (5.53) can be written using a hybrid system;

$$\tilde{\mathcal{H}} : \begin{cases} \Delta\tilde{\mathbf{d}}_{ik+1} = \Delta\tilde{\mathbf{d}}_{ik} \\ \Delta\tilde{\mathbf{d}}_{ik}^{+} = \Delta\tilde{\mathbf{d}}_{ik}^{-} + \Delta\tilde{\mathbf{R}}_{ik}\, {}_{\mathrm{B}}\mathbf{p}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_k) \qquad k \in \mathcal{S}. \end{cases} \tag{5.55}$$

Intuitively, if the contact sensor detects that the previous contact persists, then the contact position measurement remains unchanged. This comes from our assumption that the position of the contact point is fixed in the world frame. If the contact sensor detects a change in contact frames, a separate term is added that represents the relative position change between the new and old contact frames. Similarly, iterative update equations for the position noise (5.53) can also be written as a hybrid system;

$$\delta\mathcal{H} : \begin{cases} \delta\mathbf{d}_{ik+1} = \delta\mathbf{d}_{ik} + \Delta\tilde{\mathbf{R}}_{ik}\, \mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}_k^{vd}\Delta t \\ \delta\mathbf{d}_{ik}^{+} = \delta\mathbf{d}_{ik}^{-} + \Delta\tilde{\mathbf{R}}_{ik}\left({}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_k)\mathbf{w}_k^{\boldsymbol{\alpha}} - ({}_{\mathrm{B}}\mathbf{p}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_k))_\times\, \delta\boldsymbol{\phi}_{ik}\right) \qquad k \in \mathcal{S}. \end{cases} \tag{5.56}$$

When the previous contact persists, the position noise simply accumulates white noise to accommodate potential slipping. This "contact velocity noise" was represented in the contact frame, and therefore has to be rotated using forward kinematics and the relative base orientation. When a contact switch is detected, additional noise terms are added that account for uncertainty in the relative base orientation and uncertainty in the forward kinematics.

Since the noise, $\delta\mathbf{d}_{ij}$, is coupled to the base orientation noise, $\delta\boldsymbol{\phi}_{ij}$, we need to formulate this point contact factor as an extension to the *preintegrated IMU factor* described in [85]. When (5.56) is combined with the iterative preintegrated IMU equations detailed in [84],

the combined hybrid noise system can be expressed as

$$\begin{bmatrix}\delta\boldsymbol{\phi}_{ik+1}\\ \delta\mathbf{v}_{ik+1}\\ \delta\mathbf{p}_{ik+1}\\ \delta\mathbf{d}_{ik+1}\end{bmatrix} = \underbrace{\begin{bmatrix}\Delta\tilde{\mathbf{R}}_{kk+1}^{\mathsf{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ -\Delta\tilde{\mathbf{R}}_{ik}(\bar{\mathbf{a}}_k-\mathbf{b}_i^a)^\wedge\Delta t & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ -\frac{1}{2}\Delta\tilde{\mathbf{R}}_{ik}(\bar{\mathbf{a}}_k-\mathbf{b}_i^a)^\wedge\Delta t^2 & \mathbf{I}\Delta t & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}}_{\mathbf{A}_k}\begin{bmatrix}\delta\boldsymbol{\phi}_{ik}\\ \delta\mathbf{v}_{ik}\\ \delta\mathbf{p}_{ik}\\ \delta\mathbf{d}_{ik}\end{bmatrix} + \underbrace{\begin{bmatrix}\mathbf{J}_r(\tilde{\boldsymbol{\omega}}_k-\mathbf{b}_k^g)\Delta t & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \Delta\tilde{\mathbf{R}}_{ik}\Delta t & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \frac{1}{2}\Delta\tilde{\mathbf{R}}_{ik}\Delta t^2 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \Delta\tilde{\mathbf{R}}_{ik}\mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_k)\Delta t & \mathbf{0}\end{bmatrix}}_{\mathbf{B}_k}\begin{bmatrix}\mathbf{w}_k^{gd}\\ \mathbf{w}_k^{ad}\\ \mathbf{w}_k^{vd}\\ \mathbf{w}_k^{\alpha}\end{bmatrix}, \tag{5.57}$$

when the previous contact persists, and

$$\begin{bmatrix}\delta\boldsymbol{\phi}_{ik}^+\\ \delta\mathbf{v}_{ik}^+\\ \delta\mathbf{p}_{ik}^+\\ \delta\mathbf{d}_{ik}^+\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0}\\ -\Delta\tilde{\mathbf{R}}_{ik}\left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\right)^\wedge & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}}_{\mathbf{F}_k}\begin{bmatrix}\delta\boldsymbol{\phi}_{ik}^-\\ \delta\mathbf{v}_{ik}^-\\ \delta\mathbf{p}_{ik}^-\\ \delta\mathbf{d}_{ik}^-\end{bmatrix} + \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \Delta\tilde{\mathbf{R}}_{ik}\,{}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\end{bmatrix}}_{\mathbf{G}_k}\begin{bmatrix}\mathbf{w}_k^{gd}\\ \mathbf{w}_k^{ad}\\ \mathbf{w}_k^{vd}\\ \mathbf{w}_k^{\alpha}\end{bmatrix} \quad k\in\mathcal{S}, \tag{5.58}$$

when a contact switch is detected. This hybrid linear system can be used to iteratively compute the covariance, starting with $\boldsymbol{\Sigma}_{\mathcal{IC}_{ii}} = \mathbf{0}_{12\times 12}$,

$$\begin{cases}\boldsymbol{\Sigma}_{\mathcal{IC}_{ik+1}} = \mathbf{A}_k\boldsymbol{\Sigma}_{\mathcal{IC}_{ik}}\mathbf{A}_k^{\mathsf{T}} + \mathbf{B}_k\boldsymbol{\Sigma}_k^{\mathbf{w}}\mathbf{B}_k^{\mathsf{T}}\\ \boldsymbol{\Sigma}_{\mathcal{IC}_{ik^+}} = \mathbf{F}_k\boldsymbol{\Sigma}_{\mathcal{IC}_{ik^-}}\mathbf{F}_k^{\mathsf{T}} + \mathbf{G}_k\boldsymbol{\Sigma}_k^{\mathbf{w}}\mathbf{G}_k^{\mathsf{T}} \qquad k\in\mathcal{S},\end{cases} \tag{5.59}$$

where $\boldsymbol{\Sigma}_k^{\mathbf{w}}$ denotes the covariance of the sensor noise $\mathbf{w}_k = \mathrm{vec}(\mathbf{w}_k^{gd}, \mathbf{w}_k^{ad}, \mathbf{w}_k^{vd}, \mathbf{w}_k^{\alpha})$.

5.5.4 First-order Bias Update

Similar to the preintegrated measurements developed by Forster et al. [85], the hybrid preintegrated point contact position measurement, $\Delta\tilde{\mathbf{d}}_{ij}$, depends on the IMU bias estiamte at the time of preintegration. In this case, $\Delta\tilde{\mathbf{d}}_{ij}$ only depends on the gyroscope bias. When this bias estimate changes, instead of re-integrating the measurement (time and memory intensive), we simply apply a first-order correction;

$$\Delta\tilde{\mathbf{d}}_{ij}(\bar{\mathbf{b}}_i^g + \delta\mathbf{b}^g) \approx \tilde{\mathbf{d}}_{ij}(\bar{\mathbf{b}}_i^g) + \frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g. \tag{5.60}$$

The Jacobian of the preintegrated measurement with respect to the bias can be computed iteratively. Looking at (5.55), we can see that $\Delta\tilde{\mathbf{d}}_{ij}$ only depends on the bias estimate through the preintegrated rotation term, $\Delta\tilde{\mathbf{R}}_{ij}$, which arises from switching contact. Therefore, the Jacobian will only change on contact switch. The iterative form of the Jacobian can be derived as

$$\begin{aligned}
\Delta\tilde{\mathbf{d}}_{ik}^+ + \frac{\partial\Delta\tilde{\mathbf{d}}_{ik}^+}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g &= \Delta\tilde{\mathbf{d}}_{ik}^- + \frac{\partial\Delta\tilde{\mathbf{d}}_{ik}^-}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g + \Delta\tilde{\mathbf{R}}_{ik}\,\mathrm{Exp}\left(\frac{\partial\Delta\mathbf{R}_{ik}}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g\right){}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k) \\
&\approx \Delta\tilde{\mathbf{d}}_{ik}^- + \frac{\partial\Delta\tilde{\mathbf{d}}_{ik}^-}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g + \Delta\tilde{\mathbf{R}}_{ik}\left(\mathbf{I} + \left(\frac{\partial\Delta\mathbf{R}_{ik}}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g\right)_\times\right){}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k) \\
&= \Delta\tilde{\mathbf{d}}_{ik}^- + \Delta\tilde{\mathbf{R}}_{ik}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k) + \left(\frac{\partial\Delta\tilde{\mathbf{d}}_{ik}^-}{\partial\delta\mathbf{b}^g} - \Delta\tilde{\mathbf{R}}_{ik}\left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\right)_\times\frac{\partial\Delta\mathbf{R}_{ik}}{\partial\delta\mathbf{b}^g}\right)\delta\mathbf{b}^g \\
\implies \frac{\partial\Delta\tilde{\mathbf{d}}_{ik}^+}{\partial\delta\mathbf{b}^g} &= \frac{\partial\Delta\tilde{\mathbf{d}}_{ik}^-}{\partial\delta\mathbf{b}^g} - \Delta\tilde{\mathbf{R}}_{ik}\left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_k)\right)_\times\frac{\partial\Delta\mathbf{R}_{ik}}{\partial\delta\mathbf{b}^g}.
\end{aligned} \tag{5.61}$$

5.5.5 Combined Point Contact and IMU Factor

Since the noise, $\delta\mathbf{d}_{ij}$, is coupled to the base orientation noise, $\delta\boldsymbol{\phi}_{ij}$, the preintegrated hybrid point contact factor has to be coupled with the preintegrated IMU factor described in [85, 84]. This was not necessary for the rigid contact case described in Section 5.3 due to the static contact orientation assumption. The combined, preintegrated hybrid point contact and IMU residual includes the three IMU residuals defined in [85] and the new hybrid point contact

residual (5.53). Written explicitly, the full residual becomes:

$$\mathbf{r}_{\mathcal{IC}_{ij}} \triangleq \begin{bmatrix} \mathbf{r}_{\Delta\mathbf{R}_{ij}} \\ \mathbf{r}_{\Delta\mathbf{v}_{ij}} \\ \mathbf{r}_{\Delta\mathbf{p}_{ij}} \\ \mathbf{r}_{\Delta\mathbf{d}_{ij}} \end{bmatrix} = \begin{bmatrix} \operatorname{Log}\left(\left(\Delta\tilde{\mathbf{R}}_{ij}\operatorname{Exp}\left(\frac{\partial\tilde{\mathbf{R}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g\right)\right)^{\mathsf{T}}\mathbf{R}_i^{\mathsf{T}}\mathbf{R}_j\right) \\ \mathbf{R}_i^{\mathsf{T}}(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) - \left[\Delta\tilde{\mathbf{v}}_{ij} + \frac{\partial\tilde{\mathbf{v}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g + \frac{\partial\tilde{\mathbf{v}}_{ij}}{\partial\mathbf{b}^a}\delta\mathbf{b}^a\right] \\ \mathbf{R}_i^{\mathsf{T}}(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}) - \left[\Delta\tilde{\mathbf{p}}_{ij} + \frac{\partial\tilde{\mathbf{p}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g + \frac{\partial\tilde{\mathbf{p}}_{ij}}{\partial\mathbf{b}^a}\delta\mathbf{b}^a\right] \\ \mathbf{R}_i^{\mathsf{T}}(\mathbf{d}_j - \mathbf{d}_i) - \left[\Delta\tilde{\mathbf{d}}_{ij} + \frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g\right] \end{bmatrix} \tag{5.62}$$

This combined residual and its covariance, $\mathbf{\Sigma}_{\mathcal{IC}_{ij}}$ (computed using (5.59)), can be used to insert the *hybrid preintegrated contact factors* into the factor graph optimization problem (5.4).

Remark 15. If no contact switch occurs between two consecutive keyframes, the contact position noise does not get correlated to the base orientation noise. This can be seen by recognizing that the orientation noise, $\delta\boldsymbol{\phi}_{ij}$, only appears in the transition map of (5.56). Also, if both measurements are Gaussian and uncorrelated, they are also independent. Therefore, if no contact switching occurs, the preintegrated IMU measurement can be decoupled from the point-contact position measurement. This simplifies the implementation of the factors and allows for pre-existing or alternative formulations of the IMU factor to be used.

5.5.6 Forward Kinematic Position Factor

The forward kinematic factor derived in Section 5.2 relates the base frame pose to a contact frame pose. However, if a the point contact factor is used (instead of the rigid contact factor), the contact frame orientation is not being constrained. Therefore, including the contact frame pose in the state is unnecessary since the Maximum-A-Posteriori (MAP) estimate would simply be the value obtained from forward kinematics. In this section, we derive a forward kinematic position factor that only constrains the contact frame position, allowing us to remove the orientation state from the optimization problem.

The position of the contact frame at any given time can be expressed as

$${}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}}(t) = {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WB}}(t) + \mathbf{R}_{\mathrm{WB}}(t)\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\boldsymbol{\alpha}(t)), \tag{5.63}$$

where the position of the contact frame relative to the base frame, ${}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\boldsymbol{\alpha}(t))$ is measured

using forward kinematics. Re-writing this expression using the new state definition (5.39) at time t_i and encoder measurements (5.7) gives

$$\begin{aligned}\mathbf{d}_i &= \mathbf{p}_i + \mathbf{R}_i \, {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i - \mathbf{w}_i^{\boldsymbol{\alpha}}) \\ &\approx \mathbf{p}_i + \mathbf{R}_i \, {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i) - \mathbf{R}_i \, {}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)\mathbf{w}_i^{\boldsymbol{\alpha}},\end{aligned} \tag{5.64}$$

where the kinematics Jacobian appears when taking a first-order approximation of the position kinematics to separate out the encoder noise (5.49).

Now that the noise is separated out, we can construct the measurement residual and its covariance. The residual is

$$\mathbf{r}_{\mathcal{F}_i} = \mathbf{R}_i^{\mathsf{T}}(\mathbf{d}_i - \mathbf{p}_i) - {}_{\mathrm{B}}\mathbf{p}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i). \tag{5.65}$$

The covariance is computed through the linear transformation

$$\boldsymbol{\Sigma}_{\mathcal{F}_i} = {}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i) \, \boldsymbol{\Sigma}_i^{\boldsymbol{\alpha}} \, {}_{\mathrm{B}}\mathbf{J}^{\dot{p}\mathsf{T}}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i), \tag{5.66}$$

where $\boldsymbol{\Sigma}_i^{\boldsymbol{\alpha}}$ denotes the encoder covariance matrix at time t_i. This residual and covariance can now be used to insert the *forward kinematic factors* into the factor graph optimization problem (5.4).

5.6 Analytical Preintegration and Invariant Smoothing

In Section 5.5, a hybrid point contact factor was developed that relates the contact frame position between two keyframes. Due to the modeling of contact as a hybrid dynamical system, an arbitrary number of contact frames switches can be handled. This factor was derived as an extension of the preintegrated IMU factors developed by Forster et al. [85]. As explained by Eckenhoff et al. [74], discretization was done assuming piecewise constant acceleration measurements in the world frame. In contrast, Eckenhoff et al. [74] provided a method for closed-form preintegration of the acceleration measurements assuming they are piecewise constant in the body (or local) frame. This method of integration is a closer approximation to the true body acceleration that the accelerometer is designed to measure, which leads to improved state estimation.

However, both Forster et al. [85] and Eckenhoff et al. [74] used a decoupled state, similar to a quaternion-based extended Kalman filter (QEKF), where the orientation, position, and

velocity all lie in separate spaces. When decoupled, the linearized error dynamics depend on the state estimate, which can degrade the accuracy of uncertainty propagation. In addition, using this formulation, the position and velocity covariances are represented by ellipses in $\mathbb{R}^3$. In some cases, this representation of covariance can poorly reflect the underlying distributions [22] which are often curved.

In Chapter 4, based on the work of Barrau and Bonnabel [24], we developed an invariant extended Kalman filter (InEKF) that fuses inertial-contact dynamics with forward kinematic corrections to estimate the base pose and velocity along with all current contact positions. This filter was able to outperform the standard QEKF by leveraging Lie group theory to take advantage of inherent symmetries present in the error dynamics. In particular, the derived deterministic log-linear error dynamics are autonomous and exactly describe the evolution of the state error defined on the Lie group itself. This means that without sensor noise, the error covariance can be propagated without approximation. Although this does not hold in practical cases with sensor noise and bias, we showed that this formulation still outperforms the standard QEKF by exploiting these symmetries and through representation of the covariance in the Lie algebra as opposed to a Euclidean vector space. When pushed through the group's exponential map, these covariance ellipses become "curved" on the group, which better reflects the underlying distribution.

In this section, we derive a new *invariant hybrid inertial-contact factor* that leverages the results from Chapter 4 to provide an alternate factor that can be used in place of the hybrid point contact factor derived in Section 5.5. In addition to the analytical integration and improved accuracy of covariance propagation, the equations for this new factor have nice parallels to the InEKF equations of Chapter 4. Furthermore, under certain assumptions, the Jacobian of this factor's residual becomes independent of the current state estimate. This is the main idea behind the *invariant smoothing* framework developed by Chauchat et al. [48].

5.6.1 Hybrid Contact-Inertial Dynamics

Similar to the contact-aided InEKF (Chapter 4) and the hybrid preintegrated point contact factor (Section 5.5), we are interested in estimating the base orientation $\mathbf{R}_{\mathrm{WB}}(t)$, velocity ${}_{\mathrm{W}}\mathbf{v}_{\mathrm{B}}(t)$, and position ${}_{\mathrm{W}}\mathbf{p}_{\mathrm{WB}}(t)$, as well as a contact position ${}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}}(t)$. All of these states are measured in the world frame. We can combine these states into a single matrix Lie group;

$$\mathbf{X}_t = \begin{bmatrix} \mathbf{R}_{\mathrm{WB}}(t) & {}_{\mathrm{W}}\mathbf{v}_{\mathrm{B}}(t) & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WB}}(t) & {}_{\mathrm{W}}\mathbf{p}_{\mathrm{WC}}(t) \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{R}_t & \mathbf{v}_t & \mathbf{p}_t & \mathbf{d}_t \\ \mathbf{0}_{1\times 3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times 3} & 0 & 0 & 1 \end{bmatrix} \in \mathrm{SE}_3(3)$$

In addition, we are also interested in estimating the inertial measurement unit (IMU) bias $\mathbf{b}_t \triangleq \text{vec}(\mathbf{b}_t^g, \mathbf{b}_t^a) \in \mathbb{R}^6$. Together, the full state that we are estimating is a combination of this group and the bias vector, which can be written as a tuple, $(\mathbf{X}_t,\ \mathbf{b}_t)$.

Let $\tilde{\boldsymbol{\omega}}_t$ and $\tilde{\mathbf{a}}_t$ be the noisy gyroscope and accelerometer measurements. When contact is measured, the stance foot velocity is assumed zero with additive Gaussian noise to accommodate foot slip, and the continuous group dynamics can be written as

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{X}_t &= \begin{bmatrix} \mathbf{R}_t\,(\tilde{\boldsymbol{\omega}}_t - \mathbf{b}_t^g)_\times & \mathbf{R}_t(\tilde{\mathbf{a}}_t - \mathbf{b}_t^a) + \mathbf{g} & \mathbf{v}_t & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} \mathbf{R}_t & \mathbf{v}_t & \mathbf{p}_t & \mathbf{d}_t \\ \mathbf{0}_{1\times3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} (\mathbf{w}_t^g)_\times & \mathbf{w}_t^a & \mathbf{0}_{3\times1} & \mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \end{bmatrix} \\ &\triangleq f_{u_t}(\mathbf{X}_t) - \mathbf{X}_t(\mathbf{w}_t^f)^\wedge,\end{aligned} \tag{5.67}$$

where $\mathbf{w}_t^f \triangleq \text{vec}(\mathbf{w}_t^g,\ \mathbf{w}_t^a,\ \mathbf{0}_{3\times1}, \mathbf{R}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_t)\mathbf{w}_t^v)$ is the vector of inertial-contact sensor noise. The continuous bias dynamics are modeled using the standard Brownian motion model,

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{b}_t = \mathbf{w}_t^b, \tag{5.68}$$

where $\mathbf{w}_t^b \triangleq \text{vec}(\mathbf{w}_t^{bg},\ \mathbf{w}_t^{ba})$ is a vector of gyroscope and accelerometer bias noise. Refer to Chapter 4 for more information as these dynamics equations are identical to the world-centric InEKF dynamics.

However, unlike the InEKF (where integration is performed over a single time step), we are interested in preintegrating the dynamics between two keyframes. In general, the frame in contact with the ground may change an arbitrary number of times between any two keyframes. When a contact switch occurs, we can map from the old contact position, $\mathbf{d}_t^-$, to the new contact position, $\mathbf{d}_t^+$, using forward kinematics;

$$\begin{aligned}\mathbf{d}_t^+ &= \mathbf{d}_t^- + \mathbf{R}_t\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_t - \mathbf{w}_k^{\boldsymbol{\alpha}}) \\ &\approx \mathbf{d}_t^- + \mathbf{R}_t\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C+}}(\tilde{\boldsymbol{\alpha}}_t) - \mathbf{R}_t\,{}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_t)\,\mathbf{w}_t^{\boldsymbol{\alpha}}.\end{aligned} \tag{5.69}$$

In the above expression, ${}_{\mathrm{B}}\mathbf{p}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_t)$ denotes the position of the new contact frame relative to the old contact frame as measured in the body frame. The analytical Jacobian of this kinematics function, $\mathbf{J}^{\dot{p}}_{\mathrm{C^-C^+}}(\tilde{\boldsymbol{\alpha}}_t)$, is used to make a first-order approximation allowing separation of the noise from the kinematics (see Section 3.5). This discrete update can be written

in matrix form as

$$\begin{aligned}\mathbf{X}_t^+ &= \mathbf{X}_t^- \begin{bmatrix} \mathbf{I}_{3\times3} & 0 & 0 & {}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C+}}(\tilde{\boldsymbol{\alpha}}_t) \\ \mathbf{0}_{1\times3} & 1 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 1 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 1 \end{bmatrix} - \mathbf{X}_t^- \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & {}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_t)\,\mathbf{w}_t^{\boldsymbol{\alpha}} \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \\ \mathbf{0}_{1\times3} & 0 & 0 & 0 \end{bmatrix} \\ &\triangleq \Delta_{u_t}(\mathbf{X}_t^-) - \mathbf{X}_t^-(\mathbf{w}_t^{\Delta})^{\wedge},\end{aligned} \tag{5.70}$$

with $\mathbf{w}_t^{\Delta} \triangleq \mathrm{vec}(\mathbf{0}_{3\times1},\ \mathbf{0}_{3\times1},\ \mathbf{0}_{3\times1},\ {}_{\mathrm{B}}\mathbf{J}^{\dot{p}}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_t)\,\mathbf{w}_t^{\boldsymbol{\alpha}})$ being the noise vector for the discrete update. The bias states remain unchanged during the discrete map. Together the continuous group dynamics (5.67), bias dynamics (5.68), and the discrete dynamics (5.70) form a hybrid system;

$$\mathcal{H}: \begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{X}_t,\ \mathbf{b}_t\right) = \left(f_{u_t}(\mathbf{X}_t) - \mathbf{X}_t(\mathbf{w}_t^f)^{\wedge},\ \mathbf{w}_t^b\right) & t^- \notin \mathcal{S} \\ \left(\mathbf{X}_t^+,\ \mathbf{b}_t^+\right) = \left(\Delta_{u_t}(\mathbf{X}_t^-) - \mathbf{X}_t^-(\mathbf{w}_t^{\Delta})^{\wedge},\ \mathbf{b}_t^-\right) & t^- \in \mathcal{S}, \end{cases} \tag{5.71}$$

where the switching surface $\mathcal{S}$ is defined by all times where contact switch occurs.

5.6.2 Analytical Preintegration of Mean

Given the state at time t_i, we can propagate the mean to time t_j by integrating this deterministic part of the hybrid dynamics (5.71):

$$\begin{aligned}\mathbf{R}_{t_j} &= \int_{t_i}^{t_j} \mathbf{R}_t\,(\tilde{\boldsymbol{\omega}}_t - \mathbf{b}_t^g)_{\times}\,dt \\ \mathbf{v}_{t_j} &= \mathbf{v}_{t_i} + \int_{t_i}^{t_j} \mathbf{R}_t(\tilde{\mathbf{a}}_t - \mathbf{b}_t^a) + \mathbf{g}\,dt \\ \mathbf{p}_{t_j} &= \mathbf{p}_{t_i} + \int_{t_i}^{t_j} \mathbf{v}_t\,dt \\ \mathbf{d}_{t_j} &= \mathbf{d}_{t_i} + \sum_{t_k\in\mathcal{S}} \mathbf{R}_{t_k}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C^+}}(\tilde{\boldsymbol{\alpha}}_{t_k}) \\ \mathbf{b}_j &= \mathbf{b}_i.\end{aligned} \tag{5.72}$$

In practice, sensor measurements come in at discrete times. Assuming a zero-order hold on the measurements between consecutive sensor time steps, we can analytically integrate the

above expressions to yield

$$\begin{aligned}
\mathbf{R}_j &= \mathbf{R}_i \prod_{k=i}^{j-1} \mathrm{Exp}((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g)\Delta t) \\
\mathbf{v}_j &= \mathbf{v}_i + \sum_{k=i}^{j-1} (\mathbf{R}_k \boldsymbol{\Gamma}_1((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g)\Delta t)(\tilde{\mathbf{a}}_k - \mathbf{b}_i^a)\Delta t + \mathbf{g}\Delta t) \\
\mathbf{p}_j &= \mathbf{p}_i + \sum_{k=i}^{j-1} \left(\mathbf{v}_k \Delta t + \mathbf{R}_k \boldsymbol{\Gamma}_2((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i^g)\Delta t)(\tilde{\mathbf{a}}_k - \mathbf{b}_i^a)\Delta t^2 + \frac{1}{2}\mathbf{g}\Delta t^2 \right) \\
\mathbf{d}_j &= \mathbf{d}_i + \sum_{k \in \mathcal{S}} \mathbf{R}_{k\,\mathrm{B}} \mathbf{p}_{\mathrm{C}^-\mathrm{C}^+}(\tilde{\boldsymbol{\alpha}}_k) \\
\mathbf{b}_j &= \mathbf{b}_i,
\end{aligned} \tag{5.73}$$

where $t_{(\cdot)}$ was replaced with the subscript to simplify notation. The matrix $\boldsymbol{\Gamma}_m(\cdot)$ denotes the m^{th} integration of the exponential map (4.85). For more details see Section 4.10. If we were to directly use the equations above during optimization of the factor graph, every time $\mathbf{X}_i$ changes, the full integration would have to be performed again. To prevent this, we construct a set of preintegrated measurements that are independent of the group states at times t_i and t_j. This leads to a set of analytical preintegrated measurements defined as

$$\begin{aligned}
\Delta\tilde{\mathbf{R}}_{ij} &\triangleq \mathbf{R}_i^\mathsf{T} \mathbf{R}_j = \prod_{k=i}^{j-1} \mathrm{Exp}(\bar{\boldsymbol{\omega}}_k \Delta t) \\
\Delta\tilde{\mathbf{v}}_{ij} &\triangleq \mathbf{R}_i^\mathsf{T} \left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij} \right) = \sum_{k=i}^{j-1} \Delta\tilde{\mathbf{R}}_{ik} \boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t) \bar{\mathbf{a}}_k \Delta t \\
\Delta\tilde{\mathbf{p}}_{ij} &\triangleq \mathbf{R}_i^\mathsf{T} \left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_t \Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2 \right) = \sum_{k=i}^{j-1} \Delta\tilde{\mathbf{R}}_{ik} \left(\boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k \Delta t) + \boldsymbol{\Gamma}_2(\bar{\boldsymbol{\omega}}_k \Delta t) \right) \bar{\mathbf{a}}_k \Delta t^2 \\
\Delta\tilde{\mathbf{d}}_{ij} &\triangleq \mathbf{R}_i^\mathsf{T} \left(\mathbf{d}_j - \mathbf{d}_i \right) = \sum_{k \in \mathcal{S}} \Delta\tilde{\mathbf{R}}_{ik\,\mathrm{B}} \mathbf{p}_{\mathrm{C}^-\mathrm{C}^+}(\tilde{\boldsymbol{\alpha}}_k),
\end{aligned} \tag{5.74}$$

where $\bar{\boldsymbol{\omega}}_k \triangleq \tilde{\boldsymbol{\omega}}_k - \bar{\mathbf{b}}_i^g$ and $\bar{\boldsymbol{\omega}}_k \triangleq \tilde{\mathbf{a}}_k - \bar{\mathbf{b}}_i^a$ denote the IMU measurements corrected by the current bias estimate at the time of integration.

These preintegrated measurements can also be written in iterative form to allow integra-

tion as the data streams in;

$$\begin{aligned}
\Delta\tilde{\mathbf{R}}_{i,k+1} &= \Delta\tilde{\mathbf{R}}_{i,k}\,\mathrm{Exp}(\bar{\boldsymbol{\omega}}_k\Delta t)\\
\Delta\tilde{\mathbf{v}}_{i,k+1} &= \Delta\tilde{\mathbf{v}}_{i,k} + \Delta\tilde{\mathbf{R}}_{ik}\boldsymbol{\Gamma}_1(\bar{\boldsymbol{\omega}}_k\Delta t)\bar{\mathbf{a}}_k\Delta t\\
\Delta\tilde{\mathbf{p}}_{i,k+1} &= \Delta\tilde{\mathbf{p}}_{i,k} + \Delta\tilde{\mathbf{v}}_{i,k+1}\Delta t + \Delta\tilde{\mathbf{R}}_{ik}\boldsymbol{\Gamma}_2(\bar{\boldsymbol{\omega}}_k\Delta t)\bar{\mathbf{a}}_k\Delta t^2\\
\Delta\tilde{\mathbf{d}}_{i,k+1} &= \begin{cases}\Delta\tilde{\mathbf{d}}_{i,k}\\ \Delta\tilde{\mathbf{d}}_{i,k} + \Delta\tilde{\mathbf{R}}_{ik}\,{}_{\mathrm{B}}\mathbf{p}_{\mathrm{C}^-\mathrm{C}^+}(\tilde{\boldsymbol{\alpha}}_k) & k\in\mathcal{S}\end{cases}
\end{aligned} \tag{5.75}$$

Unfortunately, these preintegrated measurements still depend on the bias state at time t_i. However, instead of re-integrating the measurements, a first-order correction will be done to account for changes in the bias estimate. The analytical preintegration of the IMU states were first derived by Eckenhoff et al. [75]. Here, the preintegration was extended to include the contact position state. We can now rewrite the state $\mathbf{X}_j$ as a function of the state $\mathbf{X}_i$ and this set of preintegrated measurements;

$$\begin{aligned}
\mathbf{X}_j &= \begin{bmatrix}\mathbf{R}_i\Delta\tilde{\mathbf{R}}_{ij} & \mathbf{v}_i + \mathbf{g}\Delta t_{ij} + \mathbf{R}_i\Delta\tilde{\mathbf{v}}_{ij} & \mathbf{p}_i + \mathbf{v}_i\Delta t_{ij} + \frac{1}{2}\mathbf{g}\Delta t_{ij}^2 + \mathbf{R}_i\Delta\tilde{\mathbf{p}}_{ij} & \mathbf{d}_i + \mathbf{R}_i\Delta\tilde{\mathbf{d}}_{ij}\\ \mathbf{0}_{1\times3} & 1 & 0 & 0\\ \mathbf{0}_{1\times3} & 0 & 1 & 0\\ \mathbf{0}_{1\times3} & 0 & 0 & 1\end{bmatrix}\\
&\triangleq f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_t).
\end{aligned} \tag{5.76}$$

Preintegration of the sensor measurements turned the hybrid continuous state dynamics (5.71) into simple discrete dynamics equation.

5.6.3 Covariance Propagation

To propagate the covariance we can utilize the linearized error dynamics. The error between two states can be defined a number of different ways. However, from Chapter 4, we know that the continuous deterministic dynamics $f_{u_t}(\cdot)$ satisfies the group affine property (4.3). This means that according to Theorems 1 and 2 by [24], both the right- and left-invariant errors lead to autonomous, log-linear error dynamics. Although with contact switching the deterministic dynamics follow a hybrid system (5.71), this property is preserved. So, although the error variables can be defined a number of ways, it is natural and beneficial to choose either the right- or left-invariant error (4.2). If the right-invariant error is used, the noise will appear on the left of the propagated state, while if the left-invariant error is used,

the noise will appear on the right;

$$\begin{aligned} \mathbf{X}_j &= \operatorname{Exp}(\boldsymbol{\xi}_{ij})\, f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_t) \qquad \text{Right-invariant error} \\ \mathbf{X}_j &= f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_t) \operatorname{Exp}(\boldsymbol{\xi}_{ij}) \qquad \text{Left-invariant error.} \end{aligned} \tag{5.77}$$

Intuitively, the left-invariant error represents an error measured in the body frame, while the right-invariant error is measured in the world (spatial) frame. In the factor graph framework, this error is assumed to be a zero-mean Gaussian. The covariance of this error (and ultimately the residual) can be computed by assuming zero error at time t_i and using the linearized error dynamics to propagate the covariance until time t_j. In the following sections, we show how to compute the residual covariance for the left-invariant error definition.

5.6.3.1 Left-Invariant Error Dynamics

The left-invariant error between two trajectories is defined as $\boldsymbol{\eta}_t^r \triangleq \mathbf{X}_t^{-1}\bar{\mathbf{X}}_t \triangleq \operatorname{Exp}(\boldsymbol{\xi}_t)$, where $\boldsymbol{\xi}_t$ denotes the Lie algebra representation of the error. Since the continuous part of dynamics is identical to the derived world-centric left-invariant dynamics from Chapter 4, we only need to derive the log-linear error update when a contact switch occurs. Deriving this update using the definition of the left-invariant contact position error, $\boldsymbol{\eta}_t^d = \mathbf{R}_t^\mathsf{T}(\bar{\mathbf{d}}_t - \mathbf{d}_t)$, gives

$$\begin{aligned} \boldsymbol{\eta}_t^{d+} &= \mathbf{R}_t^\mathsf{T}(\bar{\mathbf{d}}_t^+ - \mathbf{d}_t^+) \\ &= \mathbf{R}_t^\mathsf{T}\left(\bar{\mathbf{d}}_k^- + \bar{\mathbf{R}}_t\, {}_\mathrm{B}\mathbf{p}_{\text{C-C+}}(\tilde{\boldsymbol{\alpha}}_t) - \left(\mathbf{d}_k^- + \mathbf{R}_t\, {}_\mathrm{B}\mathbf{p}_{\text{C-C+}}(\tilde{\boldsymbol{\alpha}}_t) - \mathbf{R}_t\, {}_\mathrm{B}\mathbf{J}^{\dot{p}}_{\text{C-C+}}\, \mathbf{w}_t^{\boldsymbol{\alpha}}\right)\right) \\ &= \boldsymbol{\eta}_t^{d-} + \left(\boldsymbol{\eta}_t^R - \mathbf{I}\right) {}_\mathrm{B}\mathbf{p}_{\text{C-C+}}(\tilde{\boldsymbol{\alpha}}_t) + {}_\mathrm{B}\mathbf{J}^{\dot{p}}_{\text{C-C+}}\, \mathbf{w}_t^{\boldsymbol{\alpha}} \\ \implies \boldsymbol{\xi}_t^{d+} &\approx \boldsymbol{\xi}_t^{d-} - \left({}_\mathrm{B}\mathbf{p}_{\text{C-C+}}(\tilde{\boldsymbol{\alpha}}_t)\right)_\times \boldsymbol{\xi}_t^R + {}_\mathrm{B}\mathbf{J}^{\dot{p}}_{\text{C-C+}}\, \mathbf{w}_t^{\boldsymbol{\alpha}}. \end{aligned} \tag{5.78}$$

Using the previously derived continuous left-invariant error dynamics from Chapter 4, we can express the hybrid error dynamics as

$$d\mathcal{H} : \begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \boldsymbol{\xi}_t \\ \boldsymbol{\zeta}_t \end{bmatrix} = \mathbf{A}_t^l \begin{bmatrix} \boldsymbol{\xi}_t \\ \boldsymbol{\zeta}_t \end{bmatrix} + \begin{bmatrix} \mathbf{w}_t^f \\ \mathbf{w}_t^b \end{bmatrix} & t^- \notin \mathcal{S} \\ \begin{bmatrix} \boldsymbol{\xi}_t^+ \\ \boldsymbol{\zeta}_t^+ \end{bmatrix} = \mathbf{D}_t \begin{bmatrix} \boldsymbol{\xi}_t^- \\ \boldsymbol{\zeta}_t^- \end{bmatrix} + \begin{bmatrix} \mathbf{w}_t^\Delta \\ \mathbf{w}_t^b \end{bmatrix} & t^- \in \mathcal{S}, \end{cases} \tag{5.79}$$

where the matrices $\mathbf{A}_t^l$ and $\mathbf{D}_t$ are given by

$$\begin{aligned}
\mathbf{A}_t^l &= \begin{bmatrix}
-(\bar{\boldsymbol{\omega}}_t)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\
-(\bar{\mathbf{a}}_t)_\times & -(\bar{\boldsymbol{\omega}}_t)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} \\
\mathbf{0} & \mathbf{I} & -(\bar{\boldsymbol{\omega}}_t)_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & -(\bar{\boldsymbol{\omega}}_t)_\times & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{bmatrix} \\
\mathbf{D}_t &= \begin{bmatrix}
\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
-\left({}_{\mathrm{B}}\mathbf{p}_{\mathrm{C\text{-}C+}}(\tilde{\boldsymbol{\alpha}})\right)_\times & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}
\end{bmatrix}.
\end{aligned} \tag{5.80}$$

It is important to note that in the case without IMU bias and sensor noise, the solution to (5.79) exactly describes the evolution of the left-invariant error. In other words, the error dynamics are exactly described by a hybrid log-linear system. The proof is sketched as follows. The continuous dynamics, f_{u_t}, were shown to satisfy the "continuous group-affine" property [24] in Chapter 4. The discrete dynamics, Δ_{u_t} can also be shown to satisfy the discrete "discrete group affine" property [25]. Therefore, assuming the initial error is known Theorem 2 and its discrete time analog can be used to show that the left-invariant error can be exactly recovered from the corresponding hybrid linear system.

5.6.3.2 Iterative Covariance Propagation

Now that we have a hybrid linear system describing the error dynamics, the error at time t_j can be computed iteratively by integrating the (5.79) from t_i to t_j, starting from zero error. Therefore, the covariance can be computed iteratively, starting from $\boldsymbol{\Sigma}_{\mathcal{IC}_{ii}} = \mathbf{0}_{18\times 18}$, using

$$\begin{cases}
\boldsymbol{\Sigma}_{\mathcal{IC}_{ik+1}} = \boldsymbol{\Phi}_{k+1,k}^l \, \boldsymbol{\Sigma}_{\mathcal{IC}_{ik}} \, \boldsymbol{\Phi}_{k+1,k}^{l\mathsf{T}} + \bar{\mathbf{Q}}_k^{fd} \\
\boldsymbol{\Sigma}_{\mathcal{IC}_{ik}}^+ = \mathbf{D}_k \, \boldsymbol{\Sigma}_{\mathcal{IC}_{ik}}^- \, \mathbf{D}_k + \mathbf{Q}_k^\Delta \qquad\qquad k \in \mathcal{S},
\end{cases} \tag{5.81}$$

where $\boldsymbol{\Phi}_{k,k+1}^l \triangleq \boldsymbol{\Phi}^l(t_{k+1}, t_k)$ denotes the state transition matrix for the continuous left-invariant error dynamics equation and $\bar{\mathbf{Q}}_k^{fd}$ denotes the associated discrete noise covariance

matrix. The state transition matrix, $\mathbf{\Phi}^l(t_{k+1}, t_k)$, satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{\Phi}^l(t, t_k) = \mathbf{A}_t^l\, \mathbf{\Phi}^l(t, t_k) \quad \text{with} \quad \mathbf{\Phi}^l(t_k, t_k) = \mathbf{I}, \tag{5.82}$$

which has closed form solution previously derived in Section 4.10. The discrete noise covariance matrix can be computed by solving

$$\begin{aligned} \bar{\mathbf{Q}}_k^{fd} &= \int_{t_k}^{t_{k+1}} \mathbf{\Phi}^l(t, t_k)\, \mathbf{Q}_t^f\, \mathbf{\Phi}^l(t, t_k)^\mathsf{T}\, dt \\ &\approx \mathbf{\Phi}^l_{t_{k+1},t_k}\, \mathbf{Q}_k^f\, \mathbf{\Phi}^{l\mathsf{T}}_{t_{k+1},t_k} \Delta t, \end{aligned} \tag{5.83}$$

where the latter approximation can be done for efficiency.

5.6.4 Bias Correction

When preintegrating the measurements (5.74) the IMU bias estimate is assumed to be fixed at $\mathbf{b}_i$. To prevent re-integration of the measurements, when the bias estimate changes, we apply a first-order correction. This is similar to the process done by Forster et al. [85] and Eckenhoff et al. [75]. This first-order correction can be expressed as

$$f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i + \delta\mathbf{b}) \approx f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i)\, \mathrm{Exp}\left(\frac{\partial f_{\Delta_{ij}}}{\partial \delta\mathbf{b}}\delta\mathbf{b}\right), \tag{5.84}$$

where $\delta\mathbf{b}$ is a small bias correction and $\frac{\partial f_{\Delta_{ij}}}{\partial \delta\mathbf{b}}$ is a (right) Jacobian that relates changes in bias to changes on the group. This bias Jacobian can be computed iteratively using the previously defined state transition matrix, $\mathbf{\Phi}^l_{k+1,k}$, that has an analytical solution (4.95) derived in Chapter 4. Starting with $\frac{\partial f_{\Delta_{ii}}}{\partial \delta\mathbf{b}} = \mathbf{0}_{12\times 6}$, this iterative update can be computed by

$$\begin{bmatrix} \frac{\partial f_{\Delta_{ik+1}}}{\partial \delta\mathbf{b}} \\ \mathbf{I}_6 \end{bmatrix} = \mathbf{\Phi}^l_{k+1,k} \begin{bmatrix} \frac{\partial f_{\Delta_{ik}}}{\partial \delta\mathbf{b}} \\ \mathbf{I}_6 \end{bmatrix}, \tag{5.85}$$

where the identity block comes from Jacobian of the discrete bias dynamics.

5.6.5 Invariant Hybrid Inertial-Contact Factor

Combining the discrete preintegrated dynamics (5.76), the first-order bias update 5.84, and the left-invariant error definition (5.77) allows us to fully express the our prediction of the

state at time t_j as

$$\begin{aligned} \mathbf{X}_j &= f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i) \operatorname{Exp}\left(\frac{\partial f_{\Delta_{ij}}}{\partial \delta \mathbf{b}} \delta \mathbf{b}\right) \operatorname{Exp}(\boldsymbol{\xi}_{ij}) \\ \mathbf{b}_j &= \mathbf{b}_i + \boldsymbol{\zeta}_{ij}, \end{aligned} \tag{5.86}$$

where the error vector,

$$\operatorname{vec}(\boldsymbol{\xi}_{ij},\, \boldsymbol{\zeta}_{ij}) \;\sim \mathcal{N}\left(\mathbf{0}_{18\times 1}, \boldsymbol{\Sigma}_{\mathcal{IC}_{ij}}\right), \tag{5.87}$$

is a zero-mean Gaussian whose covariance is computed using (5.81). The error residual can be expressed as

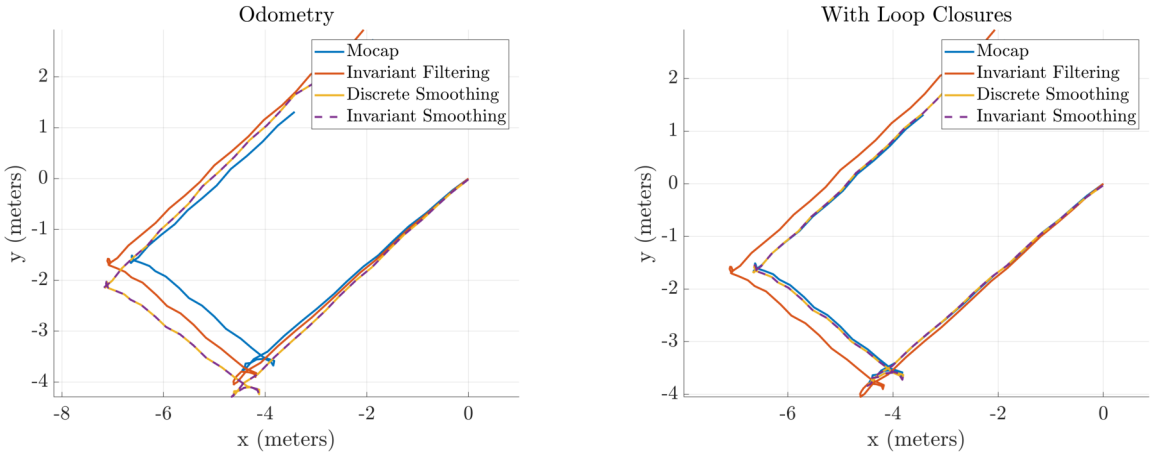
$$\begin{aligned} \mathbf{r}_f &= \operatorname{Log}\left(\operatorname{Exp}\left(-\frac{\partial f_{\Delta_{ij}}}{\partial \delta \mathbf{b}} \delta \mathbf{b}\right) f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i)^{-1} \mathbf{X}_j\right) \\ \mathbf{r}_b &= \mathbf{b}_j - \mathbf{b}_i, \end{aligned} \tag{5.88}$$

where the negative sign comes from inverting the exponential map. The residual covariance is simply $\boldsymbol{\Sigma}_{\mathcal{IC}_{ij}}$. This factor can be inserted into the factor graph optimization problem (5.4).

5.6.6 Experimental Results on Cassie

To test these contact-aided smoothing ideas experimentally, we performed a motion capture experiment in the University of Michigan's M-Air facility (shown in Figure 5.10). This outdoor space is equipped with 18 Qualisys cameras that allows for position tracking. We had the Cassie robot walk untethered for 60 sec along an approximately 15 m path. We first built a factor graph using only the inertial, contact, and kinematic data. Graph nodes were added once per second to simulate low-frequency camera keyframes. Figure 5.11a shows the result of optimizing this graph. The "discrete smoothing line" (yellow) shows the odometry result when using the hybrid point contact factor described in Section 5.5. The "invariant smoothing line" (purple) shows the odometry result when using the invariant hybrid inertial-contact factor described in this section. For this experiment, both version converged to approximately the same solution. The red line shows the InEKF odometry from Chapter 4. In general, one would expect smoothing to outperform filtering due to the ability to re-linearize measurements. However, in this case, the InEKF odometry appears to outperform the smoothing results. This can be explained through two observations. First, as indicated in Chapter 4, the developed InEKF does not suffer from the typical linearization errors of typical filtering methods. This diminishes the re-linearization advantage that the

Figure 5.10: Cassie walking in MAir for a motion capture experiment.



(a) Without loop closures

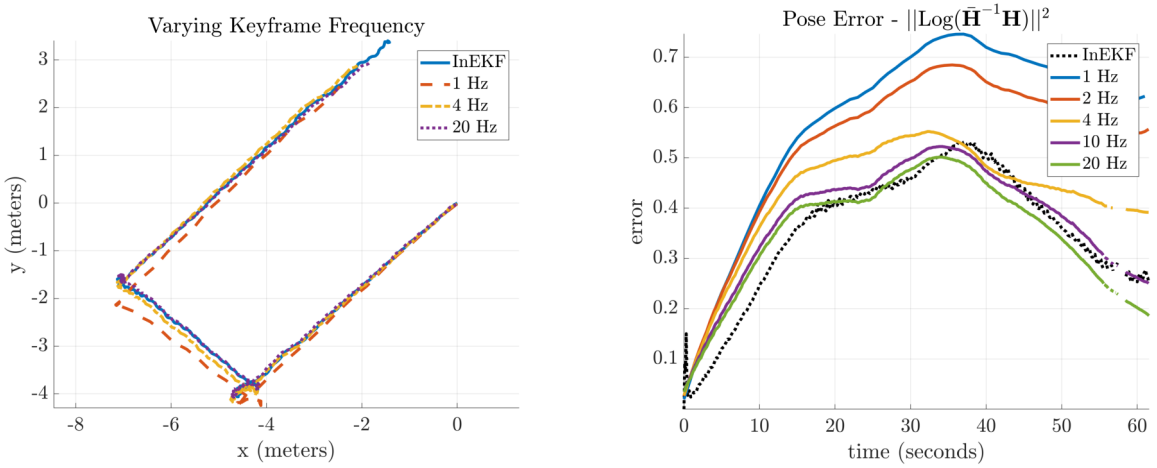
(b) With loop closures

Figure 5.11: Motion capture experiment with Contact-aided smoothing. Figure (a) shows the results of optimizing a factor graph containing inertial, contacts, and kinematics data. Figure (b) shows the result when 3 simulated loop closures are added (one every 20 seconds).

smoothing methods have[2]. Second, the forward kinematic correction are applied at 2000 Hz for the InEKF, while forward kinematic corrections for the smoothing methods are only applied at the keyframe rate (1 Hz here). In fact, increasing the keyframe rate pushes the smoothing results closer to the InEKF odometry, as shown in Figure 5.12.

The main advantage contact-aided smoothing provides is the ability to perform loop

[2]Of course, the re-linearization advantage appears again if additional (non-invariant) observations are included.



(a) Trajectory Estimate

(b) Pose Error

Figure 5.12: As the frequency of keyframes increases, more nodes are added to the graph. This results in improved state estimation due to the increasing number of forward kinematic factors added to the graph.

closures. Figure 5.11b shows the result of optimizing the factor graphs when only 3 simulated, pose-based loop closures are added (one every 20 seconds). This shows that once a few loop closures are added, the smoothing results can easily outperform the InEKF and aligns the estimated trajectory with the ground truth.

In the future, we hope to combine the developed factors with vision-based factors and loop closures to solve simultaneous localization and mapping (SLAM) for legged robots. While the work of this chapter presets a theory of contact-aided smoothing, further development and experiments need to be done to verify these results for long-term mapping.

5.7 Residual Jacobians

Solving the factor graph involves computing the *Maximum-A-Posteriori* (MAP) estimate of $\mathcal{X}_k$. This is done by solving a weighted nonlinear least-squares problem (5.4). Solvers can often by more accurate and efficient when analytical Jacobians are provided. In this section, we provide these Jacobians for all the factors presented in this chapter.

5.7.1 Forward Kinematic Pose

The forward kinematic pose factor is derived in Section 5.2. Let $\mathbf{X}_i$ and $\mathbf{C}_i$ denote the base and contact pose respectively. The relative pose transformation between these two frames, $\mathbf{H}_{\text{BC}}(\tilde{\boldsymbol{\alpha}}_i)$, can be measured through forward kinematics. The error residual is

$$\mathbf{r}_{\mathcal{F}i} = \text{Log}\left(\mathbf{C}_i^{-1}\,\mathbf{X}_i\,\mathbf{H}_{\text{BC}}(\tilde{\boldsymbol{\alpha}}_i)\right). \tag{5.89}$$

To compute the residual Jacobian, we perturb each state using the $\mathrm{Exp}(\cdot)$ retraction;

$$\mathbf{X} \leftarrow \mathbf{X}\,\mathrm{Exp}(\delta\mathbf{x}) \qquad \mathbf{C} \leftarrow \mathbf{C}\,\mathrm{Exp}(\delta\mathbf{c}) \tag{5.90}$$

A first-order approximation of the perturbed residual is then computed.

$$\begin{aligned}
\mathbf{r}_{\mathcal{F}i}\left(\mathbf{X}_i\,\mathrm{Exp}(\delta\mathbf{x}_i)\right) &= \mathrm{Log}\left(\mathbf{C}_i^{-1}\,\mathbf{X}_i\,\mathrm{Exp}(\delta\mathbf{x}_i)\,\mathbf{H}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)\right) \\
&= \mathrm{Log}\left(\mathbf{C}_i^{-1}\,\mathbf{X}_i\,\mathbf{H}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)\mathrm{Exp}(\mathrm{Ad}_{\mathbf{H}_{\mathrm{BC}}^{-1}(\tilde{\boldsymbol{\alpha}}_i)}\,\delta\mathbf{x}_i)\right) \\
&\approx \mathbf{r}_{\mathcal{F}i}(\mathbf{X}_i) + \mathbf{J}_r^{-1}(\mathbf{r}_{\mathcal{F}i}(\mathbf{X}_i))\,\mathrm{Ad}_{\mathbf{H}_{\mathrm{BC}}^{-1}(\tilde{\boldsymbol{\alpha}}_i)}\,\delta\mathbf{x}_i
\end{aligned} \tag{5.91}$$

$$\begin{aligned}
\mathbf{r}_{\mathcal{F}i}\left(\mathbf{C}_i\,\mathrm{Exp}(\delta\mathbf{c}_i)\right) &= \mathrm{Log}\left(\mathrm{Exp}(-\delta\mathbf{c}_i)\,\mathbf{C}_i^{-1}\,\mathbf{X}_i\,\mathrm{Exp}(\delta\mathbf{x}_i)\,\mathbf{H}_{\mathrm{BC}}(\tilde{\boldsymbol{\alpha}}_i)\right) \\
&\approx \mathbf{r}_{\mathcal{F}i}(\mathbf{C}_i) - \mathbf{J}_l^{-1}(\mathbf{r}_{\mathcal{F}i}(\mathbf{C}_i))\,\delta\mathbf{c}_i
\end{aligned} \tag{5.92}$$

The last line of each utilizes respectively uses the right and left Jacobian inverse of SE(3) to give a first-order approximation; see Section 3.4.4. Therefore, the forward kinematic residual Jacobians are:

$$\begin{aligned}
\frac{\partial\mathbf{r}_{\mathcal{F}i}}{\partial\delta\mathbf{x}_i} &= \mathbf{J}_r^{-1}(\mathbf{r}_{\mathcal{F}i})\,\mathrm{Ad}_{\mathbf{H}_{\mathrm{BC}}^{-1}(\tilde{\boldsymbol{\alpha}}_i)} \\
\frac{\partial\mathbf{r}_{\mathcal{F}i}}{\partial\delta\mathbf{c}_i} &= -\mathbf{J}_l^{-1}(\mathbf{r}_{\mathcal{F}i}).
\end{aligned} \tag{5.93}$$

5.7.2 Hybrid Rigid Contact

The hybrid rigid contact factor is derived in Section 5.3. Let $\Delta\tilde{\mathbf{C}}_{ij}$ be the relative contact pose transformation between t_i and t_j. The error residual is

$$\mathbf{r}_{\mathcal{C}_{ij}} = \mathrm{Log}\left(\mathbf{C}_j^{-1}\,\mathbf{C}_i\,\Delta\tilde{\mathbf{C}}_{ij}\right). \tag{5.94}$$

This residual has the same general form as the forward kinematics residual (5.89). Therefore, the Jacobian can be computed the same way, leading to

$$\begin{aligned}
\frac{\partial\mathbf{r}_{\mathcal{C}_{ij}}}{\partial\delta\mathbf{x}_i} &= \mathbf{J}_r^{-1}(\mathbf{r}_{\mathcal{C}_{ij}})\,\mathrm{Ad}_{\Delta\tilde{\mathbf{C}}_{ij}^{-1}} \\
\frac{\partial\mathbf{r}_{\mathcal{C}_{ij}}}{\partial\delta\mathbf{c}_i} &= -\mathbf{J}_l^{-1}(\mathbf{r}_{\mathcal{C}_{ij}}).
\end{aligned} \tag{5.95}$$

5.7.3 Hybrid Point Contact

The hybrid point contact factor is derived in Section 5.5. This error residual makes use of the preintegrated inertial measurement unit (IMU) quantities developed by Forster et al. [85], along with the new preintegrated relative contact translation $\Delta\tilde{\mathbf{d}}_{ij}$.

$$\mathbf{r}_{\mathcal{IC}_{ij}} \triangleq \begin{bmatrix} \mathbf{r}_{\Delta\mathbf{R}_{ij}} \\ \mathbf{r}_{\Delta\mathbf{v}_{ij}} \\ \mathbf{r}_{\Delta\mathbf{p}_{ij}} \\ \mathbf{r}_{\Delta\mathbf{d}_{ij}} \end{bmatrix} = \begin{bmatrix} \operatorname{Log}\left(\left(\Delta\tilde{\mathbf{R}}_{ij}\operatorname{Exp}\left(\frac{\partial\tilde{\mathbf{R}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g\right)\right)^{\mathsf{T}}\mathbf{R}_i^{\mathsf{T}}\mathbf{R}_j\right) \\ \mathbf{R}_i^{\mathsf{T}}(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) - \left[\Delta\tilde{\mathbf{v}}_{ij} + \frac{\partial\tilde{\mathbf{v}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g + \frac{\partial\tilde{\mathbf{v}}_{ij}}{\partial\mathbf{b}^a}\delta\mathbf{b}^a\right] \\ \mathbf{R}_i^{\mathsf{T}}(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}) - \left[\Delta\tilde{\mathbf{p}}_{ij} + \frac{\partial\tilde{\mathbf{p}}_{ij}}{\partial\mathbf{b}^g}\delta\mathbf{b}^g + \frac{\partial\tilde{\mathbf{p}}_{ij}}{\partial\mathbf{b}^a}\delta\mathbf{b}^a\right] \\ \mathbf{R}_i^{\mathsf{T}}(\mathbf{d}_j - \mathbf{d}_i) - \left[\Delta\tilde{\mathbf{d}}_{ij} + \frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g\right] \end{bmatrix} \tag{5.96}$$

The Jacobians for the first three terms are provided in [84], so only the Jacobian of $\mathbf{r}_{\Delta\mathbf{d}_{ij}}$ is derived here. Forster et al. [85] used an alternative retraction;

$$\mathbf{R} \leftarrow \mathbf{R}\operatorname{Exp}(\delta\boldsymbol{\phi}) \qquad \mathbf{v} \leftarrow \mathbf{v} + \delta\mathbf{v} \qquad \mathbf{p} \leftarrow \mathbf{p} + \mathbf{R}\delta\mathbf{p}. \tag{5.97}$$

Since this factor is formulated as an extension to their preintegrated IMU factor, we adopt a similar retraction for the contact position;

$$\mathbf{d} \leftarrow \mathbf{d} + \mathbf{R}\delta\mathbf{d}. \tag{5.98}$$

The Jacobian is again computed through first-order approximation of the perturbed residual.

$$\begin{aligned} \mathbf{r}_{\Delta\mathbf{d}_{ij}}(\mathbf{R}_i\operatorname{Exp}(\delta\boldsymbol{\phi}_i)) &= \operatorname{Exp}(-\delta\boldsymbol{\phi})\,\mathbf{R}_i^{\mathsf{T}}(\mathbf{d}_j - \mathbf{d}_i) - \Delta\tilde{\mathbf{d}}_{ij} - \frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g \\ &\approx \left(\mathbf{I} - (\delta\boldsymbol{\phi})_\times\right)\mathbf{R}_i^{\mathsf{T}}(\mathbf{d}_j - \mathbf{d}_i) - \Delta\tilde{\mathbf{d}}_{ij} - \frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g \\ &= \mathbf{r}_{\Delta\mathbf{d}_{ij}}(\mathbf{R}_i) + \left(\mathbf{R}_i^{\mathsf{T}}(\mathbf{d}_j - \mathbf{d}_i)\right)_\times\delta\boldsymbol{\phi} \end{aligned} \tag{5.99}$$

$$\begin{aligned} \mathbf{r}_{\Delta\mathbf{d}_{ij}}(\mathbf{d}_i + \mathbf{R}_i\delta\mathbf{d}_i) &= \mathbf{R}_i^{\mathsf{T}}(\mathbf{d}_j - \mathbf{d}_i - \mathbf{R}_i\delta\mathbf{d}_i) - \Delta\tilde{\mathbf{d}}_{ij} - \frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g \\ &= \mathbf{r}_{\Delta\mathbf{d}_{ij}}(\mathbf{d}_i) - \delta\mathbf{d}_i \end{aligned} \tag{5.100}$$

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{d}_{ij}}(\mathbf{d}_j + \mathbf{R}_j\delta\mathbf{d}_j) &= \mathbf{R}_i^\mathsf{T}(\mathbf{d}_j + \mathbf{R}_j\delta\mathbf{d}_j - \mathbf{d}_i) - \Delta\tilde{\mathbf{d}}_{ij} - \frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}\delta\mathbf{b}^g \\ &= \mathbf{r}_{\Delta\mathbf{d}_{ij}}(\mathbf{d}_i) + \mathbf{R}_i^\mathsf{T}\mathbf{R}_j\delta\mathbf{d}_i\end{aligned} \tag{5.101}$$

$$\begin{aligned}\mathbf{r}_{\Delta\mathbf{d}_{ij}}(\delta\mathbf{b}^g + \delta\hat{\mathbf{b}}^g) &= \mathbf{R}_i^\mathsf{T}(\mathbf{d}_j - \mathbf{d}_i) - \Delta\tilde{\mathbf{d}}_{ij} - \frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}(\delta\mathbf{b}^g + \delta\hat{\mathbf{b}}^g) \\ &= \mathbf{r}_{\Delta\mathbf{d}_{ij}}(\mathbf{d}_i) - \frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}\delta\hat{\mathbf{b}}^g\end{aligned} \tag{5.102}$$

Therefore, the residual Jacobians are:

$$\begin{aligned}\frac{\partial\mathbf{r}_{\Delta\mathbf{d}_{ij}}}{\partial\delta\boldsymbol{\phi}_i} &= \left(\mathbf{R}_i^\mathsf{T}(\mathbf{d}_j - \mathbf{d}_i)\right)_\times \\ \frac{\partial\mathbf{r}_{\Delta\mathbf{d}_{ij}}}{\partial\delta\mathbf{d}_i} &= -\mathbf{I} \\ \frac{\partial\mathbf{r}_{\Delta\mathbf{d}_{ij}}}{\partial\delta\mathbf{d}_j} &= \mathbf{R}_i^\mathsf{T}\mathbf{R}_j \\ \frac{\partial\mathbf{r}_{\Delta\mathbf{d}_{ij}}}{\partial\delta\hat{\mathbf{b}}^g} &= -\frac{\partial\Delta\tilde{\mathbf{d}}_{ij}}{\partial\delta\mathbf{b}^g}.\end{aligned} \tag{5.103}$$

5.7.4 Forward Kinematic Position

The forward kinematic position factor is derived in Section 5.5.6. This error residual is expressed as

$$\mathbf{r}_{\mathcal{F}_i} = \mathbf{R}_i^\mathsf{T}(\mathbf{d}_i - \mathbf{p}_i) - {}_\mathrm{B}\mathbf{p}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_i), \tag{5.104}$$

where ${}_\mathrm{B}\mathbf{p}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_i)$ denotes the position of the contact frame relative to the base frame as measured by forward kinematics. Sticking with the retraction (5.97) used by Forster et al. [85], and newly defined contact position retraction (5.98), we can again compute the Jacobian through first-order approximation of the perturbed residual.

$$\begin{aligned}\mathbf{r}_{\mathcal{F}_i}(\mathbf{R}_i\,\mathrm{Exp}(\delta\boldsymbol{\phi}_i)) &= \mathrm{Exp}(-\delta\boldsymbol{\phi}_i)\,\mathbf{R}_i^\mathsf{T}(\mathbf{d}_i - \mathbf{p}_i) - {}_\mathrm{B}\mathbf{p}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_i) \\ &\approx \mathbf{r}_{\mathcal{F}_i}(\mathbf{R}_i) + \left(\mathbf{R}_i^\mathsf{T}(\mathbf{d}_i - \mathbf{p}_i)\right)_\times\end{aligned} \tag{5.105}$$

$$\begin{aligned}\mathbf{r}_{\mathcal{F}_i}(\mathbf{p}_i + \mathbf{R}_i\delta\mathbf{p}_i)) &= \mathbf{R}_i^\mathsf{T}(\mathbf{d}_i - \mathbf{p}_i - \mathbf{R}_i\delta\mathbf{p}_i) - {}_\mathrm{B}\mathbf{p}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_i) \\ &= \mathbf{r}_{\mathcal{F}_i}(\mathbf{R}_i) - \delta\mathbf{p}_i\end{aligned} \tag{5.106}$$

$$\begin{aligned}\mathbf{r}_{\mathcal{F}_i}(\mathbf{d}_i + \mathbf{R}_i\delta\mathbf{d}_i)) &= \mathbf{R}_i^\mathsf{T}(\mathbf{d}_i + \mathbf{R}_i\delta\mathbf{d}_i - \mathbf{p}_i) - {}_\mathrm{B}\mathbf{p}_\mathrm{BC}(\tilde{\boldsymbol{\alpha}}_i) \\ &= \mathbf{r}_{\mathcal{F}_i}(\mathbf{R}_i) + \delta\mathbf{d}_i\end{aligned} \tag{5.107}$$

Therefore, the residual Jacobians are:

$$\begin{aligned}\frac{\partial \mathbf{r}_{\mathcal{F}_i}}{\partial \delta \boldsymbol{\phi}_i} &= \left(\mathbf{R}_i^\mathsf{T}(\mathbf{d}_i - \mathbf{p}_i)\right)_\times \\ \frac{\partial \mathbf{r}_{\mathcal{F}_i}}{\partial \delta \mathbf{p}_i} &= -\mathbf{I} \\ \frac{\partial \mathbf{r}_{\mathcal{F}_i}}{\partial \delta \mathbf{d}_i} &= \mathbf{I}.\end{aligned} \tag{5.108}$$

5.7.5 Invariant Inertial-Contact

The invariant inertial-contact factor was derived in Section 5.6. The group component of the error residual is expressed as

$$\mathbf{r}_f = \operatorname{Log}\left(\operatorname{Exp}\left(-\frac{\partial f_{\Delta_{ij}}}{\partial \delta \mathbf{b}}\delta\mathbf{b}\right)\, f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i)^{-1}\mathbf{X}_j\right) \tag{5.109}$$

The unbiased, discrete preintegrated dynamics function, $f_{\Delta_{ij}}(\mathbf{X}_i)$, can be shown to satisfy the following "discrete group-affine" property [25]:

$$f(\mathbf{X}_1\mathbf{X}_2) = f(\mathbf{X}_1)\, f(\mathbf{I}_d)^{-1}\, f(\mathbf{X}_2). \tag{5.110}$$

Therefore, the left-invariant error dynamics can be expressed as

$$g_{\Delta_{ij}}(\boldsymbol{\eta}_t^l) = f_{\Delta_{ij}}(\mathbf{I}_d)^{-1}\, f_{\Delta_{ij}}(\boldsymbol{\eta}_t^l) = \operatorname{Exp}(\mathbf{F}_{\Delta_{ij}}\boldsymbol{\xi}_t), \tag{5.111}$$

where

$$\mathbf{F}_{\Delta_{ij}} = \begin{bmatrix} \Delta\tilde{\mathbf{R}}_{ij}^\mathsf{T} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\Delta\tilde{\mathbf{R}}_{ij}^\mathsf{T}\left(\Delta\tilde{\mathbf{v}}_{ij}\right)_\times & \Delta\tilde{\mathbf{R}}_{ij}^\mathsf{T} & \mathbf{0} & \mathbf{0} \\ -\Delta\tilde{\mathbf{R}}_{ij}^\mathsf{T}\left(\Delta\tilde{\mathbf{p}}_{ij}\right)_\times & \Delta\tilde{\mathbf{R}}_{ij}^\mathsf{T}\Delta t_{ij} & \Delta\tilde{\mathbf{R}}_{ij}^\mathsf{T} & \mathbf{0} \\ -\Delta\tilde{\mathbf{R}}_{ij}^\mathsf{T}\left(\Delta\tilde{\mathbf{d}}_{ij}\right)_\times & \mathbf{0} & \mathbf{0} & \Delta\tilde{\mathbf{R}}_{ij}^\mathsf{T} \end{bmatrix} \tag{5.112}$$

can be constructed from the preintegrated measurements. The following Jacobian derivation assumes that the retraction is the exponential map on the right-hand side,

$$\mathbf{X} \leftarrow \mathbf{X}\operatorname{Exp}(\delta\mathbf{x}). \tag{5.113}$$

To keep notation simple, let $\mathbf{B}_i \triangleq \operatorname{Exp}\left(\frac{\partial f_{\Delta_{ij}}}{\partial \delta \mathbf{b}_i}\delta\mathbf{b}_i\right)$ be the first-order bias update. The Jacobian is again computed by perturbing the state using the chosen retraction. Perturbing

$\mathbf{X}_i$ gives

$$\begin{aligned}
\mathbf{r}_f(\mathbf{X}_i \operatorname{Exp}(\boldsymbol{\xi}_i)) &= \operatorname{Log}\left(\mathbf{B}_i^{-1}\left(f_{\Delta_{ij}}\left(\mathbf{X}_i \operatorname{Exp}(\boldsymbol{\xi}_i), \bar{\mathbf{b}}_i\right)\right)^{-1}\mathbf{X}_j\right) \\
&= \operatorname{Log}\left(\mathbf{B}_i^{-1}\left[f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i)\, f_{\Delta_{ij}}(\mathbf{I}_d, \bar{\mathbf{b}}_i)^{-1}\, f_{\Delta_{ij}}(\operatorname{Exp}(\boldsymbol{\xi}), \bar{\mathbf{b}}_i)\right]^{-1}\mathbf{X}_j\right) \\
&= \operatorname{Log}\left(\mathbf{B}_i^{-1}\left[f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i) \operatorname{Exp}(\mathbf{F}_{\Delta_{ij}}\boldsymbol{\xi}_i)\right]^{-1}\mathbf{X}_j\right) \\
&= \operatorname{Log}\left(\mathbf{B}_i^{-1} \operatorname{Exp}(-\mathbf{F}_{\Delta_{ij}}\boldsymbol{\xi}_i)\, f_{\Delta_{ij}}(\mathbf{X}_i, , \bar{\mathbf{b}}_i)^{-1}\mathbf{X}_j\right) \\
&= \operatorname{Log}\left(\operatorname{Exp}(-\operatorname{Ad}_{\mathbf{B}_i^{-1}}\mathbf{F}_{\Delta_{ij}}\boldsymbol{\xi}_i)\, \mathbf{B}_i^{-1}\, f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i)^{-1}\mathbf{X}_j\right) \\
&\approx \mathbf{r}_f(\mathbf{X}_i) - \mathbf{J}_l^{-1}(\mathbf{r}_f(\mathbf{X}_i)) \operatorname{Ad}_{\mathbf{B}_i^{-1}}\mathbf{F}_{\Delta_{ij}}\boldsymbol{\xi}_i,
\end{aligned} \tag{5.114}$$

where (5.110) is used to separate the perturbation from the dynamics, and (5.111) is used to represent the error dynamics as a log-linear system. The final approximation utilizes the left Jacobian inverse of $SE_K(3)$, $\mathbf{J}_l^{-1}(\cdot)$, to linearize; see Section 3.4. Perturbing $\mathbf{X}_j$ gives

$$\begin{aligned}
\mathbf{r}_f(\mathbf{X}_j \operatorname{Exp}(\boldsymbol{\xi}_j)) &= \operatorname{Log}\left(\mathbf{B}_i^{-1}\left(f_{\Delta_{ij}}\left(\mathbf{X}_i, \bar{\mathbf{b}}_i\right)\right)^{-1}\mathbf{X}_j \operatorname{Exp}(\boldsymbol{\xi}_j)\right) \\
&\approx \mathbf{r}_f(\mathbf{X}_j) + \mathbf{J}_r^{-1}(\mathbf{r}_f(\mathbf{X}_j))\, \boldsymbol{\xi}_j,
\end{aligned} \tag{5.115}$$

where $\mathbf{J}_r^{-1}(\cdot)$ is the right Jacobian inverse of $\mathrm{SE}_K(3)$. Perturbing the bias states gives

$$\begin{aligned}
\mathbf{r}_f(\delta\mathbf{b} + \delta\hat{\mathbf{b}}) &= \operatorname{Log}\left(\operatorname{Exp}\left(-\frac{\partial f_{\Delta_{ij}}}{\partial \delta\mathbf{b}}(\delta\mathbf{b} + \delta\hat{\mathbf{b}})\right)\, f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i)^{-1}\mathbf{X}_j\right) \\
&\approx \operatorname{Log}\left(\operatorname{Exp}\left(-\mathbf{J}_l\left(-\frac{\partial f_{\Delta_{ij}}}{\partial \delta\mathbf{b}}\delta\mathbf{b}\right)\delta\hat{\mathbf{b}}\right)\operatorname{Exp}\left(-\frac{\partial f_{\Delta_{ij}}}{\partial \delta\mathbf{b}}\delta\mathbf{b}\right)\, f_{\Delta_{ij}}(\mathbf{X}_i, \bar{\mathbf{b}}_i)^{-1}\mathbf{X}_j\right) \\
&\approx \mathbf{r}_f(\delta\mathbf{b}) - \mathbf{J}_l^{-1}(\mathbf{r}_f(\delta\mathbf{b}))\, \mathbf{J}_l\left(-\frac{\partial f_{\Delta_{ij}}}{\partial \delta\mathbf{b}}\delta\mathbf{b}\right)\delta\hat{\mathbf{b}},
\end{aligned} \tag{5.116}$$

where the left Jacobian of $SE_K(3)$, $\mathbf{J}_l(\cdot)$, is used to approximately separate the perturbation from the exponential map. All together, the residual Jacobians can be expressed as

$$\begin{aligned}
\frac{\partial \mathbf{r}_f}{\partial \boldsymbol{\xi}_i} &= -\mathbf{J}_l^{-1}(\mathbf{r}_f) \operatorname{Ad}_{\mathbf{B}_i^{-1}}\mathbf{F}_{\Delta_{ij}} \\
\frac{\partial \mathbf{r}_f}{\partial \boldsymbol{\xi}_j} &= \mathbf{J}_r^{-1}(\mathbf{r}_f) \\
\frac{\partial \mathbf{r}_f}{\partial \delta\hat{\mathbf{b}}} &= -\mathbf{J}_l^{-1}(\mathbf{r}_f)\, \mathbf{J}_l\left(-\frac{\partial f_{\Delta_{ij}}}{\partial \delta\mathbf{b}}\delta\mathbf{b}\right).
\end{aligned} \tag{5.117}$$

If we make the assumption that the residual error is small, the left/right Jacobians can be approximated with the identity element. Therefore, the residual Jacobians can be approxi-

mated as

$$\begin{aligned}
\frac{\partial \mathbf{r}_f}{\partial \boldsymbol{\xi}_i} &= -\mathrm{Ad}_{\mathbf{B}_i^{-1}} \mathbf{F}_{\Delta_{ij}} \\
\frac{\partial \mathbf{r}_f}{\partial \boldsymbol{\xi}_j} &= \mathbf{I} \\
\frac{\partial \mathbf{r}_f}{\partial \delta \hat{\mathbf{b}}} &= -\mathbf{J}_l \left(-\frac{\partial f_{\Delta_{ij}}}{\partial \delta \mathbf{b}} \delta \mathbf{b} \right).
\end{aligned} \tag{5.118}$$

This is in line with the invariant smoothing framework developed by Chauchat et al. [48]. Using this approximation, the Jacobians only depend on the bias terms.

CHAPTER 6

Gait and Control Design for Underactuated Legged Robots

6.1 Overview

This chapter presents ideas on how to both generate and stabilize dynamically feasible gaits on 3D underactuated legged robots using the hybrid zero dynamics (HZD) framework [233]. Gait design is done through trajectory optimization where we are searching for a trajectory that satisfies the hybrid dynamic model of legged locomotion subject to a number of constraints (such as torque limits, joint limits, friction cone, foot clearance, and periodicity constraints) [116]. The result of this optimization is a single open-loop trajectory that exhibits a desired behavior (like walking in place). The remaining challenge is to develop a feedback controller that stabilizes the walking motion. Standard controller techniques, such as input-output linearization, can be used to stabilize a set of outputs or virtual constraints [233]. However, due to underactuation, some states will remain uncontrollable. The dynamics of these states are referred to as the zero dynamics. Unfortunately, for 3D walking, it is often the case that for fixed, holonomic virtual constraint trajectories, the zero dynamics are unstable. This led to the use of heuristic foot placement strategies, gait libraries, and more general control policies that modify the virtual constraint trajectories online to render all states stable [59, 60, 91].

Section 6.2 describes a two-stage optimization framework can be used to generate and then stabilize a periodic orbit. The main idea is the optimization of a posture adjustment function (PAF) that modifies key virtual constraints online to stabilize a walking gait. The parameters of this PAF are found by perturbing the robot is optimization and minimizing the deviation from the nominal motion. A gait library consisting of forwards, backwards, sideways, and diagonal walking is then used to extend the region of attraction of the controller. These ideas were tested on an ATRIAS-series robot (MARLO). Section 6.3 describes the trajectory optimization used to develop a feedback controller on a Cassie-series robot. This

controller utilizes a gait library of only forward and backwards walking motions. Stability is achieved through a heuristically tuned foot-placement controller. The developed controller was tested in a variety of indoor and outdoor scenarios including concrete, grass, sand, and snow.

6.2 Using Posture Adjustments and Gait Libraries to Reject Velocity Disturbances on MARLO

The core content in this section was previously published in [107]. Co-authors include Xingye Da. and Jessy W Grizzle. Small changes have been made, however, much of the text and results remain unchanged.

6.2.1 Overview and Related Work

Legged robots have the potential to aid in disaster relief, package delivery, and terrain exploration. Related technology can even help humans regain the ability to walk through powered prosthetics [93, 244] and exoskeletons [4]. However, to be useful, legged robots need to exhibit extreme stability, characterized by their ability to remain upright in the presence of real-life terrain and velocity disturbances. While numerous design techniques exits that allow periodic walking gaits to be constructed and implemented using feedback controllers, most stabilization techniques rely on full actuation, simplified models, or hand-tuned foot placement controllers.

In this sections, we present a systematic, two-stage optimization process that designs and stabilizes a periodic walking gait using the full 3D dynamic model of the robot. Using this method, a discrete library of periodic walking gaits is designed over a range of longitudinal and lateral velocities. The gait library is then used to construct a continuous "unified" control policy that allows our underactuated ATRIAS-series robot, MARLO, to reject omnidirectional velocity disturbances.

6.2.1.1 Contributions

We propose a sequence of optimization techniques that, when applied sequentially, can generate and stabilize periodic orbits for underactuated dynamical systems. This work builds upon a recent optimization framework developed by Hereid et al. [117] and a "controller interpolation" technique used by Da et al. [59] to generate biped walking gaits that cover a range of operating conditions. The primary contributions include:

- introducing a parametric means of adjusting a robot's posture to exponentially stabilize periodic walking of an underactuated 3D biped (more general than foot placement);
- selection of these controller parameters through optimizing a full-order dynamic model excited by a set of judiciously chosen perturbations that move the robot away from its nominal motion;
- an extension of the "gait library" method of [59] from a 2D to a 3D robot model;
- an overall controller that allows the robot to reject both lateral and longitudinal velocity disturbances.

This work culminates with a series of simulations and preliminary experiments that show an ATRIAS-series robot stably stepping in place and rejecting both longitudinal and lateral force perturbations.

6.2.1.2 Related Work

The majority of 3D bipedal gaits are stabilized through control of the zero-moment point (ZMP) [119, 131, 142]. In general, this method produces "quasi-static" gaits that are not robust to the loss of actuation that occurs during foot roll. Therefore, without auxiliary controllers, stability is lost when large perturbations cause the ZMP to move to the edge of the support polygon [225]. In addition, the ZMP criterion is not necessary for stability, thereby artificially restricting the variety of allowable gaits [184]. Furthermore, because this method requires having a non-zero support polygon, it is not applicable to robots with curved, line, or point feet.

Another common approach to gait stabilization can be broadly grouped into "foot placement" techniques. These methods typically rely on a simplified model to determine a desired swing foot position for stability [221]. While foot placement has enjoyed considerable success on a number of robot platforms [189, 179, 194], there is no agreed upon method for determining where to place the foot. The simplified model used and the procedure for calculating the desired swing foot position vary throughout the literature.

Hodgins [124] used the notion of a "neutral point", the foot position that will leave the velocity unchanged, to determine the desired location of the swing foot. If the foot is displaced forwards or backwards from the neutral point, then the robot will slow down or speed up. The displacement from the neutral point was a hand-tuned function of velocity error. Pratt and Tedrake [184] used the closed-form solution of the linear inverted pendulum (LIP) proposed by Kajita and Tani [140] to construct a foot placement policy. Further ways to estimate the desired foot position are through the notion of capture regions [185], the

foot placement estimator (FPE) [235], or the foot placement indicator (FPI) [224]. Foot placement algorithms are able to handle larger disturbances than ZMP-based control, and can work for underactuated systems. However, most of these methods only explore changing the swing foot position. In contrast, the technique described in this paper allows adjustment to any controlled part of the robot's posture (pitch, stance knee angle, etc.). Allowing the entire robot's posture to change can potentially enhance the ability to reject perturbations.

There have been several other notable methods for 3D biped gait stabilization outside of ZMP and "foot-placement" algorithms. Chevallereau et al. [52] used optimization to directly generate a stable gait/controller by constraining the eigenvalues of the linearized Poincaré map to be strictly within the unit circle. Akbari Hamed et al. [6] performed a sensitivity analysis of the Poincaré map resulting in a method for stabilizing precomputed periodic gaits using bilinear matrix inequalities. Griffin and Grizzle [95] investigated using nonholonomic virtual constraints to implement velocity-based posture regulation that accounts for the full dynamics of the robot.

Lastly, and perhaps most similar to our work, Post and Schmiedeler [182] used heuristically chosen, hand-tuned adjustments to step length and torso pitch of a planar biped to regulate velocity and achieve stabilization. In this paper we seek to reduce this tuning process by using finite-horizon optimization to solve for a posture adjustment function. In addition, the proposed method extends to stabilizing 3D biped gaits. Finally, we show that the region of attraction can be extended by leveraging information from numerous distinct periodic gaits.

6.2.2 Stabilizing Periodic Gaits through Finite-Horizon Parameter Optimization

To generate an exponentially-orbitally-stable walking gait, hereafter called a stable walking gait, we propose decomposing the problem into two distinct stages. First, trajectory optimization is used to generate a feasible periodic orbit. Associated with this orbit will be a nominal feedback controller that does not need to stabilize the system. Second, a separate parameter optimization is run to generate stabilizing actions. Taken together, this two-stage process gives a systematic way to both design and stabilize a periodic orbit in an underactuated robot.

During the first stage, we assume that a dynamically feasible periodic orbit has been found. There are no requirements on the method used to generate this orbit. For our robot, we employ a fast, direct-collocation-based trajectory optimization framework developed by Hereid et al. [117]. The hybrid dynamics model used for both optimization and simula-

tion is described in Section 3.1. The robot model used during optimization is described in Section 3.1.2. Once found, we denote the state trajectory along the periodic orbit as $\mathbf{x}_O(t)$.

6.2.2.1 Parameterized Feedback Controller

In addition, we assume a parameterized locally Lipschitz continuous feedback controller

$$\mathbf{u}(t) = \mathbf{\Gamma}(\mathbf{x}, \boldsymbol{\alpha}, t) \tag{6.1}$$

that for a given constant value of $\boldsymbol{\alpha} \in \mathbb{R}^N$ induces a periodic orbit corresponding to $\mathbf{x}_O(t)$. This fixed value is hereafter denoted by $\boldsymbol{\alpha}^*$. Lastly, we require that the controller parameters $\boldsymbol{\alpha}(\mathbf{x}, t)$, when allowed to vary with state and time, adjust the posture of the robot. The nominal controller ($\boldsymbol{\alpha} = \boldsymbol{\alpha}^*$) is not assumed to stabilize the orbit. Stability will be achieved through a second stage in the control design process. By not imposing stability of the nominal gait/controller, a much wider range of controllers and gait optimization methods can be employed.

For our robot, $\Gamma(\cdot)$ is an input-output linearizing controller derived using the hybrid zero dynamics (HZD) framework [233], and the controller parameters $\boldsymbol{\alpha}$ affect the robot's nominal posture in its key degrees of freedom (torso pitch, torso roll, swing leg angle, swing hip angle, and knee angles). It is now clear in what sense this is an extension of "foot placement". In our robot, a constant value of $\boldsymbol{\alpha}^*$ induces periodic walking with zero yaw, constant longitudinal and lateral average velocities, and meeting key physical constraints, such as torque limits, friction cones, and joint limits. We know that our choice of controller can stabilize a periodic orbit because adjusting the leg and hip angles has previously been shown to stabilize gaits [59].

MARLO has 6 actuators, therefore 6 outputs are chosen to be controlled using input-output linearization. The controlled coordinates for both left and right stance are

$$\mathbf{h}_0(\mathbf{q}) = \begin{bmatrix} \text{Pitch Angle} \\ \text{Swing Leg Angle} \\ \text{Stance Knee Angle} \\ \text{Swing Knee Angle} \\ \text{Roll Angle} \\ \text{Swing Hip Angle} \end{bmatrix} = \begin{bmatrix} q_x \\ \frac{1}{2}(q_1^{sw} + q_2^{sw}) \\ q_2^{st} - q_1^{st} \\ q_2^{sw} - q_1^{sw} \\ q_y \\ q_3^{sw} \end{bmatrix}. \tag{6.2}$$

More information about our controller is given in Section 3.1.4.

6.2.2.2 Posture Adjustment Functions

We seek an adjustment to the nominal controller parameters $\boldsymbol{\alpha}^*$ of the following form:

$$\boldsymbol{\alpha}(\mathbf{x}, t) = \boldsymbol{\alpha}^* + \Delta\boldsymbol{\alpha}(\mathbf{x}, t) \tag{6.3}$$

such that

$$\Delta\boldsymbol{\alpha}(\mathbf{x}, t)|_{\mathbf{x}(t)=\mathbf{x}_O(t)} = \mathbf{0} \tag{6.4}$$

In other words, the adjustment to the nominal controller is zero when the system is on the periodic orbit. This ensures that the originally designed fixed point is unchanged in steady state. When off the periodic orbit, we want $\Delta\boldsymbol{\alpha}(\mathbf{x}, t)$ to modify the nominal controller parameters, appropriately adjusting the robot's posture to return the system back to the periodic orbit. We therefore call $\Delta\boldsymbol{\alpha}(\mathbf{x}, t)$ the posture adjustment function (PAF), because it adjusts the robot's posture to achieve stability.

One way to design a PAF that meets the criteria of 6.4 is to make $\Delta\boldsymbol{\alpha}$ depend solely on the deviation away from the nominal orbit. Let τ be a (state- or time-based) phase variable that strictly increases throughout the periodic orbit. The PAF then becomes:

$$\Delta\boldsymbol{\alpha}(\mathbf{x}, t) \coloneqq \Delta\boldsymbol{\alpha}(\delta\mathbf{x}(\tau), \boldsymbol{\beta}),\ \Delta\boldsymbol{\alpha}(0, \boldsymbol{\beta}) = \mathbf{0} \tag{6.5}$$

$$\delta\mathbf{x}(\tau) \coloneqq \begin{cases} \mathbf{x}(\tau) - \mathbf{x}_O(0), & \text{if } \tau < 0 \\ \mathbf{x}(\tau) - \mathbf{x}_O(\tau), & \text{if } \tau \in [0, 1] \\ \mathbf{x}(\tau) - \mathbf{x}_O(1), & \text{if } \tau > 1 \end{cases} \tag{6.6}$$

where $\boldsymbol{\beta}$ is a set of parameters that shapes the behavior of the PAF. The quantity $\delta\mathbf{x}(\tau)$ will be called the *transverse error*.

6.2.2.3 Finite-Horizon Optimization

The parameters $\boldsymbol{\beta}$ in the PAF, $\Delta\boldsymbol{\alpha}(\delta x(\tau), \boldsymbol{\beta})$, will be determined by minimizing a cost function defined over a finite number of steps of the robot. The cost function can be thought of as a norm on the "disturbance-to-state" response of the robot, which is inspired by the notion of Input-to-State Stability of Sontag [210]. The key steps in this process are:

- choosing a parameterization of the PAF;
- transcribing a parameter optimization problem that simulates the closed-loop dynamical system over a finite horizon;

- perturbing the system with a finite set of disturbances and defining a finite-horizon objective that measures the deviation of the closed-loop trajectory from the nominal "open-loop" one;
- solving the resulting nonlinear program to yield the optimal parameters, $\boldsymbol{\beta}$

It is desirable that the optimized PAF not only stabilize the robot, but can allow it to recover from perturbations. Motivated by the work of [97, 61], a discrete set of disturbances is introduced during the optimization. These disturbances should be selected to induce deviations in the robot's state that will be representative of "experimental reality". Here, they were selected to be force perturbations applied to the robot's base. These forces had both an x and y component, allowing the robot to be perturbed along multiple directions. The forces were chosen from the sets,

$$F_x = \{-100, 0, 100\}\, N$$
$$F_y = \{-200, 0, 200\}\, N$$

excluding the trivial zero force selection. All forces were applied for 0.1 seconds.

The algorithm to compute the finite-horizon objective is outlined as follows:

Algorithm 1 Finite-Horizon Objective

```
function COST(β)
    Initialize set of perturbations F
    while F ≠ ∅ do
        x ← x_O(0)                                  ▷ Initialize state
        select perturbation from set F
        simulate "closed-loop" system forward N steps
        compute perturbation recovery cost          ▷ (6.7)
        remove selected perturbation from set F
    J ← weighted sum of recovery costs              ▷ (6.8)
    return J
```

The perturbation recovery cost measures the deviation of the robot's "closed-loop" trajectory from the nominal periodic orbit. Let $\delta(\mathbf{x}_i(\tau))$ be the transverse error for step i, as defined in equation (6.6). In addition, let w_i and $\mathbf{W}_i$ respectively be a scalar and positive semi-definite matrix that allows individual steps and states to be weighted differently. The perturbation recovery cost over an N-step horizon can now be written as:

$$\mathcal{J}_k = \sum_{i=1}^{N} w_i \int_0^1 \tau \, ||\mathbf{W}_i \, \delta\mathbf{x}_i(\tau)||^2 \, d\tau \tag{6.7}$$

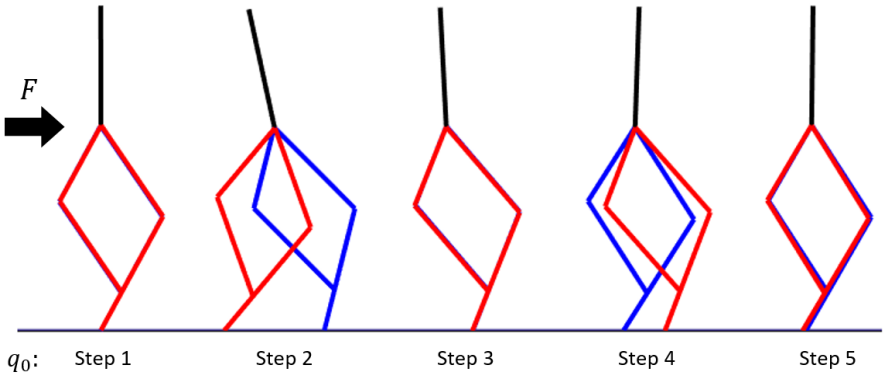


Figure 6.1: An example of the robot adjusting its posture over four steps to reject the applied force disturbance. Each frame is taken at the beginning of the corresponding step.

The inclusion of the phase variable τ in the integrand ensures that the robot's trajectory deviation at the beginning of each step is not penalized ($\tau = 0$). It is important to include this because the PAF should initially make changes to the robot's posture to reject the applied disturbance. However, by the end of the step ($\tau = 1$), the transverse error should be minimized.

For all presented results, a 4-step horizon was used with $w_i = 0$ for the first 3 steps. Therefore, to minimize the perturbation recovery cost, only the last step needs to be near the nominal trajectory. This allows the controller to command posture adjustments throughout the first three steps without affecting the cost. Figure 6.1 shows an example recovery using a 4-step optimization horizon.

This perturbation recovery cost is computed for each of the M predefined disturbances. The final cost is computed as a weighted sum of all perturbation recovery costs.

$$\mathcal{J} = \sum_{k=1}^{M} c_k \mathcal{J}_k \tag{6.8}$$

The scalar c_k allows different perturbations to be weighted individually. This would be particularly beneficial if several perturbation types were applied. In this work, we chose to weight all perturbations equally with $c_k = 1$.

Once the cost function is fully defined, the resulting nonlinear program is solved for the parameters, $\boldsymbol{\beta}$, defining the PAF. It is important to note, that the resulting controller is not guaranteed to stabilize the "closed-loop" system. Stability should be check posteriori to the optimization. It is our experience that it is not difficult to select an adequate set of perturbations and an adequate set of degrees of freedom so that the PAF achieves locally exponential orbital stability.

6.2.2.4 Simulation Results

A periodic "stepping in place" gait was generated using the trajectory optimization presented in [117]. Additional constraints were placed to enforce torque limits, friction cones, foot clearance, and other physical constraints. The objective being minimized was the sum of squared torques over the two steps. This optimization resulted in a dynamically feasible, periodic gait and a nominal HZD controller parameterized by $\boldsymbol{\alpha}^*$; see the Appendix for more details. The resulting controller was found to be unstable.

We then analyzed 3 distinct, heuristically-chosen PAFs to stabilize the robot. [1]

- Version 1 encodes a linear relationship between the velocity error, $\delta\mathbf{v}$, and adjustment to the swing leg/hip angle. This is the simplest case, and similar (hand-tuned) adjustments have been shown to stabilize our robot [59].
- Version 2 allows the swing hip angle to vary using a degree-three polynomial of lateral and longitudinal velocity errors, δv_x and δv_y. In this version, the swing leg angle changes as a function of the longitudinal velocity error and the initial stance leg error θ_{init}, which gives some information about the previous step.
- Version 3 allows all of the controlled coordinates to be adjusted linearly with velocity errors. To maintain symmetry, the knee angle adjustments use the absolute value of the longitudinal velocity.

$$
\begin{aligned}
\text{Version 1:} \quad & \begin{bmatrix} \Delta\boldsymbol{\alpha}_{\text{swing hip}} \\ \Delta\boldsymbol{\alpha}_{\text{swing leg}} \end{bmatrix} = \begin{bmatrix} \beta_1 \delta v_x \\ \beta_2 \delta v_y \end{bmatrix} \\
\text{Version 2:} \quad & \begin{bmatrix} \Delta\boldsymbol{\alpha}_{\text{swing hip}} \\ \Delta\boldsymbol{\alpha}_{\text{swing leg}} \end{bmatrix} = \begin{bmatrix} P_3(\delta v_x, \delta v_y, \boldsymbol{\beta}) \\ P_3(\delta v_y, \theta_{\text{init}}, \boldsymbol{\beta}) \end{bmatrix} \\
\text{Version 3:} \quad & \begin{bmatrix} \Delta\boldsymbol{\alpha}_{\text{pitch}} \\ \Delta\boldsymbol{\alpha}_{\text{roll}} \\ \Delta\boldsymbol{\alpha}_{\text{swing hip}} \\ \Delta\boldsymbol{\alpha}_{\text{swing leg}} \\ \Delta\boldsymbol{\alpha}_{\text{stance knee}} \\ \Delta\boldsymbol{\alpha}_{\text{swing knee}} \end{bmatrix} = \begin{bmatrix} \beta_1 \delta v_y \\ \beta_2 \delta v_x \\ \beta_3 \delta v_x \\ \beta_4 \delta v_y \\ \beta_5 |\delta v_y| \\ \beta_6 |\delta v_y| \end{bmatrix}
\end{aligned} \tag{6.9}
$$

The parameters for each of the three PAFs were computed using the finite-horizon optimization procedure outlined above. The nonlinear program was solved using MATLAB's *FMINCON* function. To compare their performance, each controller was subjected to a 100

[1] In all versions, the posture adjustment was scaled linearly over the step.

N push in the lateral plane and a 200 N push in the sagittal plane. The force perturbation was applied to the robot's base for 0.1 seconds. Figure 6.2 shows the response of the three PAFs.

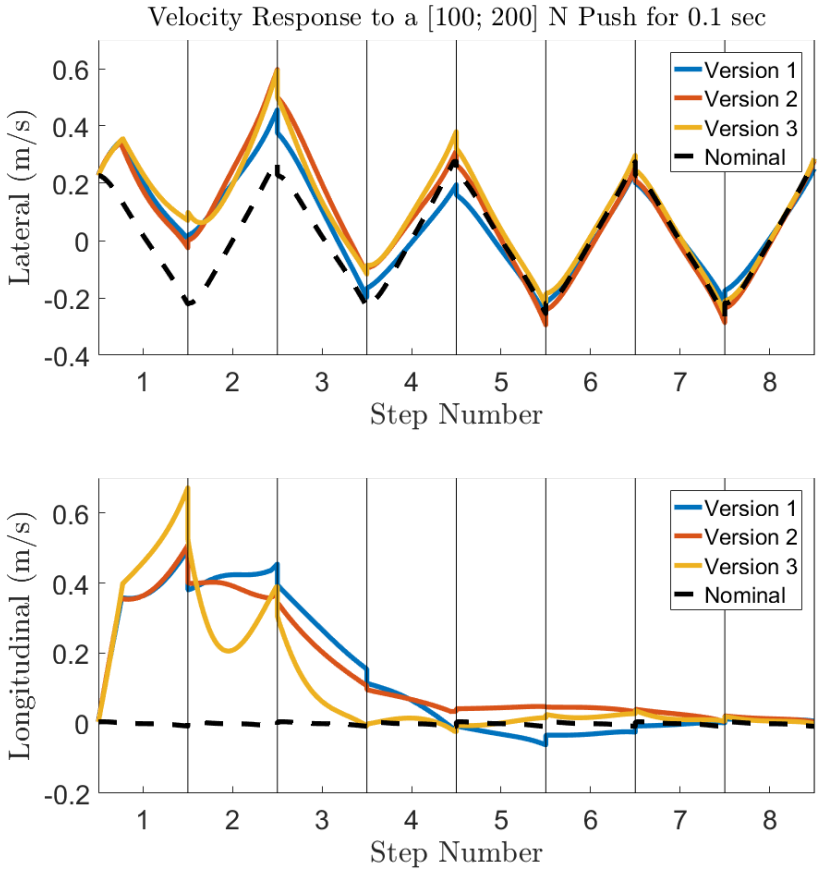


Figure 6.2: Responses of three heuristically chosen PAFs to a force perturbation. All three controllers maintained stability while converging back to the nominal zero-velocity orbit.

All three versions achieved stability while converging back to the nominal zero-velocity orbit. The lateral response was similar across the three versions. In the sagittal plane, Version 3 was the quickest to reject the perturbation. However, the maximum velocity was larger than in Versions 1 & 2. A video comparing the three responses is given in Table 7.1 .

6.2.3 Gridded-Velocity Gait Library

In the preceding section, a two-stage optimization process was outlined to design and stabilize a periodic gait. When successful, this process yields a nominal controller along with a PAF that defines how the robot's posture should change when away from the nominal trajectory. The resulting controller will allow the robot to walk at a fixed velocity and handle the range of perturbations used in the finite-horizon parameter optimization.

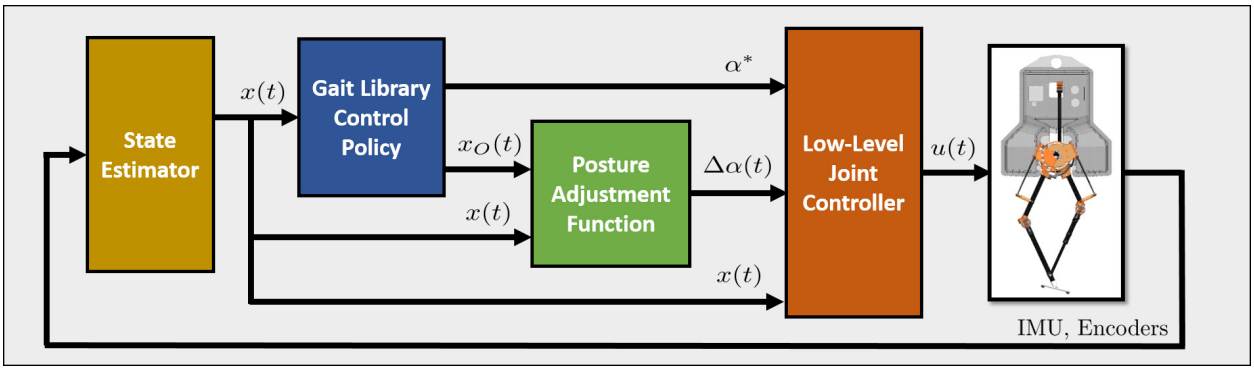


Figure 6.3: Unified Controller Block Diagram

In this section, information from numerous periodic gaits (designed over a range of velocities) is leveraged to construct a "unified" control policy. The resulting control policy is used to extend the controller's region of attraction, allowing the robot to handle perturbations outside the PAF's valid range.

6.2.3.1 Gait Library

A gait library is a finite collection of periodic gaits and controllers defined on a grid of operating conditions. The grid points can represent desired walking speeds, constant ground slopes, range of step lengths, etc. In prior work [59], the authors built a gait library consisting of periodic 2D walking gaits over a set of average longitudinal velocities. A single continuously defined controller was created by a linear interpolation of the gaits in the library based off of the current walking speed. The resulting controller was neutrally stable in velocity[2]. Stability and velocity regulation were achieved through a hand designed foot placement controller.

In this paper, we build upon this idea to create a controller that allows our robot to reject larger velocity disturbances than the PAF can handle alone. First, a series of trajectory optimizations are run to generate a gait library that contains periodic 3D walking gaits over a grid of average longitudinal and lateral velocities. This gait library includes gaits for walking forwards, backwards, sideways, and diagonally. Next, a unified control policy is defined using bilinear interpolation of the library's gaits. The interpolation input will be a desired average velocity that gets automatically shifted when the robot is perturbed outside the PAF's valid range.

[2]All but one of the eigenvalues of the linearized Poincaré were strictly within the unit circle, with the remaining eigenvalue approximately equal to +1.

6.2.3.2 Gait Library Generation

The designed gait library consists of 121 distinct periodic walking gaits, each corresponding to a unique average longitudinal and lateral velocity. The set of gaits is denoted by

$$\mathcal{A}_v = \{\boldsymbol{\alpha}^*(v_x, v_y)\} \tag{6.10}$$

where $-0.5 \leq v_x \leq 0.5$, increasing in steps of 0.1 m/s, and $-1.0 \leq v_y \leq 1.0$, increasing in steps of 0.2 m/s. Each gait was generated through trajectory optimization [117] and comes with a nominal controller, as defined by equation 6.1 .

6.2.4 Unified Control Policy through Bilinear Interpolation

To construct a unified control policy, the discrete set of gaits need to be "stitched" together. In general, this can be viewed as a supervised learning algorithm, where the set of optimized gaits is the training data [60]. The learned control policy $\pi \colon \Phi \to \mathcal{A}$ is a function that maps some feature vector $\boldsymbol{\phi} \in \Phi$ to a controller parameter vector $\boldsymbol{\alpha}^* \in \mathcal{A}$. This feature vector can, in general, be a combination of robot states, command signals, and terrain information. Once defined, the control policy will be responsible for selecting the robot's desired gait at any given time.

Since the only difference between separate gaits in $\mathcal{A}_v$ is the average velocity, the feature vector can simply chosen as $\boldsymbol{\phi} = (v_x^{des}, v_y^{des})$, where v_x^{des} and v_y^{des} are the desired lateral and longitudinal average velocities. In general, any supervised learning algorithm can be used to create the control policy [60]. However, since $\mathcal{A}_v$ is defined over a 2D grid of velocities, simple interpolation is enough to create the policy π.

Formally, the control policy $\boldsymbol{\alpha}^*(t) = \pi(v_x^{des}, v_y^{des})$ is defined using bilinear interpolation of the desired velocities,

$$\begin{aligned} \boldsymbol{\alpha}^*(t) &= K \begin{bmatrix} x_2 - v_x^{des}(t) \\ v_x^{des}(t) - x_1 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\alpha}_{11}^* & \boldsymbol{\alpha}_{12}^* \\ \boldsymbol{\alpha}_{21}^* & \boldsymbol{\alpha}_{22}^* \end{bmatrix} \begin{bmatrix} y_2 - v_y^{des}(t) \\ v_y^{des}(t) - y_1 \end{bmatrix} \\ K &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \end{aligned} \tag{6.11}$$

where (x_1, x_2) and (y_1, y_2) are the lateral and longitudinal velocity grid points that bound (v_x^{des}, v_y^{des}) in $\mathcal{A}_v$. The controller parameters defined at those grid points, $\boldsymbol{\alpha}^*(x_i, y_i)$, are denoted by $\boldsymbol{\alpha}_{ij}^*$. Figure 6.4 shows an graphical example of this interpolation.

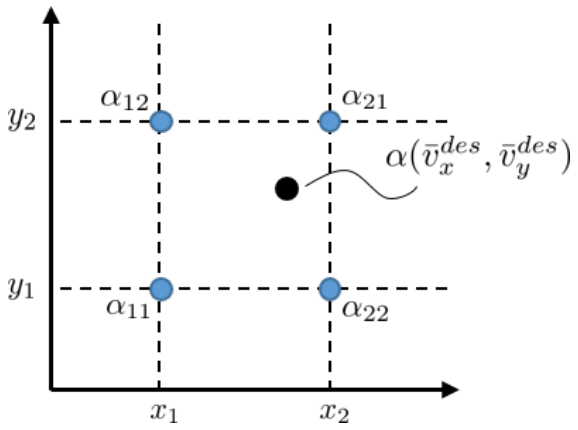


Figure 6.4: Controller parameters $\boldsymbol{\alpha}^*$ are on a uniform grid of longitudinal and lateral velocities. A bilinear interpolation is used to fit the data.

Number	Type	Description	Link
1	Simulation	Posture Adjustment Function (PAF) Comparison	`https://youtu.be/ytDTEWXDVV8`
2	Simulation	PAFs with Gait Library Control Policy	`https://youtu.be/Lte8pkmT0PQ`
3	Experiment	PAFs with Gait Library Control Policy	`https://youtu.be/PyWjTxq5PGc`

Table 6.1: Simulation/Experiment Videos for Stabilization using PAFs

6.2.5 Extending the Controller's Region of Attraction

Let $\mathbf{v}_0$ be the average velocity a periodic gait that is stabilized by a PAF; here we used the "stepping in place" gait. This parameterized PAF is only valid over the perturbation range that the system was subjected to during the finite-horizon optimization. If the robot is perturbed outside of this range, it could behave erratically. Therefore, the transverse error must be saturated at the max error values seen in optimization. To extend the region of attraction outside this perturbation range, the unified control policy can be used.

Let $\mathbf{v}(t)$ be the current velocity[3] and $\mathbf{v}^{min}$ and $\mathbf{v}_{max}$ be minimum and maximum velocity errors seen during the finite-horizon optimization. The control policy's input can be expressed as

$$\mathbf{v}^{des}(t) = \begin{cases} v(t) - \mathbf{v}_{min} & v < \mathbf{v}_{min} \\ v_0 & \mathbf{v}_{min} < v(t) < \mathbf{v}_{max} \\ v(t) - \mathbf{v}_{max} & v > \mathbf{v}_{max} \end{cases} \tag{6.12}$$

When the robot is perturbed outside the range seen during the finite-horizon optimization, the control policy will select a gait with a larger average velocity. From this new set point, the perturbation looks smaller, and the PAF can adjust appropriately. An example

[3]The lateral velocity uses the last step average to account for the non constant speed over the step.

of this is shown in Figure 6.5 . The PAF limits, $(\mathbf{v}_{min}, \mathbf{v}_{max})$, can be used to modify the velocity convergence rate. Smaller limits will decrease the maximum posture adjustment, thereby slowing down the rate of convergence to the nominal velocity. When the PAF limits are set to zero, the control policy chooses a gait corresponding to the current velocity. This reduces to the case of neutral stability discussed in [59].

To summarize, $\mathbf{v}_{des}$ is chosen to be a desired average velocity such that the velocity error is within the PAF's valid range. The control policy uses $\mathbf{v}_{des}$ to select a vector of controller parameters, $\boldsymbol{\alpha}^*$, and the corresponding nominal periodic orbit, $\mathbf{x}_O(t)$, using information from the gait library. Finally, the PAF uses this desired trajectory, along with the current state, to adjust the robot's posture to achieve stability. Figure 6.3 shows a block diagram of the full unified controller.

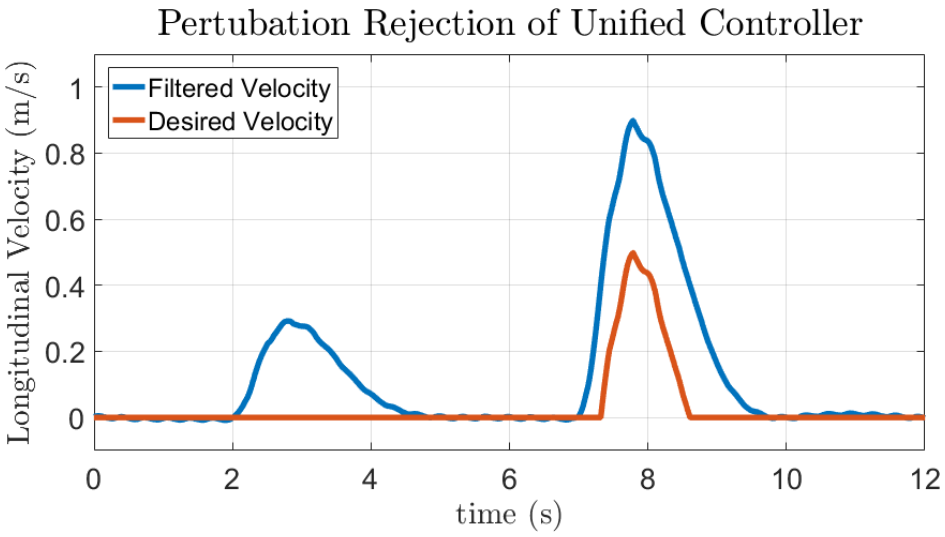


Figure 6.5: When the robot's velocity is perturbed less than 0.4 m/s, the PAF can simply reject the disturbance. If subjected to a larger velocity disturbance, the desired velocity value automatically shifts to keep the error less than 0.4 m/s.

6.2.5.1 Simulation Results

A unified controller was constructed from each of the three optimized PAFs detailed in Section 6.2.2. The PAF limits, $(\mathbf{v}_{min}, \mathbf{v}_{max})$, were chosen to be $\pm$ 0.4 m/s for the longitudinal velocity and $\pm$ 0.2 m/s for the lateral velocity.

The three controllers were subjected to several force perturbations applied for 1 second, including a 150 N push along the longitudinal axis, and a 75 N push along the lateral axis. These perturbations were enough to bring the velocity error of the robot to above 0.7 m/s laterally and 1.4 m/s longitudinally, enabling the full use of the gait library (6.10). All

three controllers were able to reject the disturbances and achieve stability. A video of the controllers' responses is given in Table 7.1.

6.2.5.2 Experimental Results

Several small adjustments were made to the controllers when translating from simulation to experiments. A low-pass filter was added to reduce spikes in the velocity estimates. This, in turn, caused a significant delay in velocity signal which introduced velocity oscillations in the perturbation response. To reduce the oscillations, a "derivative term" that compares the current filtered velocity to the last step's mean velocity was added to the PAFs as done in [59]. These oscillations were also present in simulations when the velocity signal was delayed.

In preliminary experiments, all three controllers were able to stabilize the robot, though perturbations caused significantly more velocity oscillations. This sensitivity to the delay in the velocity signal also put a limit on the maximum perturbation that the controllers could reject. Version 1 was able to withstand the largest velocity disturbance, approximately 0.8 m/s; shown in Figure 6.6. Version 2 had the least velocity oscillations and overshoot. Version 3 was the most sensitive to the velocity delay, and could therefore handle the least amount of velocity disturbance.

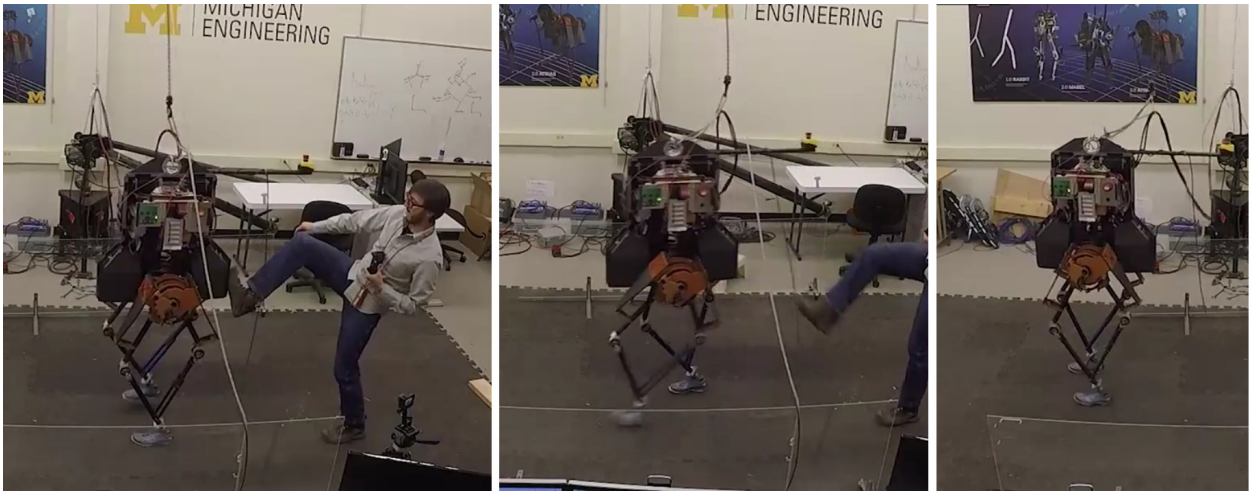


Figure 6.6: The posture adjustment function (PAF) (version 1) allows MARLO to withstand a large velocity disturbance of approximately 0.8 m/s.

Table 7.1 provides a video of these controllers implemented on MARLO. The largest kick, using PAF Version 1, brought the longitudinal velocity error to approximately 0.8 m/s.

6.2.6 Discussion and Conclusion

A systematic two-stage optimization process was presented that can design and stabilize periodic walking gaits using the full 3D dynamic model of the robot. First, a dynamically feasible (and potentially unstable) gait/controller is designed using trajectory optimization. This gait is then stabilized using a posture adjustment function (PAF) that is generated through a finite-horizon parameter optimization that emphasizes gait robustness to perturbations. Three distinct PAFs were analyzed for our robot model, each resulting in a stable controller that can reject limited velocity disturbances.

To extend the disturbance rejection range, a unified control policy was built using bilinear interpolation of 121 distinct periodic gaits. The policy's input was a desired average velocity that gets automatically shifted when the robot's velocity is perturbed outside the PAF's valid range. This policy greatly extended the controller's region of attraction allowing the simulated robot to recover from longitudinal velocity perturbations of up to 1.4 m/s and lateral perturbations of up to 0.7 m/s. Preliminary experimental results show an ATRIAS-series robot demonstrating the unified controller's capability to both stabilize a periodic orbit and to reject velocity disturbances.

Remark 16. For all results presented in this Section, the robot's velocity was being estimated through kinematics alone. This resulted in noisy signal that required further low-pass filtering. This poor/delayed velocity estimate was a factor in preventing easy translation of our controllers from simulation to experiments. This motivated the research of improved filtering techniques for legged robots, ultimately resulting in the development of the contact-aided invariant extended Kalman filter (InEKF) described in Chapter 4.

6.3 Gait Library Design for the Feedback Control of Cassie

This section outlines a gait library generation for Cassie-series robot. A description of this robot is provided in Section 3.1.3. This generated gait library was used by Gong et al. [91] to develop a feedback controller that allowed cassie to walk on concrete, grass, snow, and even sand.

6.3.1 Single Gait Optimization

First, we generated a dynamic model of the Cassie robot according to the constrained Euler Lagrange equations detailed in Section 3.1. This was done by developing a Unified Robot Description Format (URDF) file that describes the kinematic and dynamic properties of all of the robots links and joints. We then used this URDF with Fast Robot Optimization and Simulation Toolkit (FROST) [116] to generate the equations of motion for the dynamic model of the robot. FROST was also used to set up the trajectory optimization problem and to generate all constraint and cost equations (along with their Jacobians) needed for the solver. Internally, FROST transcribes the optimization problem into a nonlinear program using the direct collocation framework developed by Hereid et al. [117]. The constrained nonlinear program is then solve to a local minima using the Interior Point Optimizer (IPOPT) software [32].

In general, the direct collocation method [105] breaks the continuous time trajectory up into a set of discrete nodes. Instead of integrating the dynamics equations during the optimization (as done in shooting methods), the dynamics are enforced though equality constraints. Another key element of this particular optimization setup is the inclusion of an input-output linearization controller (described in Section 3.1.4). This provides an additional constraint on the relation between the robot' state and the motor torque.

Let $\mathcal{J}_{D_R}$ and $\mathcal{J}_{D_L}$ be the costs associated with the right stance domain and left stance domain respectively. The two-step optimization problem can now be formally stated as,

$$\arg\min_{\mathbf{z}^*} \mathcal{J}_{D_R}(\mathbf{z}) + \mathcal{J}_{D_L}(\mathbf{z}) \tag{6.13}$$

$$\text{s.t.} \quad \mathbf{z}_{\min} \leq \quad \mathbf{z} \quad \leq \mathbf{z}_{\max} \tag{6.14}$$

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{z}) \leq \mathbf{c}_{\max} \tag{6.15}$$

where $\mathbf{z}_{\min}$ and $\mathbf{z}_{\max}$ are the lower and upper optimization variable bounds, $\mathbf{c(z)}$ is a vector of constraints, and $\mathbf{c}_{\min}$ and $\mathbf{c}_{\max}$ are the constraint lower and upper bounds. The optimization variables at each node include the robot's generalized position coordinates $\mathbf{q}$, velocities $\dot{\mathbf{q}}$, accelerations $\ddot{\mathbf{q}}$, motor torques $\mathbf{u}$, holonomic constraint forces $\mathbf{F}$, virtual constraint parameters α, and impact forces $\delta\mathbf{F}$ (only needed at the end of the domain). Constraints included two-step periodicity, left/right step symmetry, joint limits, and torque limits. Additional constraints used are outlined in Table 6.2.

Table 6.2: Additional constraints included in the trajectory optimization for Cassie. For each optimized gait, the average sagittal velocity was constrained to a differing values between -0.5 and 1.0 m/s.

Average sagittal velocity	$\bar{v}_x = v_i$ m/s
Average lateral velocity	$\bar{v}_y = 0$ m/s
Step time	$= 0.4$ s
Torque for stance foot pitch	$= 0$ Nm
Friction cone	$\mu < 0.6$
Mid-step swing foot clearance	> 0.15 m
Absolute swing foot pitch	$= 0$ rad
Distance between feet	> 0.2 m
Distance between pelvis and stance foot	$\in (0.5, 1)$ m
Swing foot velocity on impact (x and y)	$= 0$ m/s
Swing foot velocity on impact (z)	$\in (-1, 0)$ m/s

The cost was equivalent for both the left and right stance domains:

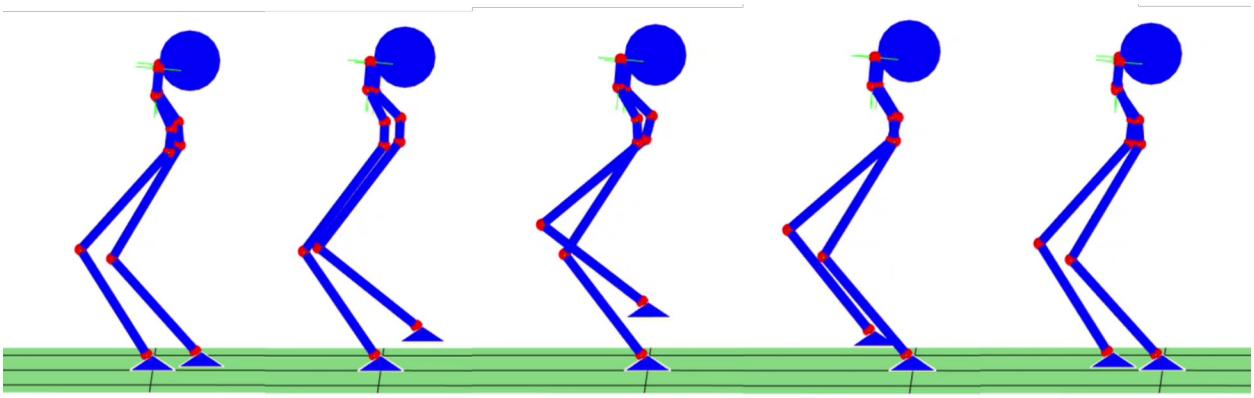
$$\mathcal{J}_D = \int_{\tau=0}^{\tau=1} \left(||u||^2 + c||q_{pitch}||^2 + c||q_{roll}||^2 + c||q_{1L}||^2 + c||q_{2L}||^2 + c||q_{1R}||^2 + c||q_{2R}||^2 \right) \quad d\tau. \tag{6.16}$$

The cost on the torso pitch/roll and the hip roll/yaw angles were multiplied by a large weight ($c = 10,000$). This guided the optimizer to find gaits with minimal movement for those degrees of freedom. This helped to keep the gaits consistent across the library of gaits.

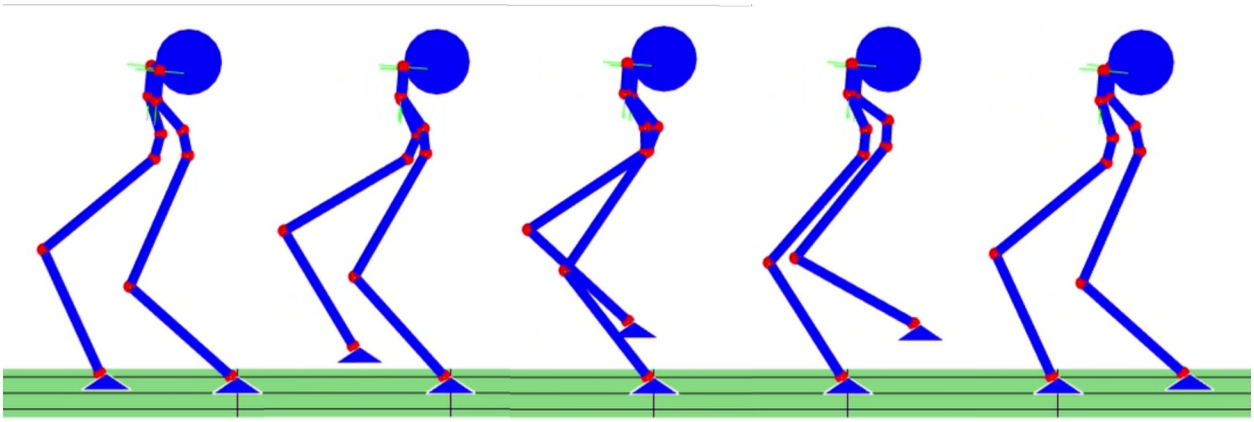
6.3.2 Sagittal Gait Library

This gait library consisted of 7 gaits where the average velocity in the sagittal plane, $\bar{v}_x$, ranged from -0.5 m/s to +1.0 m/s in 0.25 m/s increments. Each gait was generated through trajectory optimization using the costs and constraints outlined above. These optimization problems were solved sequentially where the only constraint on average velocity was changed. Each optimization problem took approximately 3 minutes to solve. In a larger number of gaits was needed, these optimizations could be done in parallel as described by Hereid et al. [118]. Figure 6.7 shows results of a few of these optimization problems.

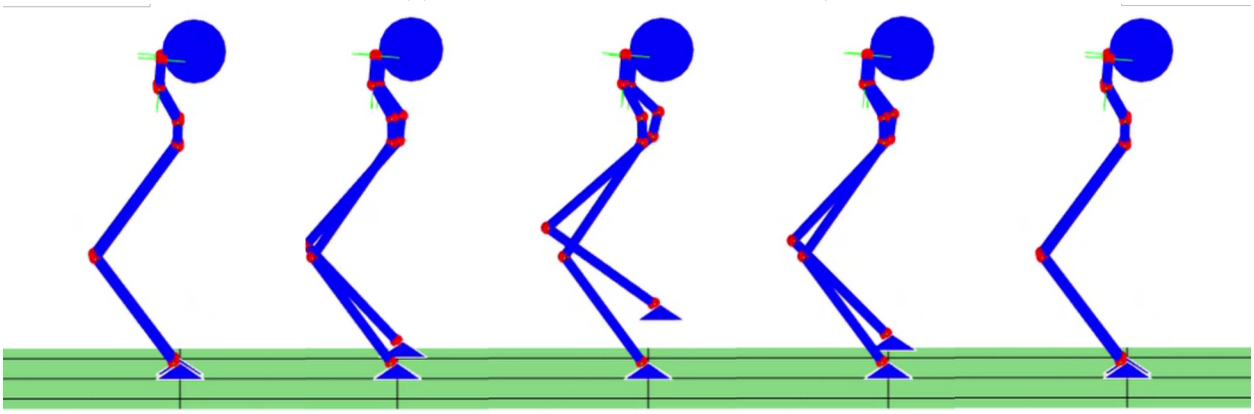
These gaits were then used by Gong et al. [91] to develop a feedback controller for Cassie. First, linear interpolation of these gaits was used to construct a continuous surface of trajectories based on average velocity. While operating, Cassie's controller chooses its current desired trajectory based on the current measured velocity. A foot placement strategy similar to the work of [59] was used for stabilization. Turning was done through simple offsets of the hip yaw angle.



(a) Cassie's optimized trajectory for $-0.5\ m/s$.



(b) Cassie's optimized trajectory for $0.0\ m/s$.



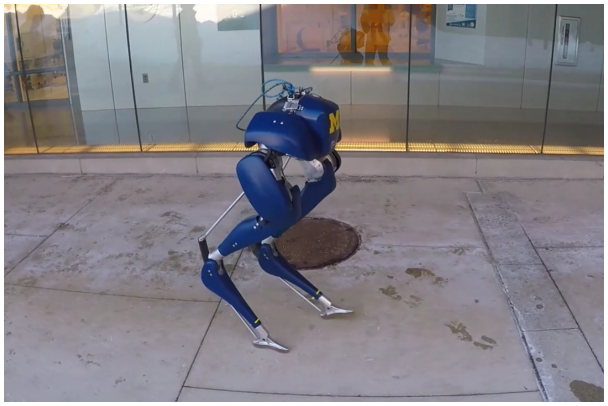
(c) Cassie's optimized trajectory for $1.0\ m/s$.

Figure 6.7: Seven gaits were generated for Cassie where the average velocity in the sagittal plane ranged from -0.5 m/s to +1.0 m/s in 0.25 m/s increments.

6.3.3 Experimental Results

Using this generated gait library, Cassie was able to walk in place, forwards, and backwards. The controller was experimentally tested across a wide variety of surfaces, including concrete, grass, snow, and even sand (as shown in Figure 6.8). A video of these results can be found at `https://youtu.be/UhXly-5tEkc`. This controller was also used for all experimental results in Chapters 4 and 5.

Remark 17. During all experiments, Cassie used a "kinematic only" estimate of velocity that was computed using the encoders and angular velocity. Due to high amounts of noise and foot slip, this velocity estimate had to be passed through a low-pass filter before being used to select a gait from the library. This heavily filtered signal was one of the reasons numerous heuristic tuning parameters had to be introduced to achieve stability. The work predated and partially motivated the developed invariant extended Kalman filter (InEKF) from Chapter 4, which can potentially alleviate some of these issues in future controllers.

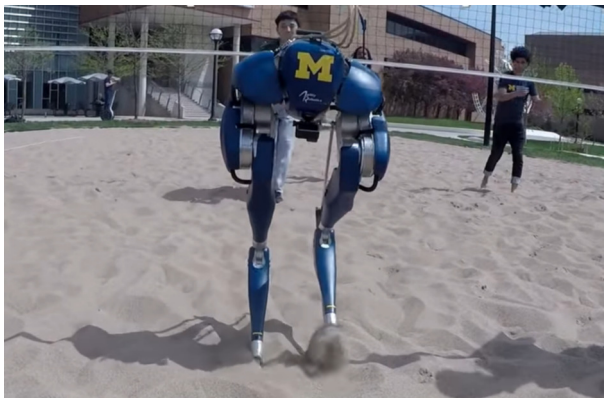


(a) Walking on concrete

(b) Walking on grass

(c) Walking on leaves, sticks, and fire during a controlled burn

(d) Walking on snow



(e) Walking on sand

(f) Walking on mud with shoes and the perception package

Figure 6.8: Cassie walking in a variety of environments. The controller was developed using a library of gaits that was generated through trajectory optimization. `https://youtu.be/UhXly-5tEkc`

CHAPTER 7

Future Research Directions

This chapter outlines several ideas for future research, some of which have already been tested in simulation. Section 7.1 discusses ideas on how to include terrain information into the gait library and how we might tackle walking up stairs. Section 7.2 presents a method for kinodynamic path planning using state lattices constructed from a generated gait library. Simulated results on MARLO are given. Finally, Section 7.3 concludes this thesis with future research directions regarding legged robots.

7.1 Including Terrain Information in the Gait Library

Assuming a local terrain map is given (perhaps using the LiDAR mapping results of Chapter 4), we can begin to make progress towards terrain adaptive control. One potential idea is to extend the gait library with information on how to handle this terrain.

7.1.1 Including Future Slope Information

One simple representation of the terrain is ground slope. Da et al. [60] utilized a gait library of forwards and backwards walking, along with transitions between them. The gait library was then utilized to regress a control policy that could stabilize the robot while allowing it to walk forwards and backwards. The resulting control policy took in the current velocity and the desired velocity to determine the desired trajectory; $\pi(v, v^d)$. To include slope, we need to generate a much larger gait library with a 2D grid of periodic gaits (velocity and slope angle), and potentially all transitions between them. The new regressed control policy takes in the next step's slope and previous step's slope as additional inputs; $\pi(v, v^d, s^{prev}, s^{next})$. If all transition gaits are included, even just 5 choices of velocity and slope angle will require solving 625 trajectory optimization problems. However, we did test this idea on MARLO with promising simulation results. We generated a gait library with slopes ranging from -20 deg to 20 deg and velocities ranging from $-0.8\ m/s$ to $0.8\ m/s$. The resulting regressed

control policy allowed the simulated robot to track velocity while walking up and down slopes. This is shown in Figure 7.1. A similar controller was developed by Nguyen et al. [175] where

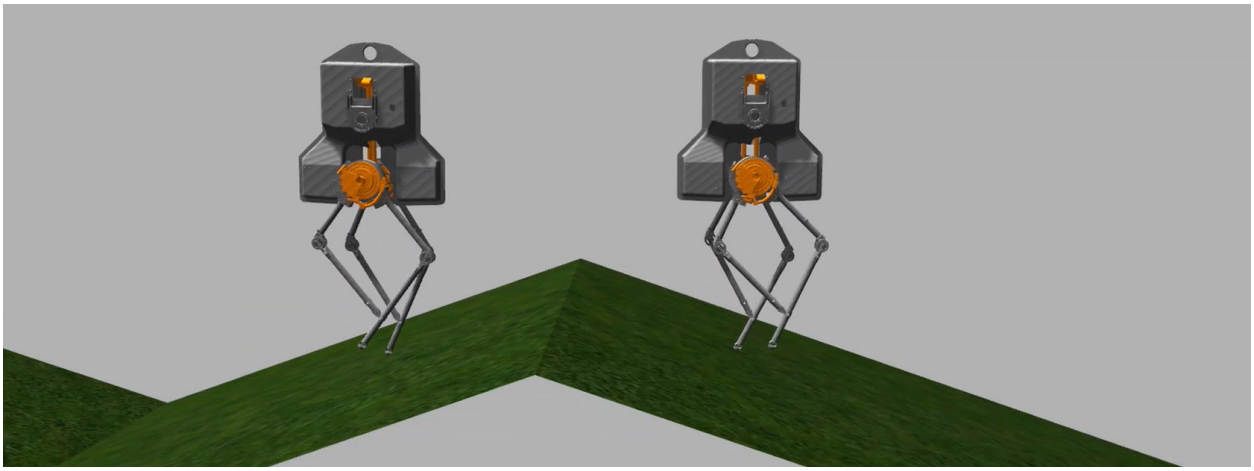


Figure 7.1: Simulated MARLO walking up and down a 20 deg slope.

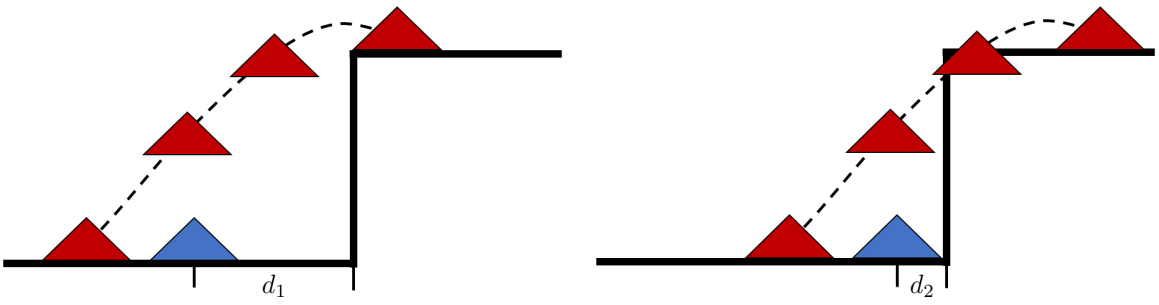
step length and step height were used in place of slope and velocity.

Although these methods may work for certain scenarios, it is not general enough to work over all types of potential terrain. For example, this type of control would not work for stepping over obstacles or walking up staircases where the swing foot might have to take many different trajectories to remain collision free depending on the location of the rise relative to the stance foot.

7.1.2 Stair Climbing

Stair climbing requires precise foot placement in addition to maintaining stability. This applies to both the stance foot (to remain on the stair runs) and the swing foot (to prevent collision with the stair rises). Therefore, step height alone is not enough information in the gait library to. This issue is depicted in Figure 7.2. One option is to consider all possible stance foot locations during the gait library generation. However, this will dramatically increase the number of gaits needed to cover all required behaviors. Another option is to simply place a constraint forcing the swing foot to remain behind the stance foot until the stair rise is cleared. With this constraint enforced, the desired swing foot trajectory will never collide with the staircase regardless of the stance foot position. This idea was tested on Cassie by generating a library of gaits for stepping up a 7 *inch* rise at various forward velocities. Based on the current velocity estimate, a desired step up trajectory was chosen through linear interpolation. Figures 7.3 and 7.4 show the results of this experiment.

Although Cassie was able to step up successfully a few times, we had numerous failures



(a) No swing foot collision

(b) Swing foot collision

Figure 7.2: The trajectory of the swing foot (red) needs to be dependent on the location of the step relative to the stance foot (blue). We have to assume some staircase position for optimization. If this location relative to the stance foot changes, collision may occur.

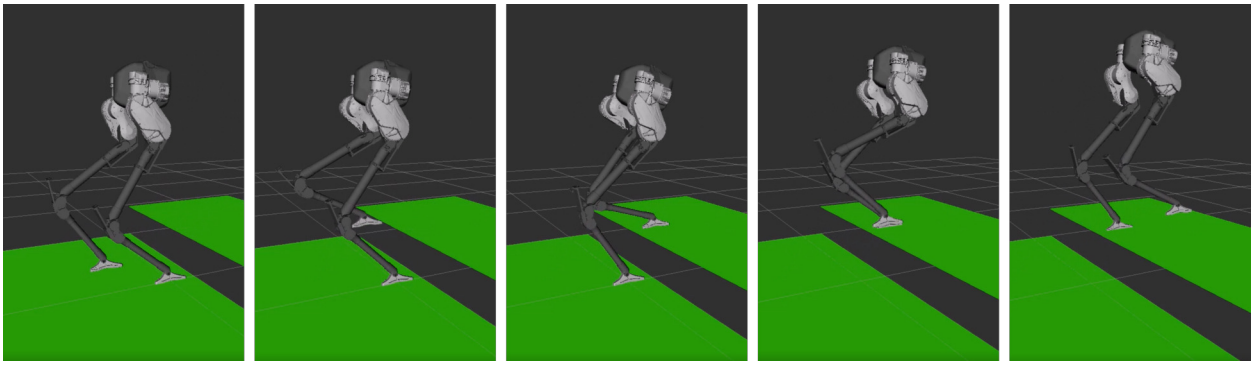


Figure 7.3: Simulated Cassie stepping up a 7 inch rise.

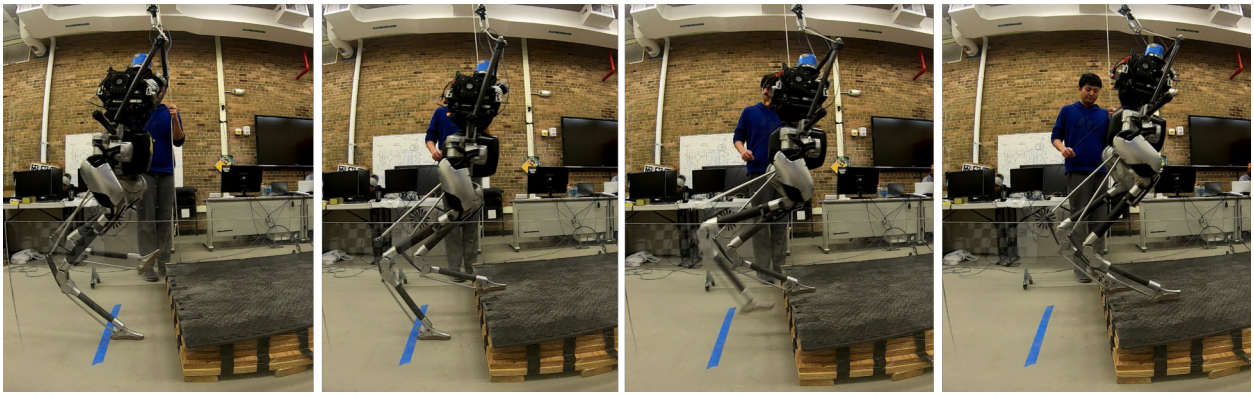


Figure 7.4: Cassie stepping up a 7 inch rise. The standard staircase in the United States has a 7 inch rise and 11 inch run.

due to tracking issues. Even through the desired swing foot trajectory will never collide with the staircase, the realized path still did. In the future, to achieve reliable step up behavior, these collision preventing constraints need to be enforced at runtime. CBF offer one potential method to enforce these constraints [13]. Quan Nguyen et al. [187] used control barrier functions (CBFs) alongside gait libraries to enforce step length constraints allowing

their simulated robot to walk over stepping stones. It is not yet clear how to extend these ideas to stair climbing.

7.2 Global Kinodynamic Planning Using State Lattices

Assuming a global map of the environment exists (perhaps generated using the ideas of Chapter 5), motion planning algorithms need to be developed to compute collision free paths throughout the world. Many general kinodynamic motion planning algorithms exist, such as the popular rapidly-exploring random tree (RRT) [149, 217]. However, it is often not possible to directly apply these algorithms on legged robots due to underactuation, multi-domain dynamics, high degrees of freedom (DOF), and complex constraints (like dynamic stability).

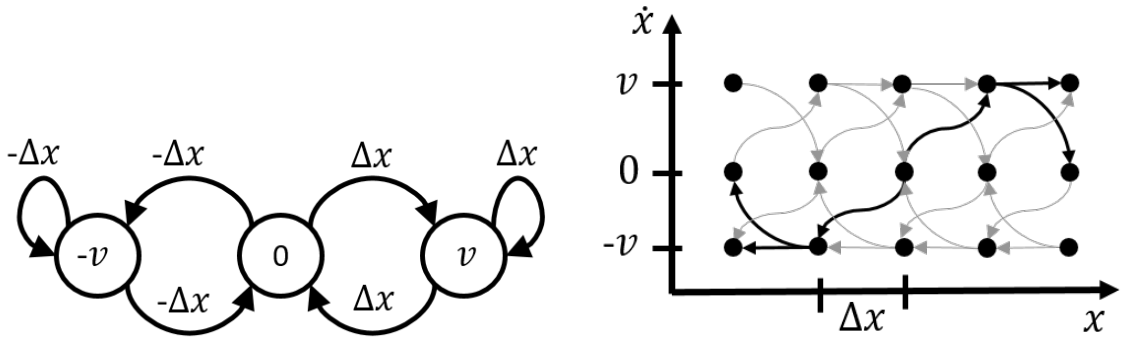
Trajectory optimization does provides a general for computing a dynamically feasible path for the robot. However, current implementations are still too slow to solve at runtime when using the full dynamic model of the robot [118]. Therefore, naively using these trajectory optimizers in a global motion planning algorithm would be impractical. A common approach to alleviate this issue is to build a library of "high-quality" motion primitives offline. Motion planning can then solved though sequential composition of these primitives [87, 163].

This section provides one idea of how gait libraries can be used to enable fast computation of dynamically feasible, obstacle-free paths. First, an offline library of gaits is designed using trajectory optimization. This library includes both periodic walking gaits and transitions between them. This library of gaits will form a set of state lattice motion primitives [180] that are regularly arranged in state space. This enables the high-dimensional motion planning problem to be recast as a low-dimensional graph search problem. Next, a discrete version of Bi-Directional RRT-Connect is used to search over this reduced dimensional space for a sequence of gaits that bring the robot to the goal while avoiding obstacles. A* search is used to smooth out the RRT path using a shortcut smoothing algorithm. When executing the path, A* is also used to find re-plan recovery paths when the robot drifts from the desired path due to controller imperfections or perturbations. These algorithms were used to produce dynamically feasible, obstacle-free paths for an underactuated ATRIAS-series robot with 12-DOF, 6 actuators, and point feet. If each gait is stabilized with an associated feedback controller, a sequence of controllers can be constructed from the path that allows stable, dynamic walking to reach the goal.

7.2.1 Design of Gait Library over State Lattice

To allow for efficient path planning, a library of gaits is computed over a state lattice [180]. Each gait is computed using offline trajectory optimization. This allows the continuous (or hybrid) high-dimensional motion planning problem to be reduced down to a low-dimensional discrete search problem. This dimensionality reduction will allow a feasible solution to be found efficiently, but an optimal solution to the original problem cannot be found. To better understand this idea, consider the following simple example.

Assume we wish to plan paths for a legged robot capable of moving along one dimension. Two periodic gaits have been computed for walking at velocities of $\{-v, v\}$. When executing one step of the gaits, the robot moves a distance of $-\Delta x$ or Δx respectively. In addition, four transition gaits are designed that move the robot from the final state on one periodic gait to the initial state on another periodic gait. These transition gaits also take place over $\{-\Delta x, \Delta x\}$. This library of gaits can be visualized as a connected state transition graph, shown in Figure 7.5a.



(a) Connected graph of dynamically feasible gaits (state lattice primitives).

(b) State lattice primitives (and position shifted versions) arranged on a sampled state space.

Figure 7.5: State lattice primitives for a simple one-dimensional walking robot.

The state space is now reduced from a potentially high-dimensional robot to a two-dimensional (position and velocity) representation. When viewed in this reduced state space, the gaits make up a set of motion primitives that describe how to transition from one state to another. In addition, because these gaits are position-invariant, the motion primitives can be utilized over any position in the search space. The discrete set of states that the robot can transition to defines the state lattice over which the gait library is designed. Figure 7.5b shows these primitives arranged over the state space.

7.2.2 Using Gait Libraries for Discrete Path Planning

Once the gait library (of state lattice primitives) is generated, the high-dimensional motion planning problem is simplified to searching for sequence of primitives that bring the robot from x_{init} to x_{goal}. Since the gaits were generated over a state lattice, discrete search methods, such as A*, can be employed. In addition, if the initial and goal position are on the lattice, the discrete search is not an approximation and an exact path can be found.

7.2.2.1 Discrete Bi-Directional RRT-Connect

In practice, A* tends to be slow if there is a high branching factor or if there are many obstacles blocking the path from x_{init} to x_{goal}. This led to the use of an Bi-Directional RRT algorithm to speed up path planning [149]. However, when growing the tree, only a discrete set of transitions can be considered. In addition, the goal tree has to be grown backwards in time in order for the gait transitions and dynamics to make sense. These small modification allows Bi-Directional algorithm to be used for discrete state lattice planning. Collision checking has to be performed over the entire gait, not just at the discrete nodes. For efficiency, swept bounding boxes of each gait in the library can be precomputed. Collision checking involves simply checking if this swept volume is in collision with any obstacle.

7.2.2.2 Pruning Tree to Accommodate Terrain Information

Uneven terrain can easily be accounted for through a simple tree pruning modification. Each gait in the library has a specific footfall pattern. For a robot to use a particular gait, the upcoming terrain height must match that gait's footfall pattern. If this pattern does not match the terrain, this node is not considered a valid successor and is pruned away inside the RRT's *steer* function. This can be thought of as a "reverse collision check", where the gait is checked to make sure the feet collide with the ground at the appropriate times.

7.2.2.3 A* Shortcut Smoothing

Standard RRT algorithms do not produce optimal paths, and often contain unnecessary loops and detours. It is necessary to smooth out the RRT solution path before having a controller execute it. One way to perform path smoothing is through shortcut smoothing algorithms. Typically this is done by iteratively selecting two random nodes along the RRT path and trying to connect them with a straight line path. If this path is collision free, then it replaces the RRT path between the two nodes.

Since legged locomotion is a nonholonomic problem and the gaits are defined along a state lattice, straight line paths cannot be used. Instead, A* search can be used to connect

the randomly selected nodes. To make the smoothing process fast, a max iterations of each A* search is limited to a small number. The cost function used for A* can either be the total time or energy expenditure. This algorithm is outlined below.

Algorithm 2: A* Shortcut Smoothing

```
1: function SMOOTH(path, K)
2:     pathSmooth ← path
3:     for k = 1 to K do
4:         (x_1, x_2) ← SelectRandomNodes()
5:         shortcut = AStar(x_1, x_2, maxIterations)
6:         if shortcutPath ≠ ∅ then
7:             pathSmooth ← ReplacePath(x_1, x_2, shortcut)
```

7.2.2.4 A* Re-planning

The result from discrete path planning with a gait library is a sequence of gaits that when concatenated together bring the robot from x_{init} to x_{goal}. If each gait in the library has an associated stabilizing controller, then these controllers can simply be concatenated together to yield a feedback controller that will drive the robot to the goal. However, while executing this path, inaccuracies in the robot model, imperfections the feedback controller, or external disturbances will inevitable cause the robot to diverge from the planned trajectory. Therefore, it is necessary to automatically re-plan a path so the robot does not collide with obstacles.

When the robot's deviation from the path greater than some threshold, an online A* re-planner is triggered. This A* search will attempt to connect the current robot's state a nearby point on the original trajectory using only gaits defined in the library. If a solution is found, the controller gets updated and the robot gets steered back to the original path. Since re-planning threshold is usually small (discretization of state lattice), the robot's deviation from the original path is likely to be small. Therefore, A* tends to find a path very quickly.

Algorithm 3: A* Re-Planner

```
1: function REPLAN(path, x)
2:     x_init ← FindClosestPointOnLattice()
3:     x_goal ← FindNearbyPointOnPath(path)
4:     recoveryPath = AStar(x_init, x_goal, maxIterations)
5:     if recoveryPath ≠ ∅ then
6:         newPath ← AppendToPath(recoveryPath)
```

Number	Description	Link
1	Maze Navigation from Discrete Bi-Directional RRT-Connect Path	`https://youtu.be/mvAXEJ7hY88`
2	Maze A* Re-Planning from External Disturbance	`https://youtu.be/Rm_vPjSsvnY`
3	Staircase Navigation from A* Computed Path	`https://youtu.be/rKvSzjuaYqU`

Table 7.1: Simulation Video Links for State Lattice Motion Planning

7.2.3 Simulation Results

The presented algorithms were tested on two different environments (maze and steps) with distinct gait libraries. All algorithms and dynamic simulations were run using *MATLAB* on an ATRIAS-series robot. Refer to Section 3.1.2 for more details about the robot. Table 7.1 gives video links to example simulations.

7.2.3.1 Maze Navigation

A flat-ground, "maze-like" environment was used to demonstrate the presented algorithms. This environment is shown in Figures 7.6 and 7.7. Eleven periodic walking gaits were designed using direct-collocation trajectory optimization. Each of these gaits were 2-step periodic and had distinct velocities for walking forwards, backwards, sideways, and diagonally. In addition, 40 asymmetric gaits were optimized that transition (over 4 steps) between a subset of these periodic gaits. All 51 gaits were designed to have the robot's position remain on a discretized grid of 0.2 m in the x direction and 0.25 m in the y direction.

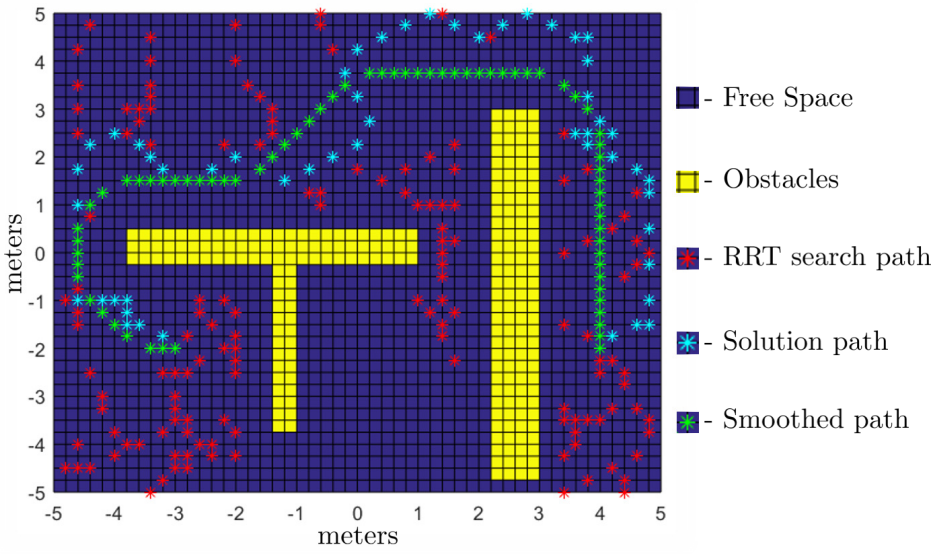


Figure 7.6: Typical Discrete Bi-Directional RRT-Connect solution for MARLO in the maze environment.

Figure 7.6 shows an example solution path. The red marks show the positions that

	Iterations	Time Elapsed (s)	Path Length (steps)
RRT Search	650.3	8.796	237.4
A* Smoothing		183.412	153.2

Table 7.2: Results from 10 runs in the maze environment. The start and goal states remained the same over all runs.

the Discrete Bi-Directional RRT-Connect algorithm explored. The resulting path, shown in blue, is unnecessarily long winding. The smoothed green path shows the result of applying the A* shortcut smoothing algorithm. Both paths successfully avoid all obstacles, shown in yellow.

To evaluate the speed and effectiveness of the search algorithm, Discrete Bi-Directional RRT-Connect and A* Shortcut Smoothing were run 10 times on with identical start and goal states. Over the 10 trials, the average time elapsed for the RRT search was 8.796 seconds. The average number of iterations was 650.3 and the average path length of 237.4 steps. A* Shortcut Smoothing was implemented for 200 iterations, with each A* search timing out after 200 iterations. With these parameter values, path smoothing took significantly longer than path searching. The average smoothing time was 183.41 seconds, resulting in a an average path length of 153.2 steps. These results are summarized in Table 7.2.

The A* re-planner was also tested by simulating a force perturbation on the robot. This perturbation was enough to push the robot away from the solution path, triggering the re-planner. The robot was able to quickly re-plan a recovery path back to the original one. A video link of this simulation is given in Table 7.1. A screen-shot from this simulation is shown in Figure 7.7.

7.2.3.2 Staircase Navigation

To test the terrain tree pruning, a staircase environment was simulated with the ground height changing in discrete values of 5 cm, 10 cm, and 15 cm. To allow walking in this environment, 13 two-step periodic gaits were designed. These gaits include walking forward at 3 different speeds, and walking up and downhill. In addition, 24 transition gaits were designed that transition between a subset of the periodic gaits. Figure 7.8 shows the connected state transition graph for all 37 gaits.

For this environment, the staircases represent the only obstacles. The pruning step checks the upcoming ground against the potential successor gait to make sure the footfall pattern is compatible with the terrain. For this simulation, simply using A* to search over the state lattice was enough to get a solution quickly (3.768 seconds). A video link of this simulation is given in Table 7.1. A screen-shot from this simulation is shown in Figure 7.9.

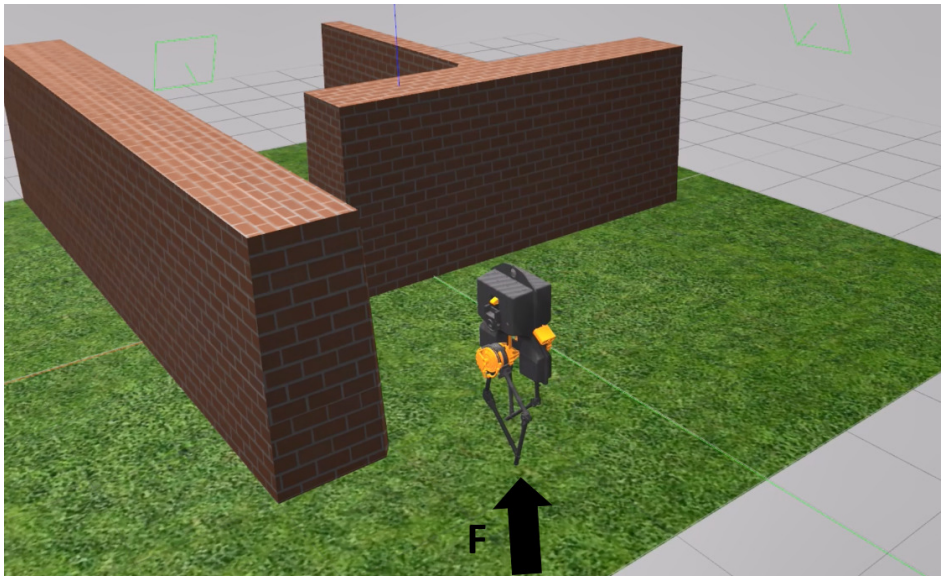


Figure 7.7: When the robot drifts or is perturbed from the solution path, A* is used to quickly plan a recovery path back to the original path.

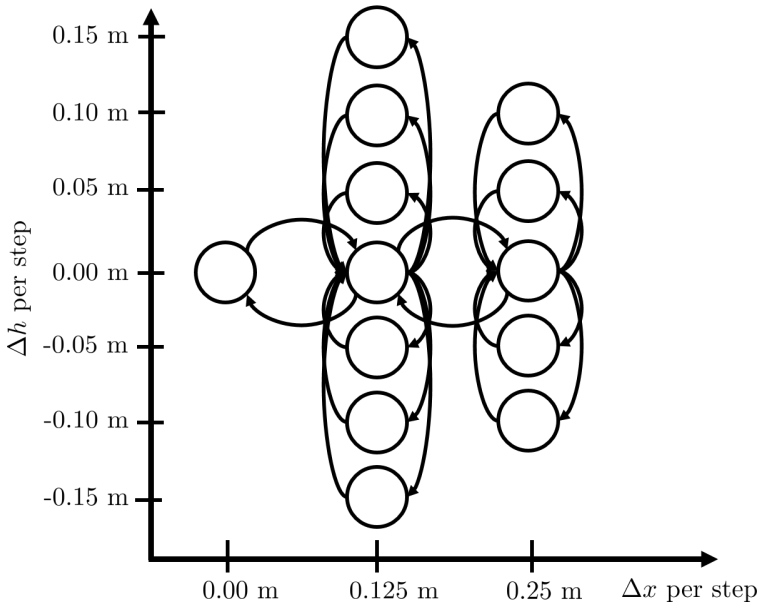


Figure 7.8: State transition graph for walking up and down stairs. The circles represent periodic two-step gaits. The arrows represent two step transition gaits that move between the corresponding periodic orbits.

7.2.4 Discussion

A state lattice motion planning algorithm was presented to allow kinodynamic path planning for high-dimensional underactuated legged robots. First, a library of stabilized gaits was generated using direct-collocation trajectory optimization. These gaits include periodic walking and aperiodic transition gaits that move between the periodic ones. Every gait sat-

isfies key constraints such as dynamics, joint limits, torque limits, and friction cones. These gaits are designed in such a way to have the robot's position and velocity always falling on a discrete set of values. This reduces the high-dimensional motion planning problem to a low-dimensional discrete search over a state lattice.

Once the gait library is designed, many different search algorithms can be applied to find the sequence of gaits (and controllers) that brings the robot from x_{init} to x_{goal}. A Discrete Bi-Directional RRT-Connect algorithm was developed to solve this search problem efficiently. During the search, a swept bounding box of the robot over each gait was used for collision checking. A tree pruning step was implemented that trims away successor gaits that have footfall patterns that do not match the upcoming terrain. A* was used to both smooth out the RRT path (using shortcut smoothing) and to re-plan when the robot drifts or is perturbed from the executed path.

Using these algorithms, the final feedback controller that drives the robot from x_{init} to x_{goal} is switching controller constructed from a particular finite sequence of controllers in the gait library. This may work well for some applications and environments, however, the discrete nature of the gait library brings some disadvantages. For example, if gaits are designed to walk up steps of 10 cm and 15 cm, no path would be found if a step of 12cm comes up. It is possible to discretize the environment to match the available gaits, but executed controller may perform poorly. If a finer discretization of gaits is used, the time it takes to build the library and find solution paths grows larger.

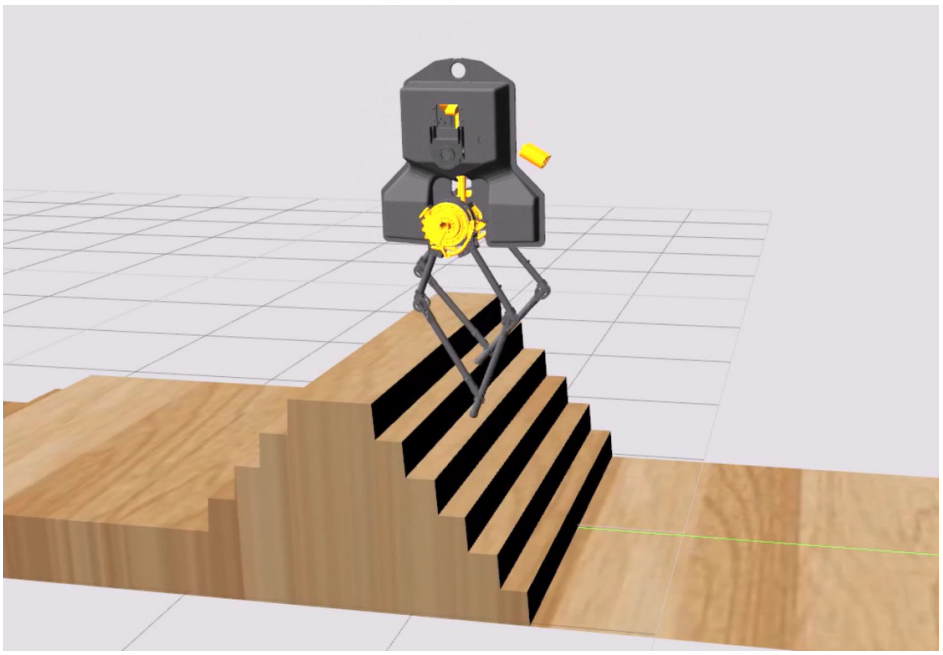


Figure 7.9: MARLO navigating down a set of stairs.

Gait libraries have found success in controller design by allowing a continuous controller to be regressed from a discrete set of gaits [60, 187]. For example, if gaits are designed to walk up steps of 10cm and 15cm, a good controller approximation for a 12 cm step could be a linear interpolation of the 10 cm and 15 cm gaits. It may be possible to perform path planning over this continuous set of controllers, allowing solutions to be found easier for more complex terrain and environments.

None of the algorithms in this optimized for efficiency, so it is unclear how fast this motion planner can be practically implemented. In addition, we assumed perfect knowledge of the global map and our position within it. More research needs to be done on motion planning algorithms that utilize the uncertainty of state information to plan conservatively. This will be import when testing these ideas experimentally.

7.3 Conclusion and Future Directions

This thesis presented several methods for contact-aided state estimation on legged robots. Both the invariant extended Kalman filter (InEKF) of Chapter 4 and the developed factors of Chapter 5 utilize Lie group theory to improve upon existing state-of-the-art methods. In particular, these state estimators assumed only inertial, contact, and encoder measurements, all of which are readily available on most legged robots. Therefore, the developed estimators have the potential to become an essential part of these platforms going forward. However, even with these improvements, legged robots are still far from being ubiquitous in our everyday lives. We conclude with several ideas on future research directions that can bring us closer to that reality.

Having a low-drift odometry system is key to realizing practical simultaneous localization and mapping (SLAM) for any mobile robot. The developed InEKF moves towards this goal by combining kinematic and contact data to reduce drift from the inertial measurement unit (IMU)-based dynamics. A key idea behind this filter is the notion of an *inferred measurement* informing the system. We have no sensor measuring the velocity of the stance foot, but when we measure contact, we can infer that the stance foot velocity is zero. I believe it may be possible to reduce drift even further by utilizing additional inferred measurements. For example, if the controller is in standing mode, perhaps we can infer that the base velocity is zero. If the robot is walking on a sidewalk, maybe we can infer that the ground height remains constant. As shown by Brossard et al. [44], the detection of these mode changes could reduce drift and improve the state estimation results.

In Chapter 5 we developed a method for incorporating "leg odometry" into the factor graph framework. Combining the proposed factors with vision-based factors and loop clo-

sures can potentially provide one solution for legged robot SLAM. While walking, the map can be constructed while the robot's pose is being estimated. Loop closures serve to correct the odometry drift. However, with just these measurements, we are missing a key measurement that I believe will further improve performance. When the robot measures contact, not only can we assume the foot velocity is zero, but we can assume that the robot is physically touching a part of the map that is being built. This type of "terrain-based loop closure" (briefly hinted at Section 5.4) literally grounds the robot to the map. Without this, the robot could be estimated to be walking above or below the ground, which is of course not physically possible. This inconsistency could potentially create issues when using the estimate/map for motion planning and control. One future research direction is to explore formulating terrain reconstruction and SLAM as a unified problem. These types of measurements could also be used for pure localization (imagine a blind robot navigating in dark room with a prior map).

One of the main sources of position/yaw drift for both contact-aided filters and smoothers comes from inaccurate kinematics. Both estimators assume that the kinematic equations for the robot are perfectly known, and the only uncertainty in forward kinematics comes from zero-mean Gaussian encoder noise. In reality this is not the case. For improved performance, uncertainties in the model parameters (leg lengths, joint offsets, etc.) should be accounted for. Unfortunately, we typically do not know the true mean values for these parameters, resulting in kinematics functions that are biased. If unaccounted for, these forward kinematic biases can lead to significant drift for inertial-contact odometry. Similar to how IMU biases are estimated, it may be possible to use factor graphs to estimate these kinematic parameters online. This self-calibrating process could be important when legged robots are mass-produced and used long-term.

To build truly autonomous legged robots we need to solve complex state estimation, motion planning and control problems. Often, these three problems are imagined to be completely independent: planning a path under the assumption that the environment is completely known; developing a feedback controller with the assumption that the state is perfectly known; or ignoring the potential to improve state estimates through clever choices in motion plans or control actions. In reality these problems are not separated. For legged robots, motion planning should be done with knowledge of the set of gaits the controller can stabilize. Humanoids should plan to grab the hand rail if they know the uncertainty in relative staircase position can be reduced or if the desired motion becomes easier to track. Uncertainty in the robot's pose and map estimate should be used when planning to reduce the chance of unwanted collisions. The list could go on. Thankfully, the field of robotics has been moving towards these goals, but as always there are still countless research opportunities for the future.

BIBLIOGRAPHY

[1] P-A Absil, Christopher G Baker, and Kyle A Gallivan. Trust-region methods on riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007.

[2] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[3] Nasradine Aghannan and Pierre Rouchon. On invariant asymptotic observers. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 2, pages 1479–1484. IEEE, 2002. ISBN 0780375165.

[4] Ayush Agrawal, Omar Harib, Ayonga Hereid, Sylvain Finet, Matthieu Masselin, Laurent Praly, Aaron D. Ames, Koushil Sreenath, and J. W. Grizzle. First steps towards translating hzd control of bipedal robots to decentralized control of exoskeletons. Preprint, 2017.

[5] 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. ISSN 2169-3536.

[6] Kaveh Akbari Hamed, Brian G Buss, and Jessy W Grizzle. Continuous-time controllers for stabilizing periodic orbits of hybrid systems: Application to an underactuated 3D bipedal robot. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 1507–1513. IEEE, 2014. ISBN 1467360902.

[7] Aaron D Ames. First steps toward automatically generating bipedal robotic walking from human data. In *Robot Motion and Control 2011*, pages 89–116. Springer, 2012.

[8] Aaron D Ames. Human-inspired control of bipedal walking robots. *IEEE Transactions on Automatic Control*, 59(5):1115–1130, 2014. ISSN 0018-9286.

[9] Aaron D Ames and Matthew Powell. Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs. In *Control of Cyber-Physical Systems*, pages 219–240. Springer, 2013.

[10] Aaron D Ames, Kevin Galloway, and Jessy W Grizzle. Control lyapunov functions and hybrid zero dynamics. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 6837–6842. IEEE, 2012.

[11] Aaron D. Ames, Kevin Galloway, Koushil Sreenath, and Jessy W. Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, 59(4):876–891, 2014. ISSN 00189286. doi: 10.1109/TAC.2014.2299335.

[12] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 6271–6278. IEEE, 2014. ISBN 1467360902.

[13] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017. ISSN 0018-9286.

[14] Brian D O Anderson and John B Moore. Optimal filtering. *Englewood Cliffs*, 21:22–95, 1979.

[15] Anon. Logistical Vehicle Off-Road Mobility. Technical report, Project TCCO 62-5; U. S. Army Transportation Combat Developments Agency, Fort Eustis, Va, 1967.

[16] Zvi Artstein. Stabilization with relaxed controls. *Nonlinear Analysis: Theory, Methods & Applications*, 7(11):1163–1173, 1983. ISSN 0362-546X.

[17] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

[18] Martin Barczyk and Alan F Lynch. Invariant extended Kalman filter design for a magnetometer-plus-GPS aided inertial navigation system. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 5389–5394. IEEE, 2011. ISBN 161284801X.

[19] Martin Barczyk and Alan F Lynch. Invariant observer design for a helicopter UAV aided inertial navigation system. *IEEE Transactions on Control Systems Technology*, 21(3):791–806, 2013. ISSN 1063-6536.

[20] John Bares, Martial Hebert, Takeo Kanade, Eric Krotkov, Tom Mitchell, Reid Simmons, and William Whittaker. Ambler: An autonomous rover for planetary exploration. *Computer*, 22(6):18–26, 1989. ISSN 0018-9162.

[21] Timothy D Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

[22] Timothy D Barfoot and Paul T Furgale. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*, 30(3):679–693, 2014.

[23] Axel Barrau. Non-linear state error based extended Kalman filters with applications to navigation, 2015.

[24] Axel Barrau and Silvère Bonnabel. The invariant extended Kalman filter as a stable observer. *IEEE Transactions on Automatic Control*, 62(4):1797–1812, 2017. ISSN 0018-9286.

[25] Axel Barrau and Silvère Bonnabel. Invariant Kalman Filtering. *Annual Review of Control, Robotics, and Autonomous Systems*, (0), 2018. ISSN 2573-5144.

[26] Billur Barshan and Hugh F Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, 1995. ISSN 1042-296X.

[27] Mehdi Benallegue and Florent Lamiraux. Estimation and stabilization of humanoid flexibility deformation using only inertial measurement units and contact information. *International Journal of Humanoid Robotics*, 12(03):1550025, 2015.

[28] Dmitry Berenson, Rosen Diankov, Koichi Nishiwaki, Satoshi Kagami, and James Kuffner. Grasp planning in complex scenes. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 42–48. IEEE, 2007. ISBN 1424418615.

[29] Paul J Besl and Neil D McKay. Method for registration of 3-D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[30] J T Betts and I Kolmanovsky. Practical methods for optimal control using nonlinear programming. *Applied Mechanics Reviews*, 55:B68, 2002.

[31] Sanjay P Bhat and Dennis S Bernstein. Continuous finite-time stabilization of the translational and rotational double integrators. *IEEE Transactions on automatic control*, 43(5):678–682, 1998. ISSN 0018-9286.

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

[33] Michael Bloesch. State Estimation for Legged Robots-Kinematics, Inertial Sensing, and Computer Vision, 2017.

[34] Michael Bloesch, Marco Hutter, Mark A Hoepflinger, Stefan Leutenegger, Christian Gehring, C David Remy, and Roland Siegwart. State Estimation for Legged Robots - Consistent Fusion of Leg Kinematics and IMU. *Proceedings of Robotics: Science and Systems*, page 2005, 2012. ISSN 2330765X. doi: 10.15607/RSS.2012.VIII.003.

[35] Michael Bloesch, Christian Gehring, Peter Fankhauser, Marco Hutter, Mark A. Hoepflinger, and Roland Siegwart. State estimation for legged robots on unstable and slippery terrain. In *IEEE International Conference on Intelligent Robots and Systems*, pages 6058–6064, 2013. ISBN 9781467363587. doi: 10.1109/IROS.2013.6697236.

[36] J-D Boissonnat, Olivier Devillers, Leonbattista Donati, and Franco P Preparata. Motion planning for spider robots. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2321–2326. IEEE, 1992. ISBN 0818627204.

[37] Silvere Bonnabel. Left-invariant extended Kalman filter and attitude estimation. In *Decision and Control, 2007 46th IEEE Conference on*, pages 1027–1032. IEEE, 2007. ISBN 1424414970.

[38] Silvere Bonnabel, Philippe Martin, and Pierre Rouchon. Non-linear symmetry-preserving observers on Lie groups. *IEEE Transactions on Automatic Control*, 54 (7):1709–1713, 2009. ISSN 0018-9286.

[39] Silvère Bonnabel, Philippe Martin, and Erwan Salaün. Invariant Extended Kalman Filter: theory and application to a velocity-aided attitude estimation problem. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 1297–1304. IEEE, 2009. ISBN 1424438713.

[40] Guillaume Bourmaud, Rémi Mégret, Audrey Giremus, and Yannick Berthoumieu. Discrete extended Kalman filter on Lie groups. In *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*, pages 1–5. IEEE, 2013. ISBN 0992862604.

[41] Guillaume Bourmaud, Rémi Mégret, Marc Arnaudon, and Audrey Giremus. Continuous-discrete extended Kalman filter on matrix Lie groups using concentrated Gaussian distributions. *Journal of Mathematical Imaging and Vision*, 51(1):209–228, 2015. ISSN 0924-9907.

[42] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. ISBN 1107394007.

[43] Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61, 2010. ISSN 0020-3157.

[44] Martin Brossard, Axel Barrau, and Silvere Bonnabel. Rins-w: Robust inertial navigation system on wheels. *arXiv preprint arXiv:1903.02210*, 2019.

[45] Brian G. Buss, Alireza Ramezani, Kaveh Akbari Hamed, Brent A. Griffin, Kevin S. Galloway, and Jessy W. Grizzle. Preliminary walking experiments with underactuated 3D bipedal robot MARLO. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2529–2536, 2014. ISBN 9781479969340. doi: 10.1109/IROS.2014.6942907.

[46] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. ISSN 1552-3098.

[47] Luca Carlone, Zsolt Kira, Chris Beall, Vadim Indelman, and Frank Dellaert. Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4290–4297. IEEE, 2014. ISBN 1479936855.

[48] Paul Chauchat, Axel Barrau, and Silvere Bonnabel. Invariant Smoothing on Lie Groups. *arXiv preprint arXiv:1803.02076*, 2018.

[49] Joel Chestnutt. *Navigation planning for legged robots*. PhD thesis, Citeseer, 2007.

[50] Joel Chestnutt, Manfred Lau, German Cheung, James Kuffner, Jessica Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 629–634. IEEE, 2005. ISBN 078038914X.

[51] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E.R. Westervelt, C. Canudas-De-Wit, and J.W. Grizzle. RABBIT: a testbed for advanced control theory. *Control Systems, IEEE*, 23(5):1–51, 2003. ISSN 1066-033X. doi: 10.1109/MCS.2003.1234651.

[52] Christine Chevallereau, J. 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. ISSN 15523098. doi: 10.1109/TRO.2008.2010366.

[53] Gregory S Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*, volume 2. Springer Science & Business Media, 2011. ISBN 0817649433.

[54] Sachin Chitta, Paul Vemaza, Roman Geykhman, and Daniel D Lee. Proprioceptive localilzatilon for a quadrupedal robot on known terrain. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4582–4587. IEEE, 2007. ISBN 1424406021.

[55] Catherine Clifford. This start-up is building a humanoid robot that could soon be delivering packages to your door. May 2018. [Online; Published Tue, May 1 2018 1:05 PM EDT].

[56] Jose A Cobano, Joaquin Estremera, and P Gonzalez De Santos. Location of legged robots in outdoor environments. *Robotics and Autonomous Systems*, 56(9):751–761, 2008. ISSN 0921-8890.

[57] John L Crassidis. Sigma-point Kalman filtering for integrated GPS and inertial navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 42(2):750–756, 2006. ISSN 0018-9251.

[58] 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. *arXiv preprint arXiv:1711.02223*, 2017.

[59] 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. ISSN 2169-3536.

[60] Xingye Da, Ross Hartley, and Jessy W Grizzle. Supervised learning for stabilizing underactuated bipedal robot locomotion, with outdoor experiments on the wave field. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3476–3483. IEEE, 2017. ISBN 150904633X.

[61] Hongkai Dai and Russ Tedrake. Optimizing robust limit cycles for legged locomotion on unknown terrain. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 1207–1213. IEEE, 2012.

[62] Anirvan Dasgupta and Yoshihiko Nakamura. Making feasible walking motion of humanoid robots from human motion capture data. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1044–1049. IEEE, 1999. ISBN 0780351800.

[63] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 279–286. IEEE, 2014. ISBN 147997174X.

[64] M H P Dekker. Zero-moment point method for stable biped walking. *Eindhoven University of Technology*, 2009.

[65] Frank Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical report, 2012.

[66] Frank Dellaert and Michael Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. In *International Journal of Robotics Research*, volume 25, pages 1181–1203, 2006. ISBN 0278364906072. doi: 10.1177/0278364906072768.

[67] Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[68] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16. Siam, 1996. ISBN 1611971209.

[69] M W M Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001. ISSN 1042-296X.

[70] Tue-Cuong Dong-Si and Anastasios I Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5655–5662. IEEE, 2011. ISBN 1612843859.

[71] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[72] Eric R Dunn and Robert D Howe. Foot placement and velocity control in smooth bipedal walking. *Proceedings of the 1996 Ieee International Conference on Robotics and Automation*, pages 578–583 vol. 1, 1996. ISSN 10504729. doi: 10.1109/ROBOT.1996.503837.

[73] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part I. *IEEE robotics & automation magazine*, 13(2):99–110, 2006. ISSN 1070-9932.

[74] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. Closed-form preintegration methods for graph-based visual–inertial navigation. *The International Journal of Robotics Research*, page 0278364919835021, 2018.

[75] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. Continuous preintegration theory for graph-based visual-inertial navigation. *arXiv preprint arXiv:1805.02774*, 2018.

[76] Craig Eldershaw and Mark Yim. Motion planning of legged vehicles in an unstructured environment. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 3383–3389. IEEE, 2001. ISBN 0780365763.

[77] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. ISSN 0018-9162.

[78] Jorhabib Eljaik, Naveen Kuppuswamy, and Francesco Nori. Multimodal sensor fusion for foot state estimation in bipedal robots using the extended kalman filter. pages 2698–2704. IEEE, 2015.

[79] Ryan Eustice, Matthew Walter, and John Leonard. Sparse extended information filters: Insights into sparsification. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3281–3288. IEEE, 2005. ISBN 0780389123.

[80] Ryan M Eustice, Hanumant Singh, and John J Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114, 2006. ISSN 1552-3098.

[81] Maurice F Fallon, Matthew Antone, Nicholas Roy, and Seth Teller. Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 112–119. IEEE, 2014. ISBN 147997174X.

[82] Péter Fankhauser, Michael Bloesch, Christian Gehring, Marco Hutter, and Roland Siegwart. Robot-centric elevation mapping with uncertainty estimates. In *Mobile Service Robotics*, pages 433–440. World Scientific, 2014.

[83] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. 33(1):1–21, 2017.

[84] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Supplementary material to: Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Technical report, Georgia Institute of Technology, 2015.

[85] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-Manifold Preintegration for Real-Time Visual–Inertial Odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017. ISSN 1552-3098.

[86] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017. ISSN 1552-3098.

[87] Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, 2005. ISSN 15523098. doi: 10.1109/TRO.2005.852260.

[88] Randy Freeman and Petar V Kokotovic. *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science & Business Media, 2008. ISBN 0817647589.

[89] Kevin Galloway, Koushil Sreenath, Aaron D Ames, and J W Grizzle. Torque Saturation in Bipedal Robotic Walking through Control Lyapunov Function Based Quadratic Programs.

[90] Demoz Gebre-Egziabher, Roger C Hayward, and J David Powell. Design of multi-sensor attitude determination systems. *IEEE Transactions on aerospace and electronic systems*, 40(2):627–649, 2004. ISSN 0018-9251.

[91] 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. *arXiv preprint arXiv:1809.07279*, 2018.

[92] Ambarish Goswami. Postural Stability of Biped Robots and the Foot-Rotation Indicator (FRI) Point. *The International Journal of Robotics Research*, 18(6):523–533, 1999. ISSN 0278-3649. doi: 10.1177/02783649922066376.

[93] Robert D Gregg, Tommaso Lenzi, Levi J Hargrove, and Jonathon W Sensinger. Virtual constraint control of a powered prosthetic leg: From simulation to experiments with transfemoral amputees. *IEEE Transactions on Robotics*, 30(6):1455–1471, 2014. ISSN 1552-3098.

[94] Mohinder S Grewal. Kalman filtering. In *International Encyclopedia of Statistical Science*, pages 705–708. Springer, 2011.

[95] Brent Griffin and Jessy Grizzle. Nonholonomic Virtual Constraints for Dynamic Walking. 2015.

[96] Brent Griffin and Jessy Grizzle. Nonholonomic virtual constraints for dynamic walking. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 4053–4060. IEEE, 2015. ISBN 1479978868.

[97] Brent Griffin and Jessy Grizzle. Walking gait optimization for accommodation of unknown terrain height variations. In *Proceedings of the American Control Conference*, volume 2015-July, pages 4810–4817, 2015. ISBN 9781479986842. doi: 10.1109/ACC.2015.7172087.

[98] Jesse A Grimes and Jonathan W Hurst. The design of ATRIAS 1.0 a unique monopod, hopping robot. In *Adaptive Mobile Robotics*, pages 548–554. World Scientific, 2012.

[99] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010. ISSN 1939-1390.

[100] J W Grizzle, Christine Chevallereau, Ryan W Sinnet, and Aaron D Ames. Models, Feedback Control, and Open Problems of 3D Bipedal Robotic Walking.

[101] J. W. Grizzle, Jonathan Hurst, Benjamin Morris, Hae Won Park, and Koushil Sreenath. MABEL, a new robotic bipedal walker and runner. In *Proceedings of the American Control Conference*, pages 2030–2036, 2009. ISBN 9781424445240. doi: 10.1109/ACC.2009.5160550.

[102] Jessy W Grizzle, Gabriel Abba, and Franck Plestan. Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. *IEEE Transactions on automatic control*, 46(1):51–64, 2001. ISSN 0018-9286.

[103] JW Grizzle and Y Song. The extended Kalman filter as a local asymptotic observer for nonlinear discrete-time systems. *Journal of Mathematical Systems, Estimation and Control*, 5(1):59–78, 1995.

[104] Brian Hall. *Lie groups, Lie algebras, and representations: an elementary introduction*, volume 222. Springer, 2015. ISBN 3319134671.

[105] Charles R Hargraves and Stephen W Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10 (4):338–342, 1987.

[106] Omar Harib, Ayonga Hereid, Ayush Agrawal, Thomas Gurriet, Sylvain Finet, Guilhem Boeris, Alexis Duburcq, M Eva Mungai, Matthieu Masselin, and Aaron D Ames. Feedback Control of an Exoskeleton for Paraplegics: Toward Robustly Stable Hands-free Dynamic Walking. *arXiv preprint arXiv:1802.08322*, 2018.

[107] 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. ISBN 1509021825.

[108] Ross Hartley, Maani Ghaffari Jadidi, Lu Gan, Jiunn-Kai Huang, Jessy W Grizzle, and Ryan M Eustice. Hybrid contact preintegration for visual-inertial-contact state estimation using factor graphs. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3783–3790. IEEE, 2018.

[109] Ross Hartley, Maani Ghaffari Jadidi, Jessy W Grizzle, and Ryan M Eustice. Contact-Aided Invariant Extended Kalman Filtering for Legged Robot State Estimation. In *Proceedings of Robotics: Science and Systems*, 2018.

[110] Ross Hartley, Josh Mangelson, Lu Gan, Maani Ghaffari Jadidi, Jeffrey M Walls, Ryan M Eustice, and Jessy W Grizzle. Legged robot state-estimation through combined forward kinematic and preintegrated contact factors. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[111] Kris Hauser, Timothy Bretl, and J-C Latombe. Non-gaited humanoid locomotion planning. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 7–12. IEEE, 2005. ISBN 0780393201.

[112] Kris Hauser, Timothy Bretl, Kensuke Harada, and Jean-Claude Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Algorithmic foundation of robotics VII*, pages 507–522. Springer, 2008.

[113] Kris Hauser, Timothy Bretl, Jean-Claude Latombe, Kensuke Harada, and Brian Wilcox. Motion planning for legged robots on varied terrain. *The International Journal of Robotics Research*, 27(11-12):1325–1349, 2008. ISSN 0278-3649.

[114] Kris Hauser, Timothy Bretl, Jean-Claude Latombe, and Brian Wilcox. Motion planning for a six-legged lunar robot. In *Algorithmic Foundation of Robotics VII*, pages 301–316. Springer, 2008.

[115] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012. ISSN 0278-3649.

[116] Ayonga Hereid and Aaron D Ames. FROST*: Fast Robot Optimization and Simulation Toolkit. 2017. ISSN 1538626837.

[117] Ayonga Hereid, Eric A Cousineau, Christian M Hubicki, and Aaron D Ames. 3D Dynamic Walking with Underactuated Humanoid Robots: A Direct Collocation Framework for Optimizing Hybrid Zero Dynamics. In *IEEE International Conference on Robotics and Automation*, 2016.

[118] 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.

[119] Kazuo Hirai, Masato Hirose, Yuji Haikawa, and Toru Takenaka. The development of Honda humanoid robot. *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, 2(May):1321–1326 vol.2, 1998. ISSN 1050-4729. doi: 10.1109/ROBOT.1998.677288.

[120] Masato Hirose and Kenichi Ogawa. Honda humanoid robots development. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 365(1850):11–19, 2007. ISSN 1364-503X.

[121] Shigeo Hirose and Osamu Kunieda. Generalized standard foot trajectory for a quadruped walking vehicle. *The International Journal of Robotics Research*, 10(1): 3–12, 1991. ISSN 0278-3649.

[122] Shigeo Hirose, Yasushi Fukuda, and Hidekazu Kikuchi. The gait control system of a quadruped walking vehicle. *Advanced robotics*, 1(4):289–323, 1986. ISSN 0169-1864.

[123] Daan G E Hobbelen and Martijn Wisse. Limit cycle walking. In *Humanoid Robots, Human-like Machines*. InTech, 2007.

[124] Jessica Hodgins. Legged robots on rough terrain: experiments in adjusting step length. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 824–826. IEEE, 1988.

[125] Jessica Kate Hodgins. Legged robots on rough terrain: experiments in adjusting step length, 1989.

[126] Steven Holmes, Georg Klein, and David W Murray. A square root unscented Kalman filter for visual monoSLAM. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3710–3716. IEEE, 2008. ISBN 1424416469.

[127] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013. ISSN 0929-5593.

[128] Shao-Chen Hsu, Xiangru Xu, and Aaron D Ames. Control barrier function based quadratic programs with application to bipedal robotic walking. In *American Control Conference (ACC), 2015*, pages 4542–4548. IEEE, 2015. ISBN 1479986844.

[129] Zheng Huai and Guoquan Huang. Robocentric visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6319–6326. IEEE, 2018.

[130] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. Observability-based rules for designing consistent EKF SLAM estimators. *The International Journal of Robotics Research*, 29(5):502–528, 2010. ISSN 0278-3649.

[131] Qiang Huang, Shuuji Kajita, Noriho Koyachi, Kenji Kaneko, Kazuhito Yokoi, Hirohiko Arai, Kiyoshi Komoriya, and Kazuo Tanie. A high stability, smooth walking pattern for a biped robot. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 1, pages 65–71. IEEE, 1999.

[132] Qiang Huang, Kazuhito Yokoi, Shuuji Kajita, Kenji Kaneko, Hirohiko Arai, Noriho Koyachi, and Kazuo Tanie. Planning walking patterns for a biped robot. *IEEE Transactions on robotics and automation*, 17(3):280–289, 2001. ISSN 1042-296X.

[133] Jemin Hwangbo, Carmine Dario Bellicoso, Péter Fankhauser, and Marco Huttery. Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3872–3878. IEEE, 2016. ISBN 1509037624.

[134] Vadim Indelman, Stephen Williams, Michael Kaess, and Frank Dellaert. Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems*, 61(8):721–738, 2013. ISSN 0921-8890.

[135] Simon J Julier and Jeffrey K Uhlmann. New extension of the Kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–194. International Society for Optics and Photonics, 1997.

[136] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008. ISSN 15523098. doi: 10.1109/TRO.2008.2006706.

[137] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012. ISSN 0278-3649.

[138] Satoshi Kagami. Autobalancer: An online dynamic balance compensation scheme for humanoid robots. *Algorithmic and Computational Robotics: New Directions*, 2001.

[139] Satoshi Kagami, Koichi Nishiwaki, James J Kuffner, Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Online 3D vision, motion planning and bipedal locomotion control coupling system of humanoid robot: H7. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2557–2562. IEEE, 2002. ISBN 0780373987.

[140] S Kajita and K Tani. Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1405–1411 vol.2, 1991. ISBN 0-8186-2163-X. doi: 10.1109/ROBOT.1991.131811.

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

[142] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, and Kiyoshi Fujiwara. Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. In *IEEE International Conference on Robotics and Automation*, pages 1620–1626, 2003. ISBN 0780377362.

[143] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. ISSN 0021-9223.

[144] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996. ISSN 1042-296X.

[145] Hassan K Khalil. Noninear systems. *Prentice-Hall, New Jersey*, 2(5):1–5, 1996.

[146] Arthur J Krener. The convergence of the extended Kalman filter. In *Directions in mathematical systems theory and optimization*, pages 173–182. Springer, 2003.

[147] James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Motion planning for humanoid robots. *Robotics Research*, (Isrr):365–374, 2005. ISSN 16107438. doi: 10.1007/11008941_39.

[148] James J Kuffner. Goal-directed navigation for animated characters using real-time path planning and control. In *Modelling and Motion Capture Techniques for Virtual Environments*, pages 171–186. Springer, 1998.

[149] James J Kuffner and Steven M LaValle. RRT-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000. ISBN 0780358864.

[150] James J Kuffner, Satoshi Kagami, Koichi Nishiwaki, Masayuki Inaba, and Hirochika Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002. ISSN 0929-5593.

[151] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning , estimation , and control design for the atlas humanoid robot. *Autonomous Robots*, 40(3):429–455, 2016. ISSN 1573-7527. doi: 10.1007/s10514-015-9479-3.

[152] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012. ISBN 1461540224.

[153] Ern J Lefferts, F Landis Markley, and Malcolm D Shuster. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 5(5):417–429, 1982. ISSN 0731-5090.

[154] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. ISSN 0278-3649.

[155] Tsai-Yen Li, Pei-Feng Chen, and Pei-Zhi Huang. Motion planning for humanoid walking in a layered environment. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 3421–3427. IEEE, 2003. ISBN 0780377362.

[156] Pei-Chun Lin, Haldun Komsuoglu, and Daniel E Koditschek. A leg configuration measurement system for full-body pose estimates in a hexapod robot. *IEEE Transactions on robotics*, 21(3):411–422, 2005. ISSN 1552-3098.

[157] Pei-Chun Lin, Haldun Komsuoglu, and Daniel E Koditschek. Sensor data fusion for body state estimation in a hexapod robot with dynamical gaits. *IEEE Transactions on Robotics*, 22(5):932–943, 2006. ISSN 1552-3098.

[158] Chenggang Liu and Christopher G Atkeson. Standing balance control using a trajectory library. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3031–3036. IEEE, 2009. ISBN 1424438039.

[159] Andrew W Long, Kevin C Wolfe, Michael J Mashner, and Gregory S Chirikjian. The banana distribution is gaussian: A localization study with exponential coordinates. *Robotics: Science and Systems VIII*, 265, 2013.

[160] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2012. ISSN 1552-3098.

[161] Wen-Loong Ma, Shishir Kolathaya, Eric R Ambrose, Christian M Hubicki, and Aaron D Ames. Bipedal Robotic Running with DURUS-2D: Bridging the Gap between Theory and Experiment. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 265–274. ACM, 2017. ISBN 1450345905.

[162] Venkatesh Madyastha, Vishal Ravindra, Srinath Mallikarjunan, and Anup Goyal. Extended Kalman filter vs. error state Kalman filter for aircraft attitude estimation. In *AIAA Guidance, Navigation, and Control Conference*, page 6615, 2011.

[163] Anirudha Majumdar and Russ Tedrake. Funnel Libraries for Real-Time Robust Feedback Motion Planning. *arXiv*, pages 1–45, 2016. URL `http://arxiv.org/abs/1601.04037`.

[164] F Landis Markley. Multiplicative vs. additive filtering for spacecraft attitude determination. *Dynamics and Control of Systems and Structures in Space*, (467-474), 2004.

[165] Peter S Maybeck. *Stochastic models, estimation, and control*, volume 3. Academic press, 1982. ISBN 0080960030.

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

[167] S J Merhav and M Koifman. Autonomously aided strapdown attitude reference system. *Journal of guidance, control, and dynamics*, 14(6):1164–1172, 1991. ISSN 0731-5090.

[168] John B Moore. Discrete-time fixed-lag smoothing algorithms. *Automatica*, 9(2):163–173, 1973. ISSN 0005-1098.

[169] Benjamin Morris and Jessy W Grizzle. A restricted Poincaré map for determining exponentially stable periodic orbits in systems with impulse effects: Application to bipedal robots. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 4199–4206. IEEE, 2005. ISBN 0780395670.

[170] Raúl Mur-Artal and Juan D Tardós. Visual-inertial monocular SLAM with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017. ISSN 2377-3766.

[171] Richard M Murray. *A mathematical introduction to robotic manipulation*. CRC press, 2017. ISBN 1351469797.

[172] J Murrell. Precision attitude determination for multimission spacecraft. In *Guidance and Control Conference*, page 1248, 1978.

[173] Quan Nguyen and Koushil Sreenath. Safety-Critical Control for Dynamical Bipedal Walking with Precise Footstep Placement.

[174] Quan Nguyen and Koushil Sreenath. Optimal Robust Control for Bipedal Robots through Control Lyapunov Function based Quadratic Programs. In *Robotics: Science and Systems*, 2015.

[175] 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 (RSS)*, 2017.

[176] Simona Nobili, Marco Camurri, Victor Barasuol, Michele Focchi, D Caldwell, Claudio Semini, and M F Fallon. Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots. 2017. ISSN 0992374731.

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

[178] Edwin Olson. A passive solution to the sensor synchronization problem. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1059–1064. IEEE, 2010. ISBN 1424466768.

[179] Hae-won Park, Sangin Park, and Sangbae Kim. Variable-speed Quadrupedal Bounding Using Impulse Planning: Untethered High-speed 3D Running of MIT Cheetah 2. *International Conference on Robotics and Automation*, pages 5163–5170, 2015. doi: 10.1109/ICRA.2015.7139918.

[180] Mihail Pivtoraiko and Alonzo Kelly. Kinodynamic motion planning with state lattice motion primitives. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2172–2179, 2011. ISBN 9781612844541. doi: 10.1109/IROS.2011.6048568.

[181] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[182] David C. Post and James P. Schmiedeler. Velocity disturbance rejection for planar bipeds walking with HZD-based control. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4882–4887, 2014. ISBN 9781479969340. doi: 10.1109/IROS.2014.6943256.

[183] Matthew J Powell, Ayonga Hereid, and Aaron D Ames. Speed regulation in 3D robotic walking through motion transitions between human-inspired partial hybrid zero dynamics. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4803–4810. IEEE, 2013. ISBN 1467356433.

[184] J. E. Pratt and R. Tedrake. Velocity-based stability margins for fast bipedal walking. In *Lecture Notes in Control and Information Sciences*, volume 340, pages 299–324, 2006. ISBN 3540361189. doi: 10.1007/978-3-540-36119-0_14.

[185] Jerry Pratt. Capturability Based Analysis and Control of Legged Locomotion, Part 2: Application to M2V2, a Lower Body Humanoid. *International Journal Of Robotics Research*, Submitted(January):[In Preparation], 2011. ISSN 02783649. doi: 10.1177/0278364912452762. URL `http://ijr.sagepub.com/content/early/2012/08/02/0278364912452762.abstract`.

[186] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[187] Xingye Da Quan Nguyen, J W Grizzle, and Koushil Sreenath. Dynamic Walking on Stepping Stones with Gait Library and Control Barrier Functions. *Arbor*, 1001:48109.

[188] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[189] Marc H. Raibert and Ernest R. Tello. Legged Robots That Balance. *IEEE Expert*, 1 (4), 1986. ISSN 0885-9000. doi: 10.1109/MEX.1986.4307016.

[190] Alireza Ramezani, Jonathan W Hurst, Kaveh Akbari Hamed, and Jessy W Grizzle. Performance analysis and feedback control of ATRIAS, a three-dimensional bipedal robot. *Journal of Dynamic Systems, Measurement, and Control*, 136(2):21012, 2014. ISSN 0022-0434.

[191] Jacob Reher, Eric A Cousineau, Ayonga Hereid, Christian M Hubicki, and Aaron D Ames. Realizing Dynamic and Efficient Bipedal Locomotion on the Humanoid Robot DURUS.

[192] Jacob P Reher, Ayonga Hereid, Shishir Kolathaya, Christian M Hubicki, and Aaron D Ames. Algorithmic foundations of realizing multi-contact locomotion on the humanoid

robot DURUS. In *The International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[193] Ioannis M Rekleitis. A particle filter tutorial for mobile robot localization. *Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02*, 2004.

[194] Siavash Rezazadeh, Christian M 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. *Proceedings of the ASME 2015 Dynamic Systems and Control Conference ASME/DSCC 2015*, 2015. doi: 10.1115/DSCC2015-9899.

[195] Gerald P Roston and Eric P Krotkov. Dead reckoning navigation for walking robots. Technical report, 1991.

[196] Nicholas Rotella, Michael Bloesch, Ludovic Righetti, and Stefan Schaal. State Estimation for a Humanoid Robot. 2014.

[197] Stergios I Roumeliotis, Gaurav S Sukhatme, and George A Bekey. Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1656–1663. IEEE, 1999. ISBN 0780351800.

[198] Yoshiaki Sakagami, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo Fujimura. The intelligent ASIMO: System overview and integration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2478–2483. IEEE, 2002. ISBN 0780373987.

[199] Kenneth D Sebesta and Nicolas Boizot. A real-time adaptive high-gain ekf, applied to a quadcopter inertial navigation system. *IEEE Transactions on Industrial Electronics*, 61(1):495–503, 2014.

[200] Peyman Setoodeh, Alireza Khayatian, and Ebrahim Frajah. Attitude estimation by separate-bias Kalman filter-based data fusion. *The Journal of Navigation*, 57(2):261–273, 2004. ISSN 1469-7785.

[201] Ching-Long Shih, Y Z Li, S Churng, T T Lee, and William A Gruver. Trajectory synthesis and physical admissibility for a biped robot during the single-support phase. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 1646–1652. IEEE, 1990. ISBN 0818690615.

[202] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, 2010. ISSN 1556-4967.

[203] Surya PN Singh, Paul J Csonka, and Kenneth J Waldron. Optical flow aided motion estimation for legged locomotion. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1738–1743. IEEE, 2006.

[204] Duncan Smith and Sameer Singh. Approaches to multisensor data fusion in target tracking: A survey. *IEEE transactions on knowledge and data engineering*, 18(12): 1696–1710, 2006. ISSN 1041-4347.

[205] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[206] Steven T Smith. Optimization techniques on Riemannian manifolds. *Fields institute communications*, 3(3):113–135, 1994.

[207] Joan Sola. Quaternion kinematics for the error-state Kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.

[208] Yongkyu Song and Jessy W Grizzle. The extended Kalman filter as a local asymptotic observer for nonlinear discrete-time systems. In *American Control Conference, 1992*, pages 3365–3369. IEEE, 1992. ISBN 0780302109.

[209] Eduardo D Sontag. A Lyapunov-like characterization of asymptotic controllability. *SIAM Journal on Control and Optimization*, 21(3):462–471, 1983. ISSN 0363-0129.

[210] Eduardo D Sontag. Input to state stability: Basic concepts and results. In *Nonlinear and optimal control theory*, pages 163–220. Springer, 2008.

[211] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar, et al. *Robot modeling and control*. 2006.

[212] K. Sreenath, H.-W. Park, I. Poulakakis, and J. W. Grizzle. A Compliant Hybrid Zero Dynamics Controller for Stable, Efficient and Fast Bipedal Walking on MABEL. *The International Journal of Robotics Research*, 30(9):1170–1193, aug 2011. ISSN 0278-3649. doi: 10.1177/0278364910379882. URL `http://ijr.sagepub.com/cgi/doi/10.1177/0278364910379882`.

[213] Koushil Sreenath, Hae Won Park, and J. W. Grizzle. Design and experimental implementation of a compliant hybrid zero dynamics controller with active force control for running on MABEL. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 51–56, 2012. ISBN 9781467314039. doi: 10.1109/ICRA.2012.6224944.

[214] Koushil Sreenath, H.-W. Park, Ioannis Poulakakis, and J. Grizzle. Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on MABEL. *The International Journal of Robotics Research*, 32(3):324–345, 2013. ISSN 0278-3649. doi: 10.1177/0278364912473344. URL `http://ijr.sagepub.com/content/32/3/324.abstract{%}5Cnhttp://ijr.sagepub.com/cgi/doi/10.1177/0278364912473344`.

[215] John Stillwell. *Naive lie theory*. Springer Science & Business Media, 2008. ISBN 038778215X.

[216] Tomomichi Sugihara, Yoshihiko Nakamura, and Hirochika Inoue. Real-time humanoid motion generation through ZMP manipulation based on inverted pendulum control. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1404–1409. IEEE, 2002. ISBN 0780372727.

[217] Russ Tedrake. Lqr-trees: Feedback motion planning on sparse randomized trees. 2009.

[218] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004. ISSN 0278-3649.

[219] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005. ISBN 0262303809.

[220] David Titterton, John L Weston, and John Weston. *Strapdown inertial navigation technology*, volume 17. IET, 2004. ISBN 0863413587.

[221] Miles A. Townsend. Biped gait stabilization via foot placement. *Journal of Biomechanics*, 18(1):21–38, 1985. ISSN 00219290. doi: 10.1016/0021-9290(85)90042-9.

[222] Nikolas Trawny and Stergios I Roumeliotis. Indirect Kalman filter for 3D attitude estimation. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2:2005, 2005.

[223] Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers. Direct visual-inertial odometry with stereo cameras. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1885–1892. IEEE, 2016. ISBN 1467380261.

[224] Pieter Van Zutven, Dragan Kostić, and Henk Nijmeijer. Foot placement for planar bipeds with point feet. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 983–988, 2012. ISBN 9781467314039. doi: 10.1109/ICRA.2012.6224823.

[225] Miomir Vukobratović and Branislav Borovac. Zero-Moment Point — Thirty Five Years of Its Life. *International Journal of Humanoid Robotics*, 01(01): 157–173, mar 2004. ISSN 0219-8436. doi: 10.1142/S0219843604000083. URL `http://www.worldscientific.com/doi/abs/10.1142/S0219843604000083https://www.cs.cmu.edu/{~}cga/legs/vukobratovic.pdf`.

[226] Miomir Vukobratovic, A A Frank, and Davor Juricic. On the stability of biped locomotion. *IEEE Transactions on Biomedical Engineering*, (1):25–36, 1970. ISSN 0018-9294.

[227] Matthew R Walter, Ryan M Eustice, and John J Leonard. Exactly sparse extended information filters for feature-based SLAM. *The International Journal of Robotics Research*, 26(4):335–359, 2007. ISSN 0278-3649.

[228] Eric A Wan and Rudolph Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. Ieee, 2000. ISBN 0780358007.

[229] Yunfeng Wang and Gregory S Chirikjian. Error propagation on the Euclidean group with applications to manipulator kinematics. *IEEE Transactions on Robotics*, 22(4): 591–602, 2006. ISSN 1552-3098.

[230] Yunfeng Wang and Gregory S Chirikjian. Nonparametric second-order theory of error propagation on motion groups. *The International journal of robotics research*, 27(11-12):1258–1273, 2008. ISSN 0278-3649.

[231] Patrick M Wensing and David E Orin. High-Speed Humanoid Running Through Control with a 3D-SLIP Model. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5134–5140, 2013. ISBN 9781467363587.

[232] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, 2003. ISSN 00189286. doi: 10.1109/TAC.2002.806653.

[233] Eric R Westervelt, Christine Chevallereau, Jun Ho Choi, Benjamin Morris, and Jessy W Grizzle. *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2007. ISBN 1420053736.

[234] Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40(12):462–467, 2007. ISSN 1474-6670.

[235] Derek L. Wight, Eric G. Kubica, and David W. L. Wang. Introduction of the Foot Placement Estimator: A Dynamic Measure of Balance for Bipedal Robotics. *Journal of Computational and Nonlinear Dynamics*, 3(1):011009, 2008. ISSN 15551423. doi: 10.1115/1.2815334.

[236] Oliver J Woodman. An introduction to inertial navigation. Technical report, 2007.

[237] Kanzhi Wu, Teng Zhang, Daobilige Su, Shoudong Huang, and Gamini Dissanayake. An Invariant-EKF VINS algorithm for improving consistency. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[238] X. Xinjilefu, Siyuan Feng, Weiwei Huang, and Christopher G. Atkeson. Decoupled state estimation for humanoids using full-body dynamics. *Proceedings - IEEE International Conference on Robotics and Automation*, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6906609.

[239] Feng Xue, Xiaoping Chen, Jinsu Liu, and Daniele Nardi. Real time biped walking gait pattern generator for a real robot. In *Robot Soccer World Cup*, pages 210–221. Springer, 2011.

[240] Zhenfei Yang and Shaojie Shen. Monocular visual–inertial state estimation with online initialization and camera–imu extrinsic calibration. *IEEE Transactions on Automation Science and Engineering*, 14(1):39–51, 2017. ISSN 1545-5955.

[241] Eiichi Yoshida, Igor Belousov, Claudia Esteves, and J-P Laumond. Humanoid motion planning for dynamic tasks. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 1–6. IEEE, 2005. ISBN 0780393201.

[242] Pifu Zhang, Jason Gu, Evangelos E Milios, and Peter Huynh. Navigation with IMU/GPS/digital compass with unscented Kalman filter. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 3, pages 1497–1502. IEEE, 2005. ISBN 078039044X.

[243] Teng Zhang, Kanzhi Wu, Jingwei Song, Shoudong Huang, and Gamini Dissanayake. Convergence and consistency analysis for a 3-D Invariant-EKF SLAM. *IEEE Robotics and Automation Letters*, 2(2):733–740, 2017. ISSN 2377-3766.

[244] Huihua Zhao, Jonathan Horn, Jacob Reher, Victor Paredes, and Aaron D Ames. A hybrid systems and optimization-based control approach to realizing multi-contact locomotion on transfemoral prostheses. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 1607–1612. IEEE, 2015. ISBN 1479978868.

[245] Ye Zhao, Benito R Fernandez, and Luis Sentis. Robust optimal planning and control of non-periodic bipedal locomotion with a centroidal momentum model. *The International Journal of Robotics Research*, 36(11):1211–1242, 2017.

[246] Yuan F Zheng and Jie Shen. Gait synthesis for the SD-2 biped robot to climb sloping surface. *IEEE Transactions on Robotics and Automation*, 6(1):86–96, 1990. ISSN 1042-296X.