Enhancing State Estimation in Quadruped Robots: Multi-Sensor Fusion for Challenging Terrains

Ylenia Nisticò

Istituto Italiano di Tecnologia, Italy
Università degli studi di Genova, Italy

Thesis submitted for the degree of Doctor of Philosophy (37° cycle)

February 2025

Ylenia Nisticò

Enhancing State Estimation in Quadruped Robots: Multi-Sensor Fusion for Challenging Terrains

Candidate Student for the Ph.D. Program in *Bioengineering and Robotics*
Curriculum in *Advanced and Humanoid Robotics*, Università degli studi di Genova.
Genoa, Italy, February 2025

Tutors:

Dr. Geoff Fink, Assistant Professor
Dynamic Legged Systems (DLS) lab, Istituto Italiano di Tecnologia (IIT), Genoa, Italy
Department of Engineering, Thompson Rivers University, Kamloops, BC, Canada

Dr. João Carlos Virgolino Soares
Dynamic Legged Systems (DLS) lab, Istituto Italiano di Tecnologia (IIT), Genoa, Italy

Dr. Claudio Semini, Principal Investigator
Dynamic Legged Systems (DLS) lab, Istituto Italiano di Tecnologia (IIT), Genoa, Italy

---

Head of the Ph.D. Program:

Dr. Paolo Massobrio, Associate Professor
Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), Università degli studi di Genova, Italy

---

Reviewers:

Dr. Maurice Fallon, Associate Professor
Department of Engineering Science, University of Oxford, UK

Dr. Joan Solà
Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Universitat Politècnica de Catalunya, Barcelona, Spain

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing that is the outcome of work done in collaboration with others, except as specified in the text. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables, and equations, and has fewer than 150 figures.

Ylenia Nisticò

Genoa, Italy, 4th February 2024

Acknowledgements

Looking back, this thesis proves that even a series of small *disasters*, tackled with patience and determination, can lead to concrete results. However, reaching this point would not have been possible without the support of many people over these three years of doctoral studies. I would like to express my deepest gratitude to all those who have contributed to this journey.

First and foremost, I extend my gratitude to Claudio, head of the DLS Lab, for fostering a motivating and collaborative work environment. Your precise and thoughtful feedback has been invaluable in shaping this work, and I deeply appreciate your consistent support and empathy during this journey.

I am also profoundly grateful to my supervisors, Geoff and João. I have gained invaluable knowledge and insights from your guidance and expertise, and I thank you for your patience and encouragement throughout my research.

My sincere thanks go to the reviewers of this thesis, Prof. Maurice Fallon and Prof. Joan Solà, whose constructive feedback significantly improved the quality of this work and challenged me to refine its details further.

To the former members of the DLS, particularly the "Office 2 gang" – Tavo, Shamel, Luca, and Abdo – thank you for creating such a welcoming and engaging atmosphere. Your fellowhsip and support were pivotal in making IIT feel like the right place to pursue this journey. A special acknowledgment goes to Chundri for his contagious positivity, which brightened many days.

To the current members of the DLS, I am grateful for the energy and enthusiasm you bring to the lab. Your diverse perspectives and expertise have been a source of inspiration and motivation. A heartfelt thank you to the "Magi", Gianluca and Marco, for their tireless work and for a relationship that went far beyond mere professional collaboration. Finally, thanks to Angelo, Giulio, Lorenzo, and Giovanni — you are one of the main reasons I willingly brave the traffic of Genoa every morning to come to the office.

A big thank you to the administrative staff, especially Laura, Monica, Ines, Valentina and Emanuele, for your constant support and for making Italian bureaucracy feel a little less daunting.

During my time in Genoa, I was fortunate to find a second home thanks to the incredible individuals who made me appreciate this city: the "Baciotti et al." group. Special thanks to Antonello, a fellow traveler through the challenges of a Ph.D.; Carlotta, whose dynamic presence is unforgettable; Giulio R., for our stimulating brainstorming sessions; Maria, an ideal flatmate and thoughtful "host"; and Riccardo and Paolo, whose

companionship and meticulous planning enhanced every journey.

I would also like to acknowledge my friends from "La Contea": Chiara (the mayor), Alessia, Andrea (Dederni), Francesco (Pepo), Gabriele (Ficognaro), Laura, Matteo, Michele, Rachele, and the *brilliant doctors* Giorgio (Giorgi), Simone (Tolo), and Alessandro (Gendili). The moments we shared remain among the most cherished of my life.

A special thank you goes to my lifelong friends, Martina, Nadia, Michela, and Federica, whose unwavering friendship has been a source of comfort and joy, even from afar.

Lastly, I extend my heartfelt gratitude to Giuseppe. Your constant support, patience, and understanding have been invaluable, and I am deeply grateful for your presence during every step of this journey.

I conclude by dedicating this thesis to my imperfect and splendid family. To my parents, Maria Grazia and Claudio (the true Eng. Nisticò), thank you for your unwavering belief in me and for supporting me in every decision, even the most unconventional ones. To my sister, Laura, thank you for being a steadfast source of strength and encouragement, helping me navigate the challenges I encountered along the way.

To all of you, *thank you*.

Ylenia Nisticò

Ringraziamenti

Riflettendo su questo percorso, questa tesi dimostra che anche una serie di piccoli *disastri*, affrontati con perseveranza e determinazione, può condurre a risultati significativi. Tuttavia, il raggiungimento di questo traguardo non sarebbe stato possibile senza il supporto e la guida di numerose persone che hanno contribuito alla mia crescita durante questi tre anni di dottorato.

Per prima cosa, desidero esprimere la mia gratitudine a Claudio, capo del DLS Lab, per aver creato un ambiente di lavoro stimolante e collaborativo. I tuoi feedback, sempre precisi e attenti, sono stati fondamentali per plasmare questo lavoro, e apprezzo profondamente il tuo costante supporto e la tua empatia durante questo percorso.

Sono inoltre profondamente grata ai miei tutor, Geoff e João. Ho acquisito conoscenze e competenze inestimabili grazie alla vostra guida e alla vostra esperienza, e vi ringrazio per la pazienza e l'incoraggiamento che mi avete dimostrato nel corso della nostra ricerca.

Un ringraziamento sincero va ai revisori di questa tesi, il Prof. Maurice Fallon e il Prof. Joan Solà, i cui commenti costruttivi hanno migliorato significativamente la qualità del lavoro e mi hanno spinto a perfezionarne ulteriormente i dettagli.

Ai membri passati del DLS, in particolare alla "Gang dell'ufficio 2" – Tavo, Shamel, Luca e Abdo – grazie per aver creato un'atmosfera accogliente e stimolante. La vostra compagnia e il vostro supporto sono stati fondamentali per farmi sentire che l'IIT fosse il posto giusto per intraprendere questo percorso. Un ringraziamento speciale va a Chundri per la sua positività contagiosa, che ha reso molte giornate più leggere.

Ai membri attuali del DLS, sono grata per l'energia e l'entusiasmo che portate nel laboratorio. Le vostre prospettive e competenze diverse sono state fonte di ispirazione e motivazione. Un sentito ringraziamento ai "Magi", Gianluca e Marco, per il loro instancabile lavoro e per un rapporto che ha superato di gran lunga la semplice collaborazione professionale. Infine, grazie ad Angelo, Giulio, Lorenzo e Giovanni: siete una delle principali ragioni per cui affronto volentieri il traffico di Genova ogni mattina per venire in ufficio.

Sono inoltre debitrice allo staff amministrativo, in particolare a Laura, Monica, Ines, Valentina ed Emanuele, per il loro supporto instancabile e per aver reso le complessità della burocrazia italiana un po' meno scoraggianti.

Durante il tempo trascorso a Genova, ho trovato una seconda casa grazie alle incredibili persone che mi hanno fatto scoprire e apprezzare questa città: i "Baciotti et al.". Un ringraziamento speciale ad Antonello, compagno di mille "sventure" del dottorato; a Carlotta, che con la sua energia non passa di certo inosservata; a Giulio R. per le nostre

stimolanti chiacchierate in cui si discuteva di Algebra di Lie e Centroidal MPC; a Maria, una coinquilina ideale e "padrona di casa" impeccabile; e a Riccardo e Paolo, straordinari compagni di viaggio e organizzatori instancabili.

Vorrei anche ringraziare i miei amici de "La Contea": Chiara (il sindaco), Alessia, Andrea (Dederni), Francesco (Pepo), Gabriele (Ficognaro), Laura, Matteo, Michele, Rachele e i dottorissimi Giorgio (Giorgi), Simone (Tolo) e Alessandro (Gendili). Alcuni dei momenti più belli della mia vita li ho vissuti insieme a voi.

Un ringraziamento speciale va alle mie amiche di sempre, Martina, Nadia, Michela e Federica, che rimangono un punto di riferimento immutabile in ogni ritorno a casa.

Infine, desidero ringraziare Giuseppe. Grazie per il tuo supporto costante, la tua pazienza e la tua comprensione, che sono stati inestimabili in ogni momento di questo percorso, e non solo.

Concludo dedicando questa tesi alla mia imperfetta e splendida famiglia. Ai miei genitori, Maria Grazia e Claudio (il vero Ing. Nisticò), grazie per il vostro continuo, incondizionato sostegno e per aver sostenuto ogni mia singola scelta, anche quelle meno convenzionali. E a Laura, mia sorella, un punto fermo che mi ha aiutato a superare molte delle sfide incontrate lungo il cammino.

A tutti voi, *Grazie*.

Ylenia Nisticò

Abstract

Quadruped robots are increasingly important in fields such as automation, inspection, and monitoring due to their unique capability to navigate complex and unstructured environments. Unlike wheeled robots, legged robots can overcome various obstacles, making them highly versatile for real-world applications. However, this versatility comes with challenges. Robust performance in such environments depends heavily on accurate state estimation, which provides essential information on the robot's position, orientation, and velocity. State estimation is inherently more complex in legged robots due to their dynamic gait and the constantly changing points of contact with the ground. The integration of leg kinematics data can play a crucial role in improving state estimation accuracy, as it supplies additional information that complements data from other sensors. My research aims to advance state estimation capabilities in quadruped robots, enabling them to autonomously recognize, interpret, and adapt to their surroundings, which is vital for maintaining balance, perceiving obstacles, and executing complex maneuvers in dynamic and unpredictable terrains.

To achieve this, I concentrated on developing state estimation algorithms that integrate real-time sensor data, leveraging multiple sensor streams for enhanced precision and reliability. This integration is critical to achieving dynamic motion control and autonomous operation, allowing robots to make immediate adjustments to their gait and trajectory in response to environmental changes. Additionally, I explored the use of Lie groups to design state estimation frameworks that employ both filtering and smoothing techniques. The filtering approach offers real-time responsiveness by using current sensor data to update the robot's state, while the smoothing approach optimizes over a set of past states for improved accuracy.

The primary outcomes of my research are embodied in three major contributions. The first is MUSE, a MUlti-sensor State Estimator designed specifically for quadruped robots. MUSE integrates data from a range of sensors, including Inertial Measurement Units (IMUs), encoders, force/torque sensors, cameras, and LiDARs, to provide accurate, reliable, and real-time state estimation even in challenging real-world environments with uneven or slippery surfaces. MUSE incorporates a dedicated slip detection module, that enables the robot to detect slippery terrain and correct the estimate by discarding the unreliable leg odometry measurements. Additionally, MUSE is built to be modular and flexible, allowing it to interface with various robot platforms and sensor configurations, making it adaptable to different applications and terrains. The real-time capabilities were demonstrated in real-time operations, where MUSE provided online feedback to the

locomotion controller. Furthermore, MUSE is developed as an open-source tool, encouraging other researchers and engineers to use, modify, and build upon this work to further advance the field. The second major contribution is the development of frameworks for invariant state estimation, specifically a multi-sensor Invariant Extended Kalman Filter (InEKF) and an Invariant Smoother (IS). Both frameworks utilize Lie groups to incorporate leg kinematics, LiDAR positional data, and GPS coordinates (when available) to refine the robot's state estimate and determine its global position. These frameworks are among the first to successfully integrate both proprioceptive (internal) and exteroceptive (external) measurements for invariant state estimation in legged robots, representing a significant innovation in the field. The third and final major contribution is the extensive testing of the proposed algorithms on multiple robots of varying sizes, ranging from the 21 Kg Unitree Aliengo to the 90 Kg HyQ robot from the Italian Institute of Technology.

Looking ahead, future research will focus on enhancing individual components of these frameworks to improve overall estimation performance. For instance, advanced terrain estimation will play a key role in achieving fully autonomous operations. Estimating parameters such as the friction coefficient, as well as the geometrical and physical properties of the terrain (e.g. inclination and softness), will enable the robot to make real-time adjustments based on the terrain it encounters. Additionally, developing increasingly reliable mapping techniques will support long-term autonomy by allowing the robot to build a comprehensive understanding of its environment, reducing its reliance on external guidance or teleoperation. These advancements are expected to push the boundaries of autonomy in legged robots, allowing them to navigate complex environments independently, maintain stability on challenging terrain, and perform sophisticated tasks with minimal human intervention.

Keywords: legged robots, state estimation, sensor fusion, localization, odometry.

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface

ATE Absolute Trajectory Error

BCH Baker-Campbell-Hausdorff

CL Contact Loop

CPU Central Processing Unit

DDS Data Distribution Service

DLIO Direct LiDAR Inertial Odometry

DLO Direct LiDAR Odometry

DOFs Degrees of Freedom

EKF Extended Kalman Filter

FSC Fire Service College

GES Globally Exponentially Stable

GNSS Global Navigation Satellite System

GPS Global Positioning System

GPU Graphics processing unit

GRF Ground Reaction Force

GRFs Ground Reaction Forces

GT Ground Truth

HyQ Hydraulically actuated Quadruped

ICP Iterative Closest Point

IEKF Iterated Extended Kalman Filter

IIT Italian Institute of Technology

IMU Inertial Measurement Unit

InEKF Invariant Extended Kalman Filter

IS Invariant Smoother

JS Joint State

k-d k-dimensional

KAIST Korean Advanced Institute of Science and Technology

KD Kinematics/Dynamics

KF Kalman Filter

KILO Kinematic Inertial Leg Odometry

KISS Keep It Small and Simple

LF Left Front

LH Left Hind

LiDAR Light Detection and Ranging

LIO LiDAR Inertial Odometry

LKF Linearized Kalman Filter

LOAM Lidar Odometry and Mapping

LTV Linear Time-Varying

MAP Maximum A Posteriori

MPC Model Predictive Controller

MUSE MUlti-sensor State Estimator

NLO Nonlinear Observer

ORB Oriented FAST and Rotated BRIEF

PCA Principal Component Analysis

PD Proportional-Derivative

PE Persistency of Excitation

PF Particle Filter

QoS Quality of Service

RA-L Robotics and Automation Letters

RF Right Front

RGB Red Green Blue

RGB-D Red Green Blue-Depth

RH Right Hind

RMSE Root Mean Square Error

ROS Robot Operating System

RPE Relative Pose Error

SAM Smoothing and Mapping

SD Slip Detection

SF Sensor Fusion

SLAM Simultaneous Localization And Mapping

SR Slip Rejection

SVO Semi-direct Visual Odometry

TSIF Two-State Information Filter

UAVs Unmanned Aerial Vehicles

UKF Unscented Kalman Filter

VILO Visual Inertial Leg Odometry

VINS Visual Inertial system

VIO Visual Inertial Odometry

WS Window Size

XKF eXogeneous Kalman Filter

Chapter 1

Introduction

1.1 Preface

In this thesis, I chose to use "we" instead of "I", to acknowledge that the research presented was enriched thanks to the valuable assistance of researchers and engineers of the DLS lab. Although I developed the core aspects of the project, the guidance, feedback, and discussions with my colleagues contributed significantly to its success. My exact contributions to each work, along with the contributions of my colleagues, are detailed in the preface of each chapter.

1.2 Motivation

In recent years, legged robots have emerged as a powerful alternative to traditional wheeled or tracked robots, particularly for navigating complex, unstructured environments. Unlike wheeled robots, which are highly efficient on flat surfaces, legged robots offer a significant advantage in rough, uneven, or unpredictable terrains. Their ability to step over obstacles, adjust to varying ground conditions, and maintain balance makes them ideal for applications ranging from search and rescue missions in disaster-stricken areas to planetary exploration. Quadruped robots, in particular, have garnered attention for their ability to achieve both stability and maneuverability in challenging environments.

The unique capabilities of legged robots stem from their biologically inspired design, mimicking the locomotion of animals such as dogs, horses, and insects. These robots can dynamically adjust their gait, posture, and limb movements to interact with the terrain

more effectively than wheeled or tracked systems. This advantage becomes critical in environments where obstacles, debris, or irregular surfaces would immobilize a wheeled robot. In scenarios such as forested areas, mountainous terrain, or urban ruins, where human intervention may be limited or unsafe, legged robots have a clear edge.

However, the successful operation of legged robots in such environments hinges on one essential component: **state estimation**, which provides the real-time data needed to navigate and adapt to unstructured environments. Accurately estimating the robot's state, including position, orientation, velocity, and interaction with the environment, is critical for robust, autonomous navigation, enabling the robot to perceive surroundings, maintain balance, and execute complex maneuvers.

State estimation is essential for (1) Perception and Localization: state estimation informs the robot of its current position, orientation, and motion relative to its surroundings. This is crucial for understanding the environment and for making decisions about movement and obstacle avoidance. Without accurate localization, the robot would be unable to navigate autonomously; (2) Global Localization: in large or outdoor environments, legged robots must often navigate using global reference frames. State estimation integrates sensor data with environmental maps, allowing for robust localization over long distances. This capability is essential for tasks such as exploration or rescue operations, where the robot needs to understand both its local surroundings and its position relative to the broader environment in a known map; (3) Simultaneous Localization And Mapping (SLAM): this process involves building a spatial representation of the environment while simultaneously localizing the robot within that map. SLAM techniques combine data from sensors such as cameras and Light Detection and Ranging (LiDAR), to create and update these maps in real time. This helps the robot identify obstacles, plan paths, and adapt to dynamic or unknown environments; (4) Obstacle Avoidance: with integrated perception systems such as cameras, LiDAR sensors, and depth sensors, quadruped robots rely on state estimation to detect and avoid obstacles in their path. Accurate state information allows for timely adjustments in trajectory to prevent collisions, enhancing the robot's ability to move through cluttered and dynamic environments; (5) Dynamic Stability Control: maintaining stability during locomotion is one of the most critical challenges for quadruped robots. State estimation helps track the robot's center of mass and the position of each limb, enabling real-time adjustments to prevent falls or tipping, particularly on difficult terrain. (6) Fault Detection and Recovery: state estimation also contributes to fault detection by monitoring sensor data or robot behavior inconsistencies. When issues are detected, the robot can switch to backup sensors or employ recovery strategies, ensuring continued operation even in the

face of sensor failure or environmental challenges.

In this context, our research focused on advancing the state estimation capabilities of quadruped robots. We aimed to improve the robots' ability to recognize and take information from the environment, with the objective of enabling more reliable and complex autonomous behaviors in the future. By optimizing the algorithms for state estimation and enhancing the coordination of diverse sensor systems, such as Inertial Measurement Unit (IMU), cameras, and LiDARs, we were able to enhance the pose and velocity estimation accuracy of quadruped robots walking in challenging terrain, which improved their navigation capabilities.

In addition, our work focused on organizing and online processing sensor data, allowing for the integration of multiple data streams to ensure precise and reliable state estimation. This enhanced state estimation was critical for facilitating dynamic motion control, enabling the robots to safely and autonomously operate in complex, unstructured environments (Chapter 4).

We also explored Lie groups to design two invariant state estimation frameworks that leverage filtering and smoothing approaches (Chapter 5). Through a comparative analysis between these two approaches, we highlight the trade-offs between real-time responsiveness and estimation accuracy, providing insights into the strengths and limitations of both techniques for quadruped robot navigation.

1.3 Contribution

Over the three years of this PhD research, we investigated and developed various approaches to address the challenges of state estimation for legged robots. The following core scientific questions guided the design and development of our state estimation framework:

1. How can we enable a robot to perceive and adapt to its surrounding environment, specifically in detecting whether the terrain it traverses is slippery?

2. Once detected, in what ways can slip detection enhance the accuracy and reliability of state estimation?

3. What are the most effective methods for achieving robust, accurate, and real-time state estimation?

This thesis focuses on addressing some of these questions, particularly those related to slippery terrain detection and achieving robust, accurate state estimation. A remaining

question is also: *How do dynamic environments, such as moving objects or subjects, influence the performance of state estimation?* The answer to this is outlined as a potential direction for future research.

Over the course of this research, we developed several approaches to address the challenges of state estimation for quadruped robots. Our contributions focus on integrating multiple sensors, handling slippery and uneven terrain, and achieving robust, real-time performance. The primary contributions of this thesis are summarized as follows:

- **Development of a MUlti-sensor State Estimator (MUSE)**. We designed MUSE, a state estimator for quadruped robots that integrates data from multiple sensors to achieve accurate and reliable state estimation. MUSE is tailored to address real-world challenges, such as navigating slippery or uneven terrain. This work is described in Chapter 4 and is the subject of a central publication currently under review for IEEE Robotics and Automation Letters (RA-L). Preliminary results of MUSE were also presented at the 2024 I-RIM 3D conference.

  Specifically, key features and advancement of MUSE include:

  - Slip detection module: to the best of our knowledge, MUSE is the first multi-sensor (including proprioceptive and exteroceptive sensors) state estimation pipeline to incorporate a dedicated slip detection module. This module is crucial for enabling the robot to detect and adapt to slippery terrain. A detailed description of this module is provided in Chapter 3, and it has been separately published in one of our previous publications (Nisticò et al. [1]).
  - Real-Time operation: Unlike previous works such as VILENS (Wisth et al. [2]), WALK-VIO (Lim et al. [3]), and STEP (Kim et al. [4]), which are described in Chapter 2, we proved that MUSE can provide real-time feedback to the locomotion controller. This capability was demonstrated in experiments conducted on the Aliengo robot, enabling responsive and adaptive navigation.
  - Comprehensive Evaluation: MUSE was validated through both online and offline experiments across different platforms and environments. We tested it on the Aliengo robot (Unitree [5]) in challenging indoor scenarios and on the ANYmal B300 robot using the Fire Service College (FSC) Dataset (Wisth et al. [2]). Our results show significant improvements compared to other state estimators for legged robots. Specifically: a 67.6% reduction in translational error compared to Pronto (Camurri et al. [6]), a 26.67% reduction compared to VILENS (Wisth et al. [2]), and a 45.9% reduction in absolute trajectory

error compared to TSIF (Bloesch et al. [7]). Additionally, MUSE outperforms Direct LiDAR Inertial Odometry (DLIO) (Chen et al. [8]), which is a LiDAR-inertial odometry algorithm, in rotational error and computation frequency.

- **Development of a Right-Invariant Extended Kalman Filter (InEKF) and a Right-Invariant Smoother (IS)**. We proposed a Right-InEKF and a Right-IS that utilize Lie groups to fuse data from leg kinematics, LiDAR odometry, and Global Positioning System (GPS) coordinates (when available). These frameworks effectively address the position drift inherent in proprioceptive-only methods. This work, conducted in collaboration with the Korean Advanced Institute of Science and Technology (KAIST), is described in Chapter 5, and a paper is currently under review for IEEE RA-L. The key contributions of this work are twofold:
  - Integration of LiDAR and GPS: to the best of our knowledge, this is the first work to incorporate both LiDAR odometry and GPS into the InEKF for legged robots, and it is the first work to include exteroceptive measurements in an IS framework for legged robots. To handle LiDAR's low frequency (approximately 10 Hz), we implemented Iterative Closest Point (ICP) registration from (Vizzo et al. [9]) in a parallel thread, ensuring the estimator maintains fast computation times.
  - Verification across platforms: We validated these frameworks on the Hound and Hound2 robots (Shin et al. [10]) in both indoor and outdoor environments, while also providing a comparison between the performance of the two proposed methods. Additionally, we benchmarked the obtained results against two state-of-the-art LiDAR-based odometry systems (Vizzo et al. [9], Xu and Zhang [11]), demonstrating the robustness and accuracy of our approach.
- **Extensive testing** of our algorithms and frameworks on different robots with varying actuation types and sizes. Specifically, the **slip detection algorithm** was tested on the **90 Kg** Hydraulically actuated Quadruped (HyQ) robot; **MUSE** was evaluated on electrically actuated robots, including the **21 Kg** Aliengo and the **30 Kg** ANYmal. Finally, the **invariant state estimation** frameworks were tested on the **45 Kg** Hound and on the **50 Kg** Hound2, which are also electrically actuated.

During my Ph.D., I also had the privilege of contributing to developing a software framework for mobile robots, DLS2. This framework played a crucial role in the real-

ization of the MUSE project, as it was employed to manage real-time communication between the sensors and the state estimator. Although not directly related to state estimation, I participated in the development of this framework (in particular in the testing and refinement) which is briefly described in Appendix A.

1.4 Organization of the Thesis

This thesis is organized as follows:

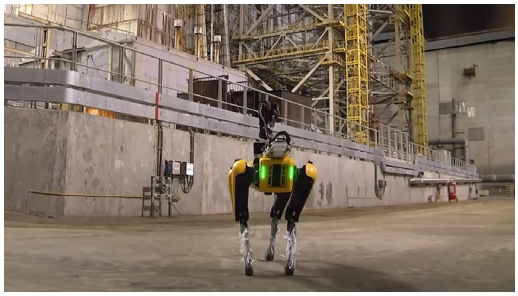
- **Chapter 2** provides an overview of the state of the art in state estimation for legged robots, highlighting the challenges and opportunities in this field.
- **Chapter 3** introduces the concept of slip detection and its importance for state estimation in quadruped robots. We present a novel approach to slip detection based on the analysis of leg kinematics data.
- **Chapter 4** describes the development of MUSE, a multi-sensor state estimator designed specifically for quadruped robots. We detail the architecture, algorithms, and performance evaluation of MUSE.
- **Chapter 5** presents the development of InEKF and IS, two invariant frameworks for state estimation that leverage Lie groups to integrate leg kinematics, LiDAR data, and GPS coordinates in the measurement model.
- **Chapter 6** concludes the thesis by summarizing the key findings and contributions, discussing the implications of the research, and outlining potential directions for future work.
- **Appendix A** provides additional information on the DLS2 software framework, which was used to support the development of the MUSE project.
- **Appendix B** contains the list of publications achieved during the Ph.D.

Chapter 2

State of the Art

In recent years, legged robots have gained significant attention due to their ability to traverse complex terrains where wheeled or tracked robots struggle. These robots have found applications across diverse fields, for instance: in rescue operations, such as ANYmal by ANYbotics (Fankhauser and Hutter [12]), depicted in Fig. 2.1a, designed to navigate hazardous environments; in inspection tasks, exemplified by Spot (Boston Dynamics Inc. [13]), deployed at Chernobyl to assess radiation levels (Ackerman [14]), as shown in Fig. 2.1b; in agriculture, as seen in the Hydraulically actuated Quadruped (HyQ) robot HyQReal (Semini et al. [15]) from the Italian Institute of Technology (IIT), aiding in the Vinum project for vineyard pruning (Semini and Gatti [16], Guadagna et al. [17]), as demonstrated in Fig. 2.1c; and even in space exploration, where DFKI's CREX robot, in Fig. 2.1d, demonstrates the potential for extraterrestrial missions (DFKI Robotics Innovation Center [18], Dettmann et al. [19]). More recently, the quadruped robot VERO (Amatucci et al. [20]) from the IIT, was employed for autonomous litter collection, specifically targeting cigarette butts, the second most common waste worldwide. Based on the Aliengo Unitree robot [5] equipped with a vacuum cleaner, VERO was successfully tested in scenarios that challenge locomotion and detection skills, including beaches in the city of Genoa, Italy, as shown in Fig. 2.1e (Ackerman [21]). Beyond these, humanoid robots such as the Boston Dynamics' Atlas have showcased exceptional agility and coordination by performing parkour [22] (Fig. 2.1f), while robots as Unitree G1 [23], in Fig. 2.1g and Figure AI's humanoid robot [24], shown in Fig. 2.1h, demonstrate remarkable dexterous capabilities and dynamic movements, highlighting advancements in robotics aimed at human-centric tasks and agile locomotion.

(a) ANYmal by ANYbotics operating in a fire scenario.

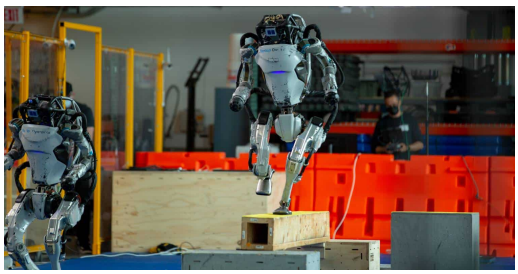


(b) Spot by Boston Dynamics Inc. operating in Chernobyl.

(c) HyQReal by IIT operating in a vineyard in Piacenza, Italy.

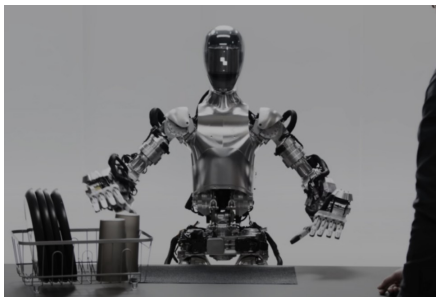
(d) CREX by DFKI Robotics Innovation Center operating in a space-like environment.

(e) VERO robot by IIT in a beach of Vernazzola, Genoa, Italy.



(f) Atlas by Boston Dynamics performing parkour.

(g) G1 by Unitree demonstrating dynamic movements capabilities.



(h) Figure AI's humanoid robot showcasing dexterous capabilities.

Figure 2.1: Legged Robots Performing Real-World and High-Difficulty Tasks.

To autonomously navigate these challenging environments, legged robots must rely on robust and accurate state estimation. The state of a robot refers to its internal and external configuration, encompassing parameters such as position, orientation, velocity, and other relevant factors that describe the robot's motion and behavior over time.

State estimation is a key and fundamental challenge in robotics, as it provides the essential information for stable locomotion, precise navigation, and effective interaction with the environment (Barfoot [25]). In the context of legged robots, accurate and reliable state estimation becomes even more critical due to the complex dynamics involved in locomotion, the need to maintain balance, and the robot's ability to traverse uneven and uncertain terrain. Legged robots experience frequent contact changes with the environment, nonlinear dynamics, and disturbances from various sources. These factors require robust state estimation algorithms that can handle the inherent uncertainties and ensure precise control over the robot's movements. At the heart of this problem lies the integration of sensory information. A legged robot relies on two primary classes of sensors for state estimation: proprioceptive sensors, which provide information about the robot's internal state (e.g., joint angles, motor currents, and inertial measurements), and exteroceptive sensors, which capture data about the external environment (e.g., cameras, LiDARs, and Global Navigation Satellite System (GNSS) receivers). Furthermore, multi-sensor fusion plays a crucial role in combining these sources of data to enhance the robustness and accuracy of state estimation, especially in environments where a single sensor modality may be insufficient or prone to failure.

To address the complexities of state estimation in legged robots, researchers have developed a range of filtering and smoothing techniques. These methods, including Kalman Filters, Particle Filters, and advanced graph-based optimization approaches, are used to process and fuse sensor data, enabling the robot to maintain an accurate estimate of its state across time.

This state-of-the-art review is structured as follows:

- Section 2.1 focuses on the internal sensing modalities and methods that rely on proprioceptive sensors.
- Section 2.2 covers exteroceptive sensors such as cameras, and LiDAR, along with the algorithms that process this data for localization and environmental mapping.
- Section 2.3 explores how proprioceptive and exteroceptive sensors are fused to provide a more comprehensive and accurate state estimate, including the use of filtering and smoothing techniques to handle uncertainties.

2.1 Proprioceptive State Estimation

Most of the modern legged robots are equipped with IMUs, joint encoders, and force/torque sensors (Fig. 2.2) to monitor their internal state and interaction with

(a) P-1775 IMU from KVH Industries [26].

(b) AEDA-3300 Series from Avago Technologies [27].

(c) HEX 6-Axis Force/-Torque Sensor from TechLabs [28].

Figure 2.2: Examples of proprioceptive sensors. From left to right: IMU sensor, encoder, and force/torque sensor.

the environment. These devices provide low-dimensional signals at high rates (250–1000 Hz), making them ideal for real-time state estimation. However, the integration of these sensors poses several challenges, including sensor noise, drift, and the need to model complex dynamics accurately. To address these issues, researchers have developed a range of filtering and estimation techniques tailored to legged robots.

Proprioceptive state estimators rely on kinematic sensors, contact detection, and inertial measurements to estimate the robot's pose, velocity, and contact points. Kinematic sensors measure the internal configuration of the robot and often directly depend on the corresponding state. The typical kinematic sensor is the encoder that measures the joint angles of the robot. Force sensors can be employed to measure internal forces as joint forces or to measure external forces such as contact forces. There are many different types of sensors including optical, resistive, capacitive, or piezoelectric sensors. One difficulty involves calibrating the force sensor, as the lack of a precise reference often requires the manufacturer to carry out the calibration, and it cannot be adjusted afterwards. This is one reason why force sensors represent a less reliable source of information when it comes to state estimation. Consequently, force sensors are often used for estimating the contact state, i.e., whether a foot is in contact with the ground or not, while torques are identified in other ways, e.g., by using the motor currents, as in the Aliengo quadruped robot (Unitree [5]). Inertial sensors, such as accelerometers and gyroscopes, often part of an IMU, provide information about the robot's linear acceleration and angular velocity. These sensors are crucial for estimating the robot's motion and orientation, particularly during dynamic maneuvers and locomotion (Bloesch and Hutter [29]).

In terms of fusion of proprioceptive sensor modalities, most works use Kalman filters because of their simple implementation, low memory usage due to the recursive formula-

tion, and undelayed estimation (i.e. a full update is generated after every step). Another possibility of information fusion is based on Maximum A Posteriori (MAP) optimization over the full data (Barfoot [25], p. 42). This often provides more accurate estimates but can be computationally more expensive and can come with an increased time delay until the estimates are available. There are also many intermediate or combined approaches, such as sliding window estimation approaches (Leutenegger et al. [30]).

One of the first notable works to propose a state estimator tailored for legged robots was done by Bloesch et al. [31]. In this paper, the authors presented a state estimation framework for legged robots that enables the estimation of the robot's complete pose without relying on assumptions about the environment's geometry. The state estimator employs an Observability Constrained Extended Kalman Filter (EKF) that combines kinematic encoder data and onboard IMU measurements. By incorporating the absolute position of all footholds into the filter state, the method accounts for uncertainties due to intermittent ground contact. The filter simultaneously estimates both the footholds' positions and the robot's body pose. Additionally, the framework ensures that the linearized filter maintains the same observability characteristics as the nonlinear system, crucial for accurate state estimation. The approach has been tested in simulation and validated experimentally on the StarlETH (Hutter et al. [32]) quadrupedal robot and tested in short indoor experiments.

Later, Rotella et al. [33] extended this work to humanoid robots. The authors proposed a state estimation framework for humanoid robots that uses only proprioceptive sensors and leg kinematic information. The approach extends previous work on quadrupeds with point feet by incorporating the rotational constraints from the flat feet of humanoid robots. The EKF accommodates contact switching and operates without assumptions about gait or terrain, making it suitable for any humanoid platform. A nonlinear observability analysis showed that adding rotational constraints simplifies singular cases and enhances the system's observability. Tests on a simulated walking dataset demonstrated the performance improvement of the flat-foot filter and confirmed the observability analysis results.

Bloesch et al. [7] proposed a filter that relies entirely on residual-based modeling, offering greater flexibility in handling available information. While related to the Kalman filter, their approach bridges the gap towards batch optimization and incorporates advanced techniques such as robust weighting for outlier rejection. They derived recursive filter equations with computational complexity comparable to the extended information filter, a Kalman filter variant. The effectiveness of the proposed method was demonstrated experimentally on two mobile robotic state estimation problems.

In 2020, Fink and Semini [34] developed a low-level state estimator for quadrupedal robots, focusing on attitude estimation, leg odometry, ground reaction forces, and contact detection. The state estimator consists of three main components. First, a nonlinear observer is used to estimate the robot's attitude by fusing inertial data. This attitude estimator is globally exponentially stable and can handle large initial state errors, making it particularly useful when the robot needs to recover after falling. Unlike traditional EKF, which may diverge in such situations, this observer remains stable. Second, leg odometry is calculated using data from encoders, force sensors, and torque sensors in the robot's joints. Lastly, inertial measurements and leg odometry are fused to estimate the robot's linear position and velocity. The state estimator is validated using data from the HyQ robot (Semini et al. [35]).

Another notable work on proprioceptive state estimation is from Hartley et al. [36]. In this work, the authors propose a contact-aided InEKF, based on Lie group theory and invariant observer design. The filter estimates the robot's pose, velocity, and contact points by combining contact-inertial dynamics with forward kinematic corrections. The error dynamics of the proposed approach follow a log-linear autonomous differential equation, which leads to several important outcomes. The observable state variables can converge with a domain of attraction that does not depend on the robot's trajectory, improving the overall robustness of the system. Unlike the standard EKF, the linearized error dynamics and observation model are independent of the current state estimate, which results in enhanced convergence properties. Furthermore, the local observability matrix remains consistent with the underlying nonlinear system, ensuring more reliable state estimation. The authors also showed how to include IMU biases, handle contact point changes, and formulate world-centric and robot-centric versions of the filter. The InEKF was compared to the commonly used quaternion-based EKF in both simulations and experiments with a Cassie-series bipedal robot [37].

Based on this work, Ramadoss et al. [38] introduced a contact-aided inertial-kinematic floating base state estimation method for humanoid robots, utilizing the concept of matrix Lie groups to represent both the evolution of the state and the observations. The state is represented using a matrix Lie group, which encapsulates the base's position, orientation, and velocity, as well as the positions and orientations of the feet and biases in the IMU. Observations were taken from the relative positions and orientations of the feet using forward kinematics. Due to the choice of uncertainty parametrization, the estimator exhibited rapid convergence even with large initialization errors. The method was experimentally validated on the iCub humanoid robot platform (Metta et al. [39]).

2.1.1 Proprioceptive State Estimation on Difficult Terrains

One of the most crucial pieces of information for a legged robot navigating unstructured and uneven terrains is reliable contact estimation. While many legged robots detect contact using dedicated foot sensors, this approach can be impractical for agile robots operating in the field, as these sensors are prone to deterioration and failure. For this purpose, Camurri et al. [40] proposed a robust leg odometry module that does not rely on contact sensors. Instead, the system infers foot impacts using internal force sensing and incorporates this information to improve the kinematic-inertial state estimation of the robot's body. The results showed that this approach performs comparably to systems equipped with foot sensors. Extensive experiments were conducted over more than an hour using the 90 Kg HyQ quadrupedal robot, performing various gaits to validate the robustness and accuracy of the proposed method.

To address the challenges of unstable terrain, Bloesch et al. [41], introduced a state estimation approach for legged robots that leverages stochastic filtering techniques. The main concept is to utilize information from the kinematic constraints provided by intermittent ground contacts and fuse it with inertial measurements. To achieve this, the authors developed an Unscented Kalman Filter (UKF) based on a well-defined stochastic model. The robustness of the filter was improved by incorporating an outlier rejection mechanism during the update step. The paper also includes a nonlinear observability analysis, simplifying the observability matrix by considering the unique properties of 3D rotations. This analysis showed that – **apart from the global position and yaw angle** – most states are observable, even when only one foot is in contact with the ground. The filter's performance was evaluated on a quadruped robot traversing uneven and slippery terrain.

Dynamic locomotion on unstructured and uneven terrain presents significant challenges for legged robots, especially when dealing with slippery surfaces, where traditional state estimation and control algorithms often struggle due to the no-slip assumption.

In (Jenelten et al. [42]), the author tackled the issue of slip by separately addressing slip detection and recovery. The authors introduced a probabilistic slip estimator using a Hidden Markov Model to detect slip events. For recovery, the paper proposes the use of impedance control and friction modulation as effective strategies to regain stability during traction loss. The proposed estimation and control framework was demonstrated on the ANYmal quadrupedal robot [43], which successfully navigated and walked dynamically over slippery terrain, showing the effectiveness of their approach.

In another work, Wisth et al. [44] introduced a novel factor graph-based approach

for estimating the pose and velocity of a quadrupedal robot on slippery and deformable terrain. The key innovation of the paper is the incorporation of a pre-integrated velocity factor, which fuses leg odometry with additional velocity inputs and accounts for related biases. The authors recognized that uncertainties at the contact points, such as slip, deforming terrain, and leg flexibility, are challenging to model, leading to potential drift in leg odometry. To mitigate this drift, they extended the robot's state vector by adding a bias term for the pre-integrated velocity factor, which can be effectively estimated through the integration of stereo vision and IMU data. The system was validated through experiments involving dynamic movements of the ANYmal robot across loose rocks, slopes, and muddy ground. Results showed a reduction in Relative Pose Error (RPE) compared to previous works and an improvement over state-of-the-art proprioceptive state estimators.

The paper of Fourmy et al. [45] introduced a state estimation approach for legged robots, focusing on the integration of contact force measurements into a factor graph framework. The goal is to improve the accuracy of estimating both the base state (position, orientation, and velocity) and the centroidal state (center of mass position, velocity, and angular momentum) by combining data from multiple sensors, such as IMU, joint encoders, and contact force sensors. The primary innovation in this work is the extension of IMU preintegration techniques, widely used in visual-inertial odometry, to the preintegration of contact force measurements. This allows the estimator to efficiently incorporate high-frequency contact force data into the factor graph, reducing computational load by avoiding the need to process the raw data repeatedly during optimization. By fusing contact force information with the base state estimates, the approach makes the centroidal state observable, which helps mitigate biases caused by inaccuracies in the robot's kinematic and dynamic models.

A study done in (Fahmi et al. [46]) investigated the impact of soft terrain on proprioceptive state estimation. Soft terrains complicate state estimation due to the variability in physical terrain properties. Most state estimation methods are designed for rigid contacts, neglecting the effects of soft terrain. The authors explored how and why soft terrain impacts state estimation, utilizing a state estimator that combines IMU data with leg odometry, originally developed for rigid contact conditions. The HyQ robot was used in experiments, trotting on both soft and rigid terrains. The results demonstrated that soft terrain negatively affects state estimation, causing a significant drift in state estimates compared to rigid terrain, highlighting the need for approaches that account for soft ground properties in legged robots.

Kim et al. [47] introduced a state estimation algorithm for legged robots, framing

the problem as a MAP estimation task and solving it using the Gauss-Newton algorithm. To maintain a fixed problem size, the Schur Complement method was employed for marginalization. The algorithm uses the $SO(3)$ manifold structure to derive the cost function and Jacobians, and the state is reparameterized with a nominal state and variation to ensure the correct application of linear algebra and vector calculus. A slip rejection technique is also included to mitigate errors caused by faulty kinematic models. The proposed algorithm was tested against the InEKF in various real-world environments, demonstrating its effectiveness.

A recent work of Yang et al. [48] has also explored the use of neural networks in state estimation, highlighting their potential to enhance the accuracy of sensor fusion by learning optimal weight functions and improving estimation in challenging environments. The paper focused on state estimation for hydraulic quadruped robots by fusing IMU measurements with leg odometry, applying the InEKF. Furthermore, neural networks were employed to train weight functions for foot force and leg odometry states, which enhanced observation accuracy compared to conventional weighted average methods. Experimental results showed that the proposed method reduced the Root Mean Square Error (RMSE) and decreased Absolute Trajectory Error (ATE) compared to the traditional InEKF, achieving a drift of less than 4 cm per meter traveled.

More recently, Santana et al. [49], introduced a novel InEKF designed specifically for legged robots, which relies solely on proprioceptive sensors. The proposed methodology integrates recent advancements in state estimation theory with robust cost functions applied during the measurement update process. Experimental results on quadrupedal robots, using both real-world tests and public datasets, showed that this approach reduced pose drift by up to 40% over trajectories exceeding 450 m, compared to a state-of-the-art InEKF.

Additionally, Yoon et al. [50] implemented for the first time an invariant smoothing (IS) framework to fuse proprioceptive data for achieving accurate state estimation in legged robots. This paper builds on the previous work by Hartley et al. [36], on contact-aided InEKF. Hartley's approach focused on fusing inertial measurements and leg kinematics while taking into account contact information to enhance the accuracy of state estimation. Yoon et al. [50] extended that concept by proposing an invariant smoother that is built on a MAP formulation of the state estimation problem. This further improves estimation performance, especially under dynamic contact events, by leveraging the group-affine property of residual functions, which results in state-independent Jacobians and better convergence properties during optimization. Unlike previous methods, the formulation with state-independent Jacobians led to enhanced convergence, espe-

cially in dynamic contact scenarios. Additionally, the slip rejection method introduced by Kim et al. [47] and a contact loop model were employed, improving accuracy by re-evaluating foot velocity during dynamic events.

2.2 Exteroceptive State Estimation

Exteroceptive sensors (Fig. 2.3), primarily used for accurate and low-drift pose estimation, have significantly advanced navigation and pose estimation techniques, and they have greatly aided visual odometry, LiDAR odometry, visual Simultaneous Localization And Mapping (SLAM), and LiDAR SLAM.

(a) An example of camera sensor: the depth camera D435 from Intel RealSense [51].

(b) An example of LiDAR sensor: the VLP-16 Mid-range LiDAR sensor from Ouster [52].

Figure 2.3: Examples of exteroceptive sensors. Left: Realsense Depth camera D435. Right: VLP-16 Mid-range LiDAR sensor.

Odometry is crucial for robot navigation, particularly in situations where global positioning methods are unavailable. The main goal of odometry is to predict the robot's motion. Various sensors, such as wheel and leg encoders, cameras, radar, and LiDAR, are used for odometry in robotics (Lee et al. [53]). On the other hand, SLAM is a method used by autonomous systems, such as robots or drones, to build a map of an unknown environment while simultaneously tracking their own location within that map. SLAM systems use sensors, such as cameras, LiDARs, or IMUs, to gather data about the surroundings and their movement. This information is then processed online to generate a map and estimate the robot's pose, allowing it to navigate and understand its environment without a pre-existing map (Al-Tawil et al. [54]).

The main difference between SLAM and pure odometry lies in the scope of their functionality. Odometry is focused on tracking the relative motion of a robot or vehicle, usually by measuring wheel rotations (wheel odometry), leg kinematics (leg odometry), or by using visual or LiDAR data (visual odometry and LiDAR odometry, respectively). It calculates the position incrementally based on previous data, but it does not account

for global consistency and errors that accumulate over time, which can lead to drift. SLAM, on the other hand, not only tracks motion but also constructs a global map. A key feature of SLAM is loop closure, where the system can recognize when it returns to a previously visited location, allowing it to correct drift and improve the accuracy of both the map and the estimated position (Tourani et al. [55]).

Loop closure is a powerful mechanism in SLAM that minimizes long-term drift by detecting previously visited places and adjusting the map or trajectory accordingly. Modern SLAM frameworks often employ real-time loop closure, allowing the map to be refined without causing abrupt motion changes in the robot's local frame. For example, if a system distinguishes between a local "odom" frame and a global "map" frame (as is common in Robot Operating System (ROS) TF [56]), loop closure can alter the map while keeping the robot's locally tracked pose smooth and continuous. However, despite these design considerations, there are still scenarios in which loop closure can pose challenges. In feature-poor environments (e.g., empty rooms) or in locations where loops are not detected for long periods (e.g., prolonged corridors), substantial drift may accumulate before a loop closure event occurs. Although the final result of loop closure generally improves overall accuracy, the system's performance can degrade temporarily until a loop is detected and incorporated. Moreover, in highly ambiguous settings (e.g., places with very similar features), there is a risk of incorrect loop closures, which can lead to larger corrections in the global map and, consequently, more noticeable shifts if the local and global frames are not well managed.

In summary, while loop closure remains essential for reducing drift in SLAM, its effectiveness in real-time applications depends on careful system design. By properly managing local and global reference frames, modern SLAM systems can integrate loop closures smoothly, ensuring minimal disruption to the robot's real-time control and feedback loops.

In the next sections, we describe the most relevant work on Visual and LiDAR odometry, as well as SLAM algorithms, with a focus on the most recent developments and those already applied to legged robots.

2.2.1 Visual Odometry and SLAM

The term "Visual Odometry" was first introduced by Nister et al. [57] for its similarity to the concept of wheel odometry. In this pioneering paper, the authors presented a system designed to estimate the motion of either a stereo head or a single-moving camera using video input. The system generates motion estimates for navigation in real-time

with minimal delay. The process begins with a feature tracker, where point features are matched between consecutive frames and linked into image trajectories at video speed. From these feature tracks, robust camera motion estimates are derived using a geometric hypothesize-and-test framework. This method produced what is referred to as visual odometry-motion estimates based solely on visual input, without requiring any prior knowledge of the scene or the camera's movement. For a complete overview of Visual Odometry methods, we suggest the interested reader to refer to (Scaramuzza and Fraundorfer [58], Fraundorfer and Scaramuzza [59], Cadena et al. [60]).

One early work on vision in SLAM is from Sola et al. [61]. This paper introduced Bi-CamSLAM, an approach that combines the strengths of both monocular vision and stereo vision in SLAM. The method leverages monocular SLAM techniques on a stereo-vision rig, allowing the system to achieve more flexible and robust mapping and localization. The idea is to benefit from the immediate 3D information provided by stereo vision for nearby objects while maintaining the long-range angular accuracy offered by monocular methods for distant landmarks. One of the main advantages of this approach is the ability to rapidly map objects that are close to the robot using stereo vision, which is particularly important for reactive navigation. At the same time, distant landmarks, which may fall outside the effective range of stereo vision, can still be used as long-term angular references through monocular techniques. This combination helps improve the consistency and accuracy of the SLAM process, particularly by reducing angular drift.

Later, Huang et al. [62] presented a system designed for visual odometry and mapping using an Red Green Blue-Depth (RGB-D) camera, specifically applied to autonomous flight. RGB-D cameras offer both color images and per-pixel depth information, making them highly valuable for mobile robotics due to their rich data output and the availability of affordable sensors. The system integrated advancements in algorithms and hardware to enable real-time 3D navigation in cluttered environments, relying solely on onboard sensor data. This independence from external communication links enhanced reliability, particularly in challenging conditions where wireless connections may be unstable. The described system was implemented for a quadrotor micro air vehicle, demonstrating its ability to stabilize and control the vehicle autonomously. It was also used to construct detailed 3D maps of indoor environments. The paper further evaluated the system's performance in stabilizing flight and mapping tasks and discussed limitations, such as sensor noise or challenges in dynamic or low-texture environments.

Among the most notable work on visual odometry, Mur-Artal et al. [63] introduced Oriented FAST and Rotated BRIEF (ORB)-SLAM. ORB-SLAM is a real-time, feature-based monocular SLAM system that performs effectively in various environments, both

large and small, indoors and outdoors. ORB-SLAM is robust to motion clutter and supports wide-baseline loop closure and relocalization. It includes automatic initialization, which simplifies system deployment. The system is designed to use the same ORB features for all essential SLAM tasks: tracking, mapping, relocalization, and loop closure. A key strength of ORB-SLAM is its "survival of the fittest" strategy, which carefully selects points and keyframes for the map reconstruction process. This approach ensures robustness and keeps the map compact, allowing it to grow only when the scene changes, supporting long-term use. The paper provided a comprehensive evaluation of ORB-SLAM across 27 sequences from well-known datasets, where it outperformed other state-of-the-art monocular SLAM systems.

ORB-SLAM has been expanded to ORB-SLAM2 (Mur-Artal and Tardós [64]) and ORB-SLAM3 (Campos et al. [65]). ORB-SLAM2 builds upon the foundation of the original ORB-SLAM by introducing support for monocular, stereo, and RGB-D cameras. It provides robust real-time performance in various environments, from small indoor sequences to large-scale outdoor scenarios. Key features of ORB-SLAM2 include map reuse, loop closure, and relocalization, enabling it to handle long-term operations and large-scale mapping. The back-end of ORB-SLAM2 incorporates bundle adjustment for accurate trajectory estimation with metric scale, and it includes a lightweight localization mode that uses visual odometry in unmapped regions while matching points from existing maps to reduce drift. Its ability to run on a standard Central Processing Unit (CPU) and its accuracy made it a state-of-the-art SLAM system at the time of its release.

ORB-SLAM3 was a direct evolution of ORB-SLAM2. It further advances the system by introducing several enhancements. ORB-SLAM3 is the first SLAM system capable of performing visual, visual-inertial, and multi-map SLAM across monocular, stereo, and RGB-D cameras, with support for both pinhole and fisheye lens models. One of the major innovations in ORB-SLAM3 was its tightly integrated visual-inertial SLAM, which relies entirely on MAP estimation, even during IMU initialization. This resulted in real-time robust operation with higher accuracy, making it 2 to 10 times more accurate than previous approaches. Additionally, ORB-SLAM3 introduced a multi-map system, which allows to create and merge multiple maps seamlessly, improving the performance in environments with poor visual information. Unlike visual odometry systems, ORB-SLAM3 can reuse information from previous keyframes, even if they are separated by long periods or different mapping sessions, significantly boosting accuracy.

Another relevant work on visual odometry is from Forster et al. [66] introduced a semi-direct monocular visual odometry algorithm that is highly precise, robust, and faster than other state-of-the-art methods. By adopting a semi-direct approach, their algorithm

avoids the need for computationally expensive feature extraction and robust matching techniques typically used in motion estimation. Instead, it operates directly on pixel intensities, delivering subpixel precision at high frame rates. For mapping, they employ a probabilistic method that explicitly models outlier measurements, leading to fewer outliers and more reliable 3D point estimates. The algorithm's ability to estimate motion with high precision and at high frame rates enhanced its robustness in environments with limited, repetitive, or high-frequency texture. It has been successfully applied to micro-aerial vehicle state estimation in Global Positioning System (GPS)-denied environments. Running at 55 frames per second on an onboard embedded computer and more than 300 frames per second on a consumer laptop, they call this approach Semi-direct Visual Odometry (SVO).

Most visual SLAM systems struggle in dynamic environments, often relying on deep-learning-based methods to detect and filter moving objects. However, these approaches fail to handle unknown moving objects. To deal with this problem, Abati et al. [67] introduced Panoptic-SLAM, an open-source visual SLAM system designed for robustness in dynamic scenarios, even with unknown objects. It leverages panoptic segmentation to filter dynamic elements during state estimation. Built upon ORB-SLAM3, Panoptic-SLAM was tested on real-world datasets and compared against state-of-the-art systems. Results showed that Panoptic-SLAM is, on average, four times more accurate than PVO (Ye et al. [68]), the latest panoptic-based visual SLAM approach.

Despite the enormous progress in this field, challenges remain in developing large-scale, long-term visual odometry and SLAM systems, such as for autonomous driving over hundreds of miles. Currently, systems that use LiDAR sensors excel in such scenarios due to their high accuracy, robustness, and reliability. For visual odometry to replace these systems, technical improvements in robustness and long-term stability are needed.

2.2.2 LiDAR Odometry and SLAM

LiDAR (Fig. 2.3b), an acronym for Light Detection And Ranging, is a powerful remote sensing technology employed for measuring distances and constructing highly detailed 3D representations of objects and environments. The sensing process commences with a LiDAR system emitting laser pulses toward a designated area. When these pulses encounter obstacles, a portion of the light reflects back to the LiDAR sensor. Measuring the time each laser pulse takes to return and leveraging the constant speed of light, LiDAR calculates the distance to the target.

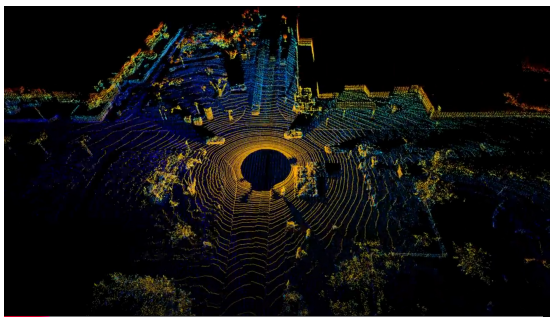
LiDAR-only odometry determines a robot's position by analyzing consecutive LiDAR

scans. In this thesis, LiDAR odometry is classified into two main categories: (1) direct matching, and (2) feature-based matching. The main difference between these two approaches lies in the method used to match points between consecutive scans. Direct matching methods directly calculate the transformation between two consecutive LiDAR scans, while feature-based methods extract feature points in the LiDAR point cloud and use them to estimate the transformation.

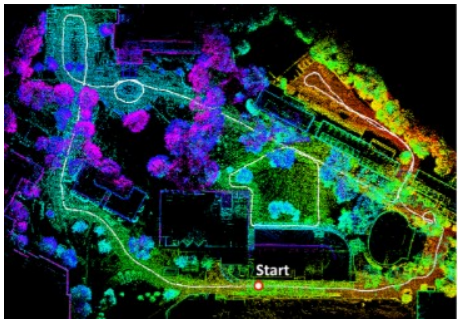
2.2.2.1 Direct matching

The direct matching method directly calculates the transformation between two consecutive LiDAR scans, representing the most straightforward approach in LiDAR-only odometry. The Iterative Closest Point (ICP) algorithm (Zhang [69]) is a commonly used technique for estimating this transformation iteratively by minimizing an error matrix, typically the sum of squared distances between the matched point pairs. Robot odometry is derived by calculating the transformation between each pair of consecutive scans using the ICP algorithm. However, ICP has drawbacks, including susceptibility to local minima, which necessitates a reliable initial guess. The algorithm is also sensitive to noise, such as dynamic objects. Additionally, its iterative nature can result in computational expense, sometimes causing prohibitively slow computation speed. Consequently, substantial efforts have been dedicated to enhancing the performance of the ICP algorithm for improved odometry. For instance, Segal et al. [70] proposed Generalized-ICP, a probabilistic framework that combines both the ICP and point-to-plane ICP algorithms. Unlike the traditional point-to-plane method, which models planar surface structures using only the 'model' (target) scan, the Generalized-ICP approach models planar structures from both (source and target) scans. This effectively transforms the method into a "plane-to-plane" approach, because instead of matching points to planes in just one scan, it aligns planes from both scans. One of the key advantages of this approach is its robustness to incorrect correspondences, which simplifies the tuning of the maximum match distance parameter, a common challenge in ICP variants. Additionally, the Generalized-ICP supports more expressive probabilistic models, allowing for the incorporation of terms to handle outliers and measurement noise, for instance.

However, modern LiDAR sensors produce dense point clouds that can overwhelm traditional odometry algorithms, particularly on computationally limited platforms. To address this, Chen et al. [71] introduced Direct LiDAR Odometry (DLO), a lightweight method that provides consistent and accurate localization while being efficient enough for real-time operation on limited hardware. DLO achieves this efficiency through several key innovations. First, it uses dense, minimally preprocessed point clouds, allowing for



(a) KISS-ICP. Image taken from [72].



(b) LeGO-LOAM. Image taken from [73].

Figure 2.4: Examples of a a direct matching and of a feature-based matching LiDAR odometry algorithm: KISS-ICP (left), and LeGO-LOAM (right) in action.

detailed pose estimation without sacrificing speed. It also implements a novel keyframing system, which effectively manages historical map data, reducing computational load over time. Additionally, DLO incorporates a custom ICP solver designed to accelerate point cloud registration by recycling data structures, further enhancing computational efficiency. The DLO system has been extensively tested in complex environments on both aerial and legged robots, as part of NASA JPL's Team CoSTAR efforts in the DARPA Subterranean Challenge.

More recently, Vizzo et al. [9], presented Keep It Small and Simple (KISS)-ICP (Fig. 2.4a), a simple and robust LiDAR-based odometry system that prioritizes core functionality and efficiency over complexity. Unlike many sensor-based odometry systems that increase complexity to improve ego-motion estimation, this approach focuses on removing non-essential components and refining the fundamental elements, resulting in a streamlined system capable of operating effectively under a variety of environmental conditions and with different LiDAR sensors. The proposed method utilizes point-to-point ICP combined with adaptive thresholding for correspondence matching, along with a robust kernel and a simple motion compensation strategy. Additionally, it incorporates point cloud subsampling to improve efficiency. Remarkably, this system requires only a few parameters, which typically do not need sensor-specific tuning, making it adaptable across different platforms. It has been successfully tested in diverse applications using the same parameter configuration, including automotive platforms, Unmanned Aerial Vehicles (UAVs), Segways, and handheld LiDARs. The system does not rely on IMU data and operates solely on 3D point clouds, making it highly versatile for various operational conditions.

More recently, Ferrari et al. [74] presented MAD-ICP, an algorithm that builds upon the well-established ICP framework. It utilizes a Principal Component Analysis (PCA)-based k-dimensional (k-d)-tree implementation to extract structural information from

the point clouds and calculate the minimization metric for alignment. To manage drift, the system adjusts the local map based on the estimated uncertainty of the tracked pose, ensuring stable performance over time.

2.2.2.2 Feature-based matching

Feature-based approaches in LiDAR-only odometry extract feature points in the LiDAR point cloud and match them to estimate the transformation. Utilizing only feature points instead of the entire point cloud can improve computational speed and overall performance by eliminating outliers such as noise. The main challenge with feature-based methods lies in the selection of "good" feature points that enhance point cloud registration performance.

Zhang et al. [75] presented Lidar Odometry and Mapping (LOAM) a real-time approach for odometry and mapping using range data from a 2-axis LiDAR operating in 6 Degrees of Freedom (DOFs). The range measurements are received at different times, and inaccuracies in motion estimation can lead to misalignments in the resulting point cloud. Traditionally, coherent 3D maps are generated using offline batch methods, often relying on loop closure to correct drift over time. Their approach offers low drift and computational efficiency without requiring highly precise range or inertial measurements. The key to this performance lies in dividing the complex SLAM problem, which typically optimizes many variables at once, into two distinct algorithms. One algorithm runs at a high frequency with lower accuracy to estimate the LiDAR's velocity (odometry), while the second algorithm runs at a lower frequency but with greater precision for point cloud matching and registration. This combination enables real-time mapping.

A development of this work is LeGO-LOAM (Shan and Englot [73]). LeGO-LOAM (Fig. 2.4b) is a lightweight and ground-optimized LiDAR odometry and mapping method designed for real-time 6 DOFs pose estimation in ground vehicles. LeGO-LOAM is considered lightweight because it enables real-time pose estimation on low-power embedded systems. It is also ground-optimized, as it takes advantage of the ground plane during both segmentation and optimization steps. The process begins with point cloud segmentation to reduce noise, followed by feature extraction to identify distinctive planar and edge features. These features are then used in a two-step Levenberg-Marquardt optimization process to solve for different components of the 6-DOFs transformation between consecutive scans. Results demonstrated that LeGO-LOAM achieves comparable or improved accuracy while reducing computational overhead, compared to LOAM.

More recently, Guadagnino et al. [76] introduced a new method that utilizes the intensity channel of 3D LiDAR scans for high-frequency odometry estimation. Unlike

traditional approaches that use full point clouds, their method extracts a sparse set of key points from intensity images using feature extraction architectures originally developed for Red Green Blue (RGB) images. They also proposed a self-supervised fine-tuning process to improve feature extraction accuracy online without needing ground truth data.

The literature shows that visual and LiDAR systems have proven to be accurate and robust in many situations. However, there are challenging scenarios where these systems can fail when used independently. To address this, researchers have explored combining cameras and LiDARs with other sensor modalities to improve performance. The details of this integration and its impact on system performance are further discussed in Section 2.3.

2.3 Multi-Sensor State Estimation

Visual-only and LiDAR-only odometry or SLAM algorithms can struggle to obtain precise measurements in challenging conditions. For instance, visual systems may fail in featureless environments or when visual feedback is unreliable, such as in fog, dust, or darkness. Similarly, LiDAR systems are vulnerable in areas with limited or repetitive geometric features, such as tunnels or highways. Additionally, exteroceptive state estimation methods are limited by the sensor's frequency, leading to delays in obtaining measurements. To mitigate these issues, they are often paired with proprioceptive sensors that provide high-frequency data and ensure faster updates on the robot's measurements. While proprioceptive sensors alone cannot offer low-drift pose estimation (as demonstrated by Bloesch et al. [31]), they can operate effectively in those environments where exteroceptive sensors face challenges, such as in poor lighting, occlusions, or areas lacking significant features.

For example, Bloesch et al. [77], presented a Visual Inertial Odometry (VIO) framework that tightly integrates inertial and visual data from one or more cameras using an Iterated Extended Kalman Filter (IEKF). Instead of relying on traditional feature extraction, it employs image patches as landmark descriptors, incorporating a photometric error into the filter update step. This allows tracking of non-corner features, such as lines, and simplifies the data association process. The robot-centric filter formulation reduces nonlinearity errors and provides undelayed landmark initialization, resulting in a robust and compact solution, effective in low-texture scenes and motion blur.

A monocular Visual Inertial system (VINS) is a system composed of one camera and a low-cost IMU, that provides six DOFs state estimation. VINS-Mono (Fig. 2.5a), presented by Qin et al. [78], integrates IMU and visual data through tightly coupled

(a) VINS-Mono. Image taken from [78].

(b) FAST-LIO2. Image taken from [80].

Figure 2.5: Examples of VINS-Mono (left), a visual-inertial odometry system, and FAST-LIO (right), a LiDAR-inertial odometry system in action.

nonlinear optimization for accurate VIO. It includes loop detection for relocalization and 4-DOFs pose graph optimization to ensure global consistency. The system can reuse and merge maps, supports various applications requiring high-accuracy localization, and has been tested on public datasets and real-world experiments.

Among LiDAR-based methods, Shan et al. [79], introduced a Smoothing and Mapping (SAM) technique, named LiDAR Inertial Odometry (LIO)-SAM, which is tightly-coupled LiDAR-inertial odometry framework designed for accurate, real-time trajectory estimation and map-building. LIO-SAM is based on a factor graph structure, allowing integration of various measurements, including loop closures, for enhanced accuracy. IMU preintegration provides initial motion estimates and deskews point clouds for LiDAR odometry optimization. To maintain real-time performance, the system uses a local scan-matching approach, marginalizing old LiDAR scans and employing a sliding window of keyframes.

FAST-LIO by Xu and Zhang [11] is a computationally efficient and robust LIO framework that tightly fuses LiDAR feature points and IMU data using an IEKF. It handles fast motion and cluttered environments by lowering the computational load with a Kalman gain formula based on state dimension rather than measurement dimension. FAST-LIO2 (Xu et al. [80]) extends this by introducing two key improvements. First, it registers raw LiDAR points directly to the map without feature extraction, increasing accuracy and adaptability to different LiDAR types. Second, it incorporates an incremental k-d data structure for efficient map updates and dynamic rebalancing, surpassing other data structures such as octrees. These innovations make FAST-LIO2 faster, more robust, and versatile, supporting real-time mapping at up to 100 Hz, and ensuring high accuracy across different platforms and environments, including solid-state LiDAR with small fields of view. An example of an application in Fig. 2.5b.

More recently, Chen et al. [8] presented Direct LiDAR Inertial Odometry (DLIO), a

lightweight algorithm that corrects motion distortion through a coarse-to-fine trajectory construction approach. DLIO was designed to address the problem of the negative impact of motion distortions in LiDAR scans, caused by aggressive motions from agile flights or rough terrain. DLIO uses time-parameterized analytical equations for fast, parallelizable point-wise deskewing, which removes distortions caused by sensor movement. By optimizing motion correction and scan registration, DLIO demonstrated better computational efficiency compared to other state-of-the-art methods.

To achieve accurate localization, sometimes GNSS is used since it can provide absolute measurements outdoors and mitigate long-term drift. Fusing GNSS data with other sensors is challenging, particularly when a robot transitions between areas with and without sky visibility. To address this, Beuchert et al. [81] proposed a robust method that tightly integrates raw GNSS receiver data with inertial measurements and optionally LiDAR observations for precise and smooth mobile robot localization. Their approach employs a factor graph incorporating two types of GNSS factors. The first type uses pseudorange measurements, enabling global localization on Earth. The second type uses carrier phase measurements, providing highly accurate relative localization, which is particularly useful when other sensors encounter challenges. The approach was validated on a public urban driving dataset and with data from both a car and a quadruped robot operating in environments with limited sky visibility, such as forests.

The recent Hilti SLAM Challenges (2021–2023) [82] have provided a competitive forum where academic and industrial research teams showcased LiDAR-based SLAM solutions in realistic, large-scale industrial environments. These competitions emphasize robustness to real-world factors such as cluttered workspace, dynamic objects (e.g. moving workers or machinery), and challenging geometries. Among the top-performing teams:

- CSIRO: this teams has demonstrated highly accurate LiDAR-inertial odometry systems based on the work in (Ramezani et al. [83]), which demontrated robust performance in handling large-scale indutrail sites. Their methods often incorporate advanced loop closure strategies and precise motion compensation, contributing to low-drift performance in unsturctured environments.

- KAIST (Urban Robotics Group): this group has produced LiDAR-inertial odometry frameworks using AdaLIO (Lim et al. [84]) as a SLAM frontend and Quatro (Lim et al. [85]) for loop closure detection, with pose graph optimization handled via a factor graph. Their systems have demonstrated high accuracy and robustness in large-scale industrial environments, with a focus on real-time operation and low computational overhead.

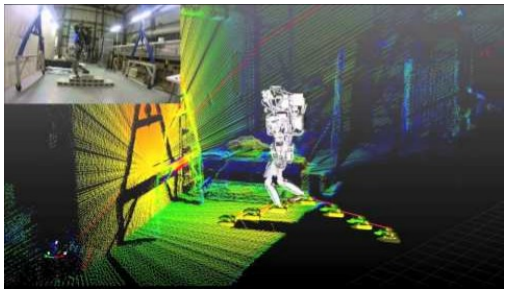
By comparing algorithms on common datasets and standardized metrics, the Hilti SLAM Challenges highlight tangible improvements in drift reduction, real-time performance, and robustness to partial sensor occlusions, dynamic objects, and challenging geometries. These advancements are crucial for deploying LiDAR-based SLAM systems in real-world applications, particularly in industrial settings where safety, efficiency, and reliability are paramount.

However, all of the previously mentioned algorithms are general and platform-agnostic, making them applicable to various types of robots. However, for legged robots, state estimation is also particularly challenging due to the instability of legged locomotion and frequent shifts in terrain contact, which introduce issues such as vibration, abrupt changes in motion, and inconsistencies in sensor readings. Incorporating leg kinematics can provide additional valuable information to complement visual, LiDAR, GNSS, and inertial data, which can significantly enhance the accuracy of state estimation in such dynamic environments. Methodologies specifically designed for legged robots are discussed in the following section (Section 2.3.1).

2.3.1 Multi-Sensor State Estimation for Legged Robots

One of the most famous multi-sensor state estimators for legged robots is Pronto, (Camurri et al. [6]). Pronto is a modular and flexible state estimation framework for legged robots in challenging real-world environments, for instance, with low light, rough terrain, and dynamic obstacles. The core of Pronto is an EKF that fuses IMU and leg odometry data for pose and velocity estimation. It also integrates occasional low-frequency pose corrections from visual and LiDAR odometry. Pronto runs high-frequency proprioceptive estimation (250–1000 Hz) for control loops, and its effectiveness has been demonstrated on multiple-legged platforms, including humanoid and quadruped robots such as Atlas, (Boston Dynamics Inc. [86]) depicted in Fig. 2.6a, Valkirye (NASA [87]), ANYmal (Fankhauser and Hutter [12]) and HyQ (Semini et al. [35]). The algorithms are available as open-source ROS packages.

Lim et al. [3] presented WALK-VIO, a VIO system designed for quadruped robots, incorporating walking-motion-adaptive leg kinematic constraints that adjust based on the robot's body motion. Robots often rely on VIO for fast localization, but challenges arise in outdoor settings, where extraneous features from the sky or ground cause tracking failures, and walking motions introduce wobbling, affecting the accuracy of both cameras and IMU data. Existing approaches use leg kinematic constraints, but WALK-VIO goes further by adapting these constraints dynamically based on other factors such as the



(a) Pronto has been used on the Boston Dynamics Atlas robot, by the MIT DRC team in the DARPA Robotics Challenge [88].

(b) Vilens was tested on several Dataset. This image shows the results obtained from the LSM Dataset of the Urban Circuit of the DARPA SubT Challenge [89].

Figure 2.6: Pronto and **VILENS** are two main examples of multi-sensor state estimators for legged robots.

robot's controller, gait, and speed. This ensures that the VIO remains effective regardless of changes in walking motion.

However, it is important to highlight that both Pronto and WALK-VIO are developed under a no-slip assumption, meaning they assume the robot maintains continuous ground contact without slipping or falling. While this simplifies the state estimation process, it neglects potential slippage or loss of contact, which can severely affect the accuracy and reliability of the estimated state, particularly in challenging terrains where such conditions are likely to occur.

To deal with slippage, Teng et al. [90] introduced a state estimator for legged robots in slippery environments, utilizing an InEKF to fuse inertial, velocity, and leg kinematic data with tracking camera inputs. It models camera-robot misalignment, enabling auto-calibration of the camera pose. Velocity from leg kinematics is treated as a right-invariant observation, and observability analysis confirms the system's consistency, except for rotation around the gravity vector and absolute position in some singular cases. The method is tested on the Cassie bipedal robot (Agility Robotics [37]), walking over slippery terrain, with online noise tuning for variable camera noise.

In another study, named Periodic-SLAM (Kumar et al. [91]), the authors address the challenges of state estimation using visual information on legged robots, particularly due to rapid changes in the camera's viewing angle. They demonstrate that leveraging the structured and periodic nature of legged locomotion can improve the accuracy of visual-inertial SLAM in these difficult scenarios. Their method capitalizes on the predictability of the robot's gait cycle to enhance the feature tracking module. By performing multi-session SLAM, their approach reduces absolute trajectory error and outperforms state-of-the-art SLAM methods in both simulated environments and real-world tests on dynamic

quadrupedal gaits.

Another work by Kim et al. [4] proposed a state estimator for legged robots, called STEP, which introduces a pre-integrated foot velocity factor that does not depend on the traditional non-slip assumption. Instead, STEP makes the end-effector velocity observable by leveraging body velocity data from a stereo camera, enabling accurate estimation of the end-effector's pose. Furthermore, STEP eliminates the need for contact detection, a common requirement in other approaches. The method is validated through extensive simulations and real-world experiments in challenging environments, such as uneven and slippery terrains, showcasing robustness and adaptability.

Yang et al. [92] presented Cerberus, an open-source Visual Inertial Leg Odometry (VILO) system designed for real-time state estimation in legged robots. Cerberus utilizes standard sensors, including stereo cameras, IMU, joint encoders, and contact sensors, to estimate precise position across various terrains. A key feature is its online kinematic parameter calibration and outlier rejection, which significantly reduce position drift, achieving less than 1% drift during long-distance, high-speed locomotion.

However, it is important to note that these state estimators are highly reliant on camera inputs, which can sometimes become unreliable. This dependence may affect the overall accuracy and robustness of the estimated states, particularly in environments where visual data quality is compromised, such as in poor lighting, occlusions, or rapid motion changes.

For this reason, other state estimators such as VILENS, introduced by Wisth et al. [2] (Fig. 2.6b), proposed to use also LiDAR in combination with proprioception and camera data. Specifically, VILENS is an odometry system for legged robots that tightly fuses data from four sensor modalities: vision, LiDAR, IMU, and leg odometry, using a factor graph-based framework. A key feature is introducing a linear velocity bias term to address leg odometry drift, which is estimated online due to the tight fusion of pre-integrated velocity factors with the other sensor inputs. This approach makes the linear velocity bias observable and corrects for degenerate conditions that would otherwise impact state estimation when relying on individual sensors.

A recent paper by Ou et al. [93] presented a Kinematic Inertial Leg Odometry (KILO) framework for legged robots, named Leg-KILO, addressing challenges from high-dynamic motion, such as foot impacts causing IMU degradation and LiDAR distortion. Using graph optimization, the framework tightly integrates leg odometry, LiDAR odometry, and loop closure. It features a kinematic-inertial odometry method based on an error-state Kalman filter to reduce height fluctuations and an adaptive scan slicing and splicing technique for LiDAR data. Experiments in various environments show that Leg-KILO

with loop-closure significantly reduces drift during high-dynamic motion compared to state-of-the-art methods, and the dataset and code are open-sourced.

2.4 Summary and Discussion

In this chapter, we reviewed the latest advancements in state estimation for legged robots, focusing on the use of proprioceptive and exteroceptive sensors. Key works in each category were discussed, along with their associated challenges and limitations. We identified the drawbacks of relying solely on either proprioceptive or exteroceptive sensors, such as drift accumulation, and dependence on feature-rich environments. Additionally, we highlighted the most recent developments in multi-sensor state estimation, emphasizing the critical role of integrating diverse sensor modalities to enhance accuracy and robustness in complex and unstructured environments. For legged robots, we showed evidence in the literature that leg kinematics provide valuable additional information to improve state estimation robustness and accuracy. However, these robots often operate in challenging and unpredictable terrains, where environmental uncertainties, such as uneven surfaces or slippery terrain, pose significant challenges to reliable state estimation.

A recent survey of the DARPA Subterranean Challenge [89] by Ebadi et al. [94] clears up on the unique difficulties of deploying state estimation in extreme, GPS-denied, and visually degraded environments. The survey highlights the critical need for robust sensor fusion, robustness to perceptual failures, and the incorporation of non-visual information. Teams participating in the challenge often encountered severe dust, darkness, and uneven terrain, prompting the development of approaches that combined LiDAR, inertial, thermal, and foot-contact data to mitigate slip events and compensate for drift. These experiences underscore the importance of integrating proprioceptive and exteroceptive sensing when navigating challenging domains, reaffirming that leg kinematics and contact forces can significantly enhance the accuracy and robustness of state estimation in slippery or uneven environments.

Over the past decades, there has also been significant progress across visual-inertial, LiDAR-based, and leg-odometry approaches for legged robots. Modern LiDAR odometry pipelines, for instance, can achieve drift rates as low as 1% over long distances when paired with high-quality motion deskewing and calibration. Likewise, visual-inertial odometry methods have substantially improved in robustness and computational efficiency, thanks to better feature detection, Graphics processing unit (GPU)-accelerated SLAM frameworks, and learned semantic cues. Nevertheless, real-world complexities such as sensor occlusion, dynamic lighting, and unsturctured terrain continue to present

major challenges. This reality underscore the ongoing need for research in multi-sensor fusion, slip detection, and fault-tolerant algorithms that can ensure reliable performance despite environmental uncertainties.

The primary goal of this dissertation is to develop state estimation frameworks capable of addressing some of these uncertainties and delivering accurate and robust state estimation for legged robots. Specifically, our work focuses on the challenges posed by slippery terrains, which frequently disrupt state estimation processes. In the next chapter, we introduce a study dedicated to detecting slippage events. The subsequent chapter builds upon this by presenting state estimation frameworks designed to handle such events effectively, leveraging multi-sensor fusion to ensure accurate and reliable performance in diverse and challenging environments.

Chapter 3

Slip Detection on Quadruped Robots

3.1 Preface

This chapter addresses the problem of slip detection, a critical component in mitigating the drift caused by uncertainties on slippery terrains during state estimation. We present a slip detection algorithm that operates independently of gait type and does not rely on position or velocity estimations in the inertial frame, which are prone to drift. The method can detect multiple simultaneous foot slippages by leveraging measurements expressed in a non-inertial frame. The approach was validated on the 90 kg HyQ robot, developed by the Italian Institute of Technology (IIT), Genoa.

This work was primarily carried out during my Master's Thesis internship at IIT while I was a student at the University of Pisa and was later finalized during my Ph.D. studies at IIT. Consequently, some of the images included in this chapter also appear in my Master's Thesis. This study serves as a foundational step for the research presented in the following chapters and has been published in (Nisticò et al. [1]):

Ylenia Nisticò, Shamel Fahmi, Lucia Pallottino, Claudio Semini, and Geoff Fink, "*On Slip Detection for Quadruped Robots*", *Sensors*, vol. 22, no. 8, p. 2967, April 2022, https://doi.org/10.3390/s22082967.

This work was conceptualized and developed by me, supported by Shamel Fahmi, and Geoff Fink, who also contributed to the methodology and investigation. I handled the software development, validation, formal analysis, data curation, and visualization. The initial draft was prepared by me, with all authors contributing to review and editing. Supervision was provided by Lucia Pallottino, Claudio Semini, and Geoff Fink.

3.2 Introduction

Sensor-based slip detection methods, such as those proposed by Park et al. [95], Okatani and Shimoyama [96], Massalim et al. [97], have limited applicability for real-world legged robots because they require sensors attached to the foot tip, which are prone to damage from repetitive impacts during locomotion. Additionally, touchdown events can introduce force signal discontinuities, complicating detection. In contrast, kinematics-based detection strategies are more suitable for legged robots, where ground impacts are frequent.

Several prior works have explored slip detection and recovery. Takemura et al. [98] proposed a dual strategy: adjusting gait parameters as a long-term measure and adding forces to keep the Ground Reaction Force (GRF) within the friction cone as a short-term solution, assuming accurate normal force estimation.

For the bipedal robot HRP–2 [99], Kaneko et al. [100] developed a slip observer to detect skids on slippery floors, enabling balance control by adjusting footholds to compensate for torso rotation.

Jenelten et al. [42] developed a probabilistic contact and slip estimation approach using a Hidden Markov Model for ANYmal, tested on frozen ground. Their slip recovery strategy involved impedance control and friction modulation, demonstrating effective stabilization in field tests.

Focchi et al. [101] introduced a slip detection and friction parameter estimation methodology using proprioceptive sensors, coupled with a recovery strategy leveraging a whole-body controller optimized for Ground Reaction Forces (GRFs). This method, implemented for HyQ locomotion, is further detailed in Section 3.8 as it serves as a baseline for the algorithm presented in this chapter.

3.3 Contribution

Previous works on slip detection have various limitations that impact their robustness and adaptability. For instance, Takemura et al. [98] relied on specific sensors, such as accelerometers attached to the leg, to detect slip. These sensors are prone to deterio-

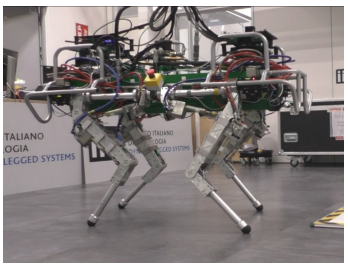
ration due to the continuous ground contact experienced by legged robots, which can degrade performance over time.

The work by Jenelten et al. [42] assumes that the ground conditions, such as friction coefficients, remain relatively stable during locomotion. If the surface properties change rapidly or are highly variable (e.g., mixed patches of ice and gravel), the model may struggle to adapt quickly, resulting in delayed or incorrect slip detection.

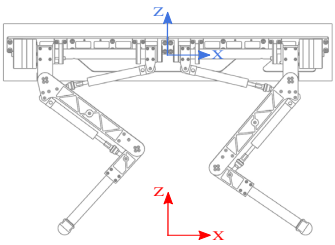
In the work by Focchi et al. [101], slip detection relies on estimating the robot's states in an inertial (world) frame under the assumption that the foot velocity remains constant in this frame. Although this approach can effectively identify slip when state estimates are accurate, even minor errors in state estimation can trigger false positives. Furthermore, because the method integrates IMU-measured accelerations, it is susceptible to drift and divergence over time, which can undermine the overall reliability of slip detection.

The approach presented in Kaneko et al. [100] introduces a slip detection observer that estimates slip based on foot-ground contact forces and kinematic data. However, this method has limitations as it assumes consistent contact forces and may not perform well when there is dynamic variability in the contact interactions or when the robot's gait is highly dynamic. Moreover, the method can struggle on complex terrains with mixed friction conditions, and its performance is influenced by the accuracy of the force sensors, which are susceptible to noise and calibration issues.

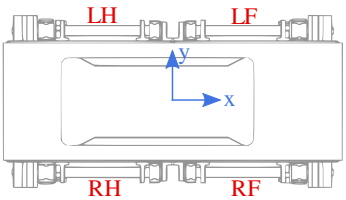
While it is true that slip physically occurs in the world frame (i.e., the foot moves relative to the ground), our approach reduces reliance on potentially drift-prone inertial estimates by tracking foot velocity and position in the robot's base frame. When a foot is commanded to be stationary relative to the ground, we expect minimal relative motion between that foot and the base; any unexpected motion measured in the base frame can then be interpreted as slip, even without directly referencing the world frame. This kinematic-based algorithm is independent of gait type, enabling slip detection for one or more legs simultaneously. Such flexibility allows our method to handle both trot gait (two legs swinging simultaneously) and crawling gait (one leg swinging at a time). We validated our approach on the 90 kg hydraulically actuated quadruped robot HyQ and compared it against the slip detection algorithm by Focchi et al. [101]. Our results demonstrate improved robustness and adaptability across varying terrain conditions and gait patterns, illustrating that reliable slip detection can be achieved by focusing on local (base-frame) measurements rather than complete inertial-frame estimates.



(a) The HyQ robot.



(b) Location of the base frame $\mathcal{B}$ (blue), and the world frame $\mathcal{W}$ (red).



(c) LF, RF, LH, and RH are the left front, right front, left hind, and right hind legs, respectively.

Figure 3.1: The HyQ robot and the reference frames used in this work.

3.4 Outline

The rest of this chapter is organized as follows: Section 3.5 presents the robot model and the sensors used for slip detection. Section 3.7 gives an overview of the baseline approach used for comparison, Section 3.8 details the slip detection algorithm, which is validated on the HyQ robot in Section 3.9. Finally, Section 3.10 provides a discussion, and Section 3.11 concludes the chapter.

3.5 Modelling and Sensing

HyQ (Semini et al. [35]), shown in Fig. 3.1a, is a 90 kg hydraulically actuated quadruped robot developed by IIT. It features a torso and four identical legs arranged in a forward/backward configuration, with knees pointing inward. The robot has 12 torque-controlled joints powered by hydraulic actuators, providing 12 DOFs. The base frame $\mathcal{B}$ is located at the geometric center of the trunk, while the world-inertial frame $\mathcal{W}$ initially aligned with the base frame, considering an offset along the z-axis by the robot's height, as depicted in Figs. 3.1b and 3.1c. We refer to the legs as Left Front (LF), Right Front (RF), Left Hind (LH), and Right Hind (RH).

HyQ is equipped with a six-axis KVH 1750 IMU (KVH Industries [102]) on the trunk, which provides linear acceleration $\mathbf{a} \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$. Each joint has an encoder and torque sensor, allowing measurements of joint positions $\mathbf{q} \in \mathbb{R}^{12}$, velocities $\dot{\mathbf{q}} \in \mathbb{R}^{12}$, and torques $\boldsymbol{\tau} \in \mathbb{R}^{12}$. Measurement noise and bias are assumed to vary slowly over time, with zero mean and a Gaussian distribution. This work uses only the data from the mentioned proprioceptive sensors.

3.6 Contact Estimation

To determine the contact states $\boldsymbol{\alpha} \in \mathbb{R}^4$ (i.e., whether a foot is in contact with the ground), GRFs are first estimated as described in (Fahmi et al. [46]), assuming that all external forces are applied to the feet during the *stance* phase. The contact state $\boldsymbol{\alpha}_i$ for each leg i is a boolean variable, set to 1 when the GRFs exceed a predefined threshold, and 0 otherwise. Specifically, the contact state $\boldsymbol{\alpha}_i$ is defined as:

$$\boldsymbol{\alpha}_i = \begin{cases} 1 & \text{if } \mathbf{F}_{\text{grf},i} > \mathbf{F}_{\text{min}} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where $\mathbf{F}_{\text{min}}$ is the threshold value, and $\mathbf{F}_{\text{grf},i} \in \mathbb{R}^3$ is the GRF of the leg i, computed using the dynamics equation of motion:

$$\mathbf{M}(\bar{\mathbf{x}})\ddot{\bar{\mathbf{x}}} + \mathbf{h}(\bar{\mathbf{x}}, \dot{\bar{\mathbf{x}}}) = \bar{\boldsymbol{\tau}} + \mathbf{J}^\top \mathbf{F}_{\text{grf}} \tag{3.2}$$

where $\bar{\mathbf{x}} = [\mathbf{x}^\top \boldsymbol{\eta}^\top \mathbf{q}^\top]^\top \in \mathbb{R}^{18}$ is the generalized robot state, given by the position and attitude of the base in the world frame, and the joint angles. Then $\dot{\bar{\mathbf{x}}} \in \mathbb{R}^{18}$ and $\ddot{\bar{\mathbf{x}}} \in \mathbb{R}^{18}$ are the corresponding generalized velocities and accelerations, $\mathbf{M} \in \mathbb{R}^{18\times18}$ is the joint-space inertia matrix, $\mathbf{h} \in \mathbb{R}^{18}$ is the vector of Coriolis, centrifugal, and gravity forces, $\bar{\tau} = [\underline{0}\ \tau] \in \mathbb{R}^{18}$ where $\tau \in \mathbb{R}^{12}$ is the vector of joint torques, and finally $\mathbf{F}_{\text{grf}} \in \mathbb{R}^{12}$ is the vector of GRFs, while $\mathbf{J} \in \mathbb{R}^{18\times12}$ is the floating base Jacobian.

Then, we solve for the GRFs F_{grf} of each leg using the actuated part of the dynamics:

$$\mathbf{F}_{\text{grf},i} = -\boldsymbol{\alpha}_i(\mathbf{J}_i^\top(\mathbf{q}_i))^{-1}(\boldsymbol{\tau}_i - \mathbf{h}_i(\bar{\mathbf{x}}_i, \dot{\bar{\mathbf{x}}}_i)) \tag{3.3}$$

3.7 Baseline Approach

In this section, we introduce a state-of-the-art algorithm by Focchi et al. [101], which serves as the baseline for comparison. After discussing its limitations, we present our proposed approach.

The baseline algorithm uses two kinematics-based strategies for slip detection at the foot-velocity level. The first strategy detects slippage in a single leg (Section 3.7.1) by estimating the stance foot velocities in the body frame ($\dot{\mathbf{x}}_f^b$). The second strategy detects slippage involving two or more legs by estimating the stance foot velocities in the world frame ($\dot{\mathbf{x}}_f^w$) (Section 3.7.2).

3.7.1 Single-Leg Slip Detection

The first strategy compares the stance foot velocities in the body frame ($\dot{\mathbf{x}}_f^b$) to identify outliers using statistical tools. At each iteration of the control loop, the median norm of the stance foot velocities is computed. A leg is flagged as slipping if its velocity deviates from the median of the velocity of stance-feet by more than a predefined threshold ϵ, which is experimentally tuned. Each leg has an associated flag that is set to *true* when a slip is detected.

3.7.2 Multiple-Leg Slip Detection

A more complex situation arises when two or more legs are slipping simultaneously, making it difficult to identify which legs are slipping or only in the stance phase using the median approach. To address this, the authors proposed a second strategy: checking which foot velocities $\dot{\mathbf{x}}_f$ are kinematically consistent with the base velocity $\dot{\mathbf{x}}_b^b$.

The intuitive idea is to verify that the Cartesian velocities of the stance feet $\dot{\mathbf{x}}_f^w$ are zero in the inertial frame $\mathcal{W}$. These velocities can be expressed as:

$$0 \approx \dot{\mathbf{x}}_f^w = \dot{\mathbf{x}}_b^w + \mathbf{R}_b^w(\dot{\mathbf{x}}_f^b + \boldsymbol{\omega} \times \mathbf{x}_f^b) \tag{3.4}$$

where $\mathbf{R}_b^w \in SO(3)$ is the rotation matrix representing the base orientation (from $\mathcal{B}$ to $\mathcal{W}$). Here, $\mathbf{x}_f^b \in \mathbb{R}^3$, $\dot{\mathbf{x}}_f^b \in \mathbb{R}^3$, are the foot position and velocity in the body frame $\mathcal{B}$, respectively, while $\boldsymbol{\omega}$ is the angular velocity, measured by an on-board IMU sensor. The base linear velocity $\dot{\mathbf{x}}_b^w \in \mathbb{R}^3$ in the inertial frame is estimated via short-time integration of the base linear acceleration measured by the IMU accelerometers.

3.7.3 Drawbacks of the Baseline Approach

Although the slip detection approaches previously described can exhibit certain limitations, several successful legged robot estimators also rely on velocities expressed in the world frame and IMU integration. These methods have demonstrated robust performance in real-world scenarios when paired with carefully designed state estimation frameworks and calibration procedures that mitigate drift and false positives. Nonetheless, potential challenges remain. For instance, the single-leg slip detection method may not generalize from slow crawl gaits to faster gaits such as trotting, where legs exhibit pairwise velocity differences which can reduce the accuracy of the single-leg approach. Similarly, integrating IMU-measured accelerations in the world frame can be prone to

drift over longer periods, necessitating strategies for managing accumulated errors, such as careful *short-time* integration.

3.8 Proposed Slip Detection Algorithm

The idea behind the proposed method is to overcome the aforementioned problems by detecting slippage using foot velocities expressed in $\mathcal{B}$. This approach, based on $\dot{\mathbf{x}}_f^b = [\dot{\mathbf{x}}_{f_x}^b \dot{\mathbf{x}}_{f_y}^b \dot{\mathbf{x}}_{f_z}^b]^\top \in \mathbb{R}^3$, is more robust as it directly relies on sensor measurements (e.g., encoders) and avoids issues such as drift typically associated with estimating base states in the world frame or using numerical integration of IMU data.

A slipping leg could be identified when the foot velocity deviates from the desired velocity. The deviation is quantified by $\Delta V = \left| {}_d\dot{\mathbf{x}}_f^b - \dot{\mathbf{x}}_f^b \right|$, which represents the norm of the difference between the desired and actual foot velocities in the body frame $\mathcal{B}$.

Slippage can be detected when ΔV exceeds a threshold ϵ during the stance phase. However, calculating the norm alone is not a reliable tool for slip detection. The reason is that the difference between the desired and actual velocity increases along the predominant direction of motion due to larger tracking errors. Additionally, if the robot's velocity changes during motion, ϵ must be adjusted accordingly. To ensure that the direction and velocity of motion do not affect ΔV, we introduce a weight to scale each component of the vector ${}_d\dot{\mathbf{x}}_f^b - \dot{\mathbf{x}}_f^b$. The weight is $\left| {}_d\dot{\mathbf{x}}_{f_i}^b \right|$, the norm of the desired foot velocity in $\mathcal{B}$ for each component. This scaling minimizes the impact of larger tracking errors in any specific direction and ensures that $\overline{\Delta V}$ remains consistent during motion, even if the desired body velocity changes. This effect is illustrated in Fig. 3.2. The top plot shows how ΔV changes when the robot's velocity is modified during a trotting task with a variable feed rate (velocity increases along the x-direction in the interval $[10 - 20]$ seconds). The bottom plot demonstrates that $\overline{\Delta V}$ maintains a consistent trend throughout the task.

To ensure numerical stability and avoid a zero denominator, we divide by $\left| {}_d\dot{\mathbf{x}}_{f_i}^b \right|$ and add a margin m, which is experimentally tuned. When slippage occurs, the value of $\overline{\Delta V}$ increases. Therefore, an upper limit is imposed on $\overline{\Delta V}$, and any further increase beyond this limit is classified as slippage. The final formulation for ΔV is given by:

$$\overline{\Delta V} = \sqrt{\sum_{i=x,y,z} \left(\frac{{}_d\dot{\mathbf{x}}_{f_i}^b - \dot{\mathbf{x}}_{f_i}^b}{\left| {}_d\dot{\mathbf{x}}_{f_i}^b \right| + m} \right)^2} > \epsilon_v \tag{3.5}$$

where the threshold ϵ_v is a *conditioned* value that depends on the phase of foot motion.

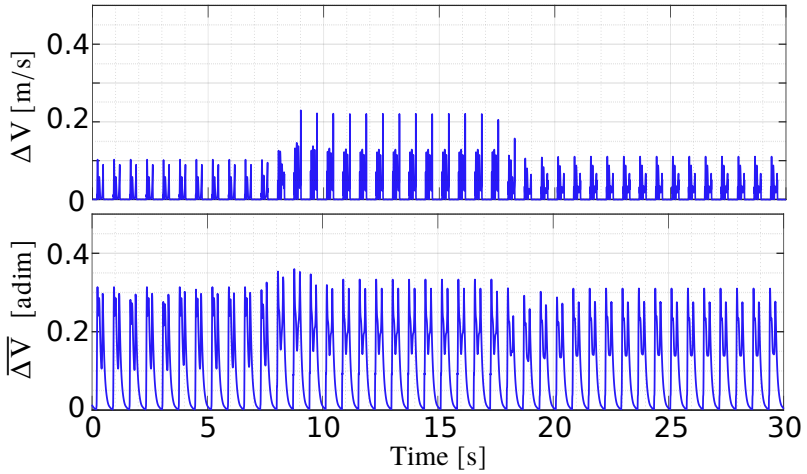


Figure 3.2: ΔV (**top**) and $\overline{\Delta V}$ (**bottom**) in a simple trotting task with variable feed rate (no slippage). The higher peaks in the top figure correspond to a higher linear velocity along the x-axis.

During the *swing phase*, ϵ_v is set to ∞, as a swing leg cannot slip. During the *stance phase*, ϵ_v is assigned a constant value for each leg. This value is carefully tuned to detect the highest peaks of $\overline{\Delta V}$.

In the case of slippage, an important question arises: *How far did we slip?* A quantitative measure of the slipping length can be derived from the foot *position*. During slippage, the foot position deviates from the desired position, and this deviation can be quantified as: $\Delta P = \left| ||_d\mathbf{x}^b_{f_i}|| - ||\mathbf{x}^b_{f_i}|| \right|$. Additionally, ΔP is particularly important for another reason: at the beginning of foot-ground contact, there is a short time interval where the difference between the desired and actual foot velocities increases instantaneously, as shown by the dotted circles in Fig. 3.3. This may be caused by several factors: (i) an actual but small (possibly insignificant) slippage, (ii) a delay in control, or (iii) the current implementation of stance detection (Section 3.6), which introduces a slight delay in recognizing contact with the ground. This initial foot slippage (typically ~ 1 or 2 cm) is negligible compared to the robot's size. To ensure it is not detected as significant, we add a condition on ΔP to account for this negligible slippage:

$$\Delta P = \left| ||_d\mathbf{x}^b_{f_i}|| - ||\mathbf{x}^b_{f_i}|| \right| > \epsilon_p \tag{3.6}$$

If ΔP remains below the threshold ϵ_p, the slippage is considered negligible. The value of ϵ_p is constant and determined experimentally. A slippage is detected when $\overline{\Delta V}$ and ΔP exceed their respective thresholds. For each leg i, we introduce a flag $\beta_i \in 0, 1$, where $\beta_i = 1$ indicates a slip detection, and $\beta_i = 0$ otherwise.

The pseudo-code implementation of the proposed algorithm is in Algorithm 1.

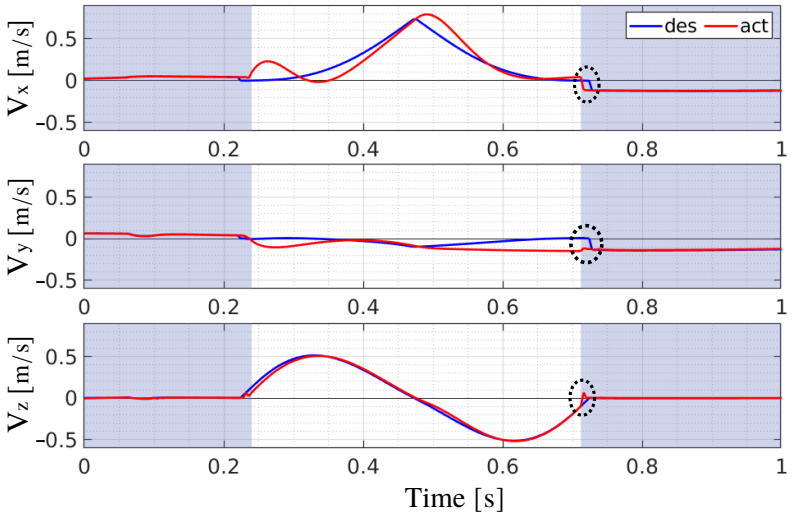


Figure 3.3: Desired (blue line) and actual (red line) foot velocity in a simulation task of crawling (no slippage). Velocity $\mathbf{v} = [\mathbf{v}_x\ \mathbf{v}_y\ \mathbf{v}_z]$ is expressed wrt $\mathcal{B}$. The shaded areas indicate that the foot is in stance.

Algorithm 1 detectSlippage $(\dot{\mathbf{x}}^b_{f_i}, {}_d\dot{\mathbf{x}}^b_{f_i}, \mathbf{x}^b_{f_i}, {}_d\mathbf{x}^b_{f_i})$

1: $\overline{\Delta V} \leftarrow$ scaled diff$(\dot{\mathbf{x}}^b_{f_i}, {}_d\dot{\mathbf{x}}^b_{f_i})$; ▷ $\dot{\mathbf{x}}^b_{f_i}$ and ${}_d\dot{\mathbf{x}}^b_{f_i}$ are the *actual* and *desired* foot velocity in $\mathcal{B}$
2: $\Delta P \leftarrow$ diff$(\mathbf{x}^b_{f_i}, {}_d\mathbf{x}^b_{f_i})$; ▷ $\mathbf{x}^b_{f_i}$ and ${}_d\mathbf{x}^b_{f_i}$ are the *actual* and *desired* foot position in $\mathcal{B}$
3: **for** each stance leg *i* **do**
4: $\beta_i \leftarrow (\overline{\Delta V_i} > \epsilon_v)\ \&\ (\Delta P_i > \epsilon_p)$; ▷ ϵ_v and ϵ_p are the thresholds for $\overline{\Delta V}$ and ΔP
5: **end**

3.9 Results

In this section, we first present the results obtained in simulation with the HyQ robot trotting on a patch of ice (Section 3.9.1), and compare the proposed method with the baseline approach. Then we show experimental results obtained while crawling on a slippery surface (Section 3.9.2). Also in this case we benchmark the proposed method with the baseline approach.

3.9.1 Simulation Results: Trotting onto Patches of Ice

To demonstrate the versatility of the proposed method across different gaits, we tested it in simulation with the robot trotting on terrain featuring four low-friction patches ($\mu = 0.08$), as shown in Fig. 3.4.

We used the following parameters: $\epsilon_v = \text{percentile}(\overline{\Delta V}, 95\%)$, $\epsilon_p = 0.03$, and $m = 0.3$. For comparison, we implemented the baseline strategy described in Section 3.7.2 to

detect multiple legs slipping simultaneously, setting the baseline threshold to $\epsilon_{v_{BL}} = 1$.

Fig. 3.5 illustrates $\overline{\Delta V}$ and ΔP when the robot trots over ice slabs, with shaded red areas indicating actual slippage. Fig. 3.6 compares the slip detection flags obtained using the baseline and proposed approaches. The results show that correctly identifying slipping stance legs does not impact the detection status of other stance legs, as their flags remain 0 due to ΔP staying below ϵ_p. Furthermore, the proposed method demonstrates improved efficiency in detecting slippage for each foot throughout its entire duration.

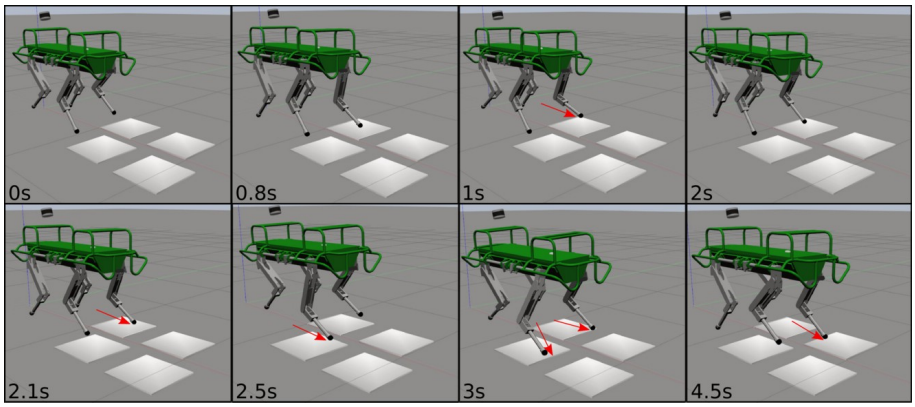


Figure 3.4: Simulation of the HyQ robot traversing slippery terrain patches with a trot gait (low friction patches illustrated in white). The image sequence starts from the top left to right and continues at the bottom left. The red arrows indicate slipping feet.

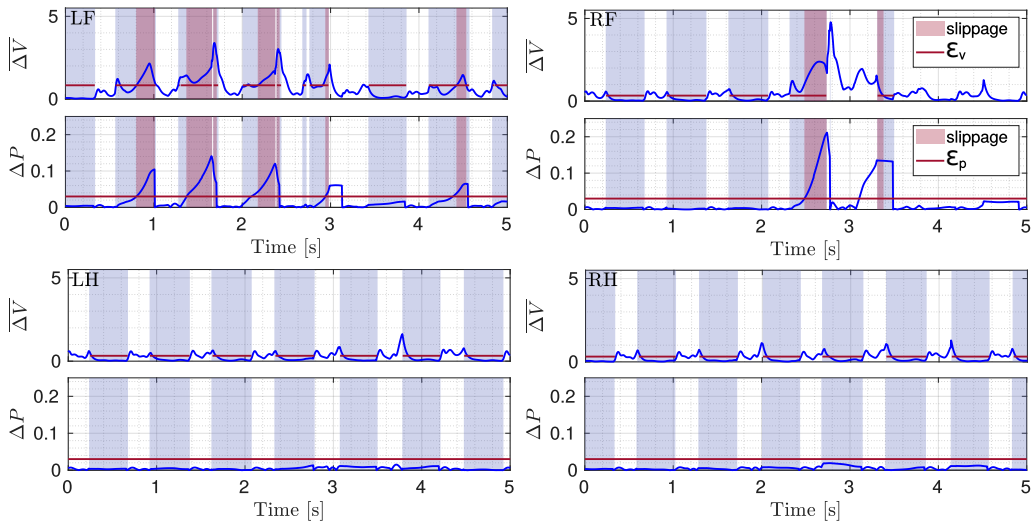


Figure 3.5: Plots of $\overline{\Delta V}$ and ΔP (blue) with respective thresholds (red line) illustrated for the four legs (LF, RF, LH, RH) during a trot gait. The gray shaded area shows stance phases and the red shaded area marks slippage.

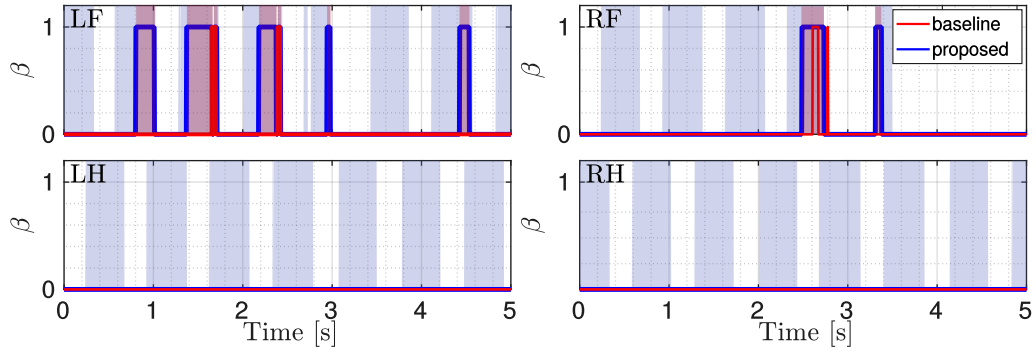


Figure 3.6: Comparison between the flags. The red one is obtained with the baseline approach, the blue one with the proposed approach. The gray shaded area shows stance phases and the red shaded area indicates the ground truth.

3.9.2 Experimental Results on the HyQ robot: Crawling on a Slippery Surface

In this section, we present experimental results on HyQ. The first experiment involved the robot walking on non-slippery terrain, while the second used a slippery patch created by sprinkling soap on its surface. Both experiments were conducted using the crawl gait.

Data from these experiments were recorded and analyzed offline to tune the thresholds. We established the ground truth by identifying atypical foot trajectory displacements during the stance phase and confirming them with the corresponding time stamps and video recordings. Then, we determined the minimum ϵ_p and ϵ_v values that avoided detecting slippage in the first experiment while correctly identifying actual slips in the second.

After the first experiment, we selected the following parameters: $\epsilon_p = 0.04$, $m = 0.3$, and $\epsilon_v = \text{percentile}(\overline{\Delta V}, 95\%)$. During the test, the LH and RF legs slipped simultaneously. For comparison, we implemented the baseline *multiple leg slip detection* method, setting the threshold for the baseline approach to $\epsilon_{v_{BL}} = 1$.

Although only the LF and RF legs were intended to slip during the experiment, slippage also occurred for the RH leg, leaving the LH leg as the only non-slipping leg. Fig. 3.8 shows $\overline{\Delta V}$ and ΔP for all four legs during the slippery terrain experiment depicted in Fig. 3.7. Fig. 3.9 compares the flags obtained using the two methods.

As shown in Fig. 3.8 and Fig. 3.9, all detections using the proposed method are accurate. The results demonstrate that the proposed method outperforms the baseline by correctly detecting actual slipping events, avoiding false positives, and accurately indicating the duration of the slips. In contrast, the baseline approach produces false positives, detecting slippage for the LH leg, which did not actually slip.

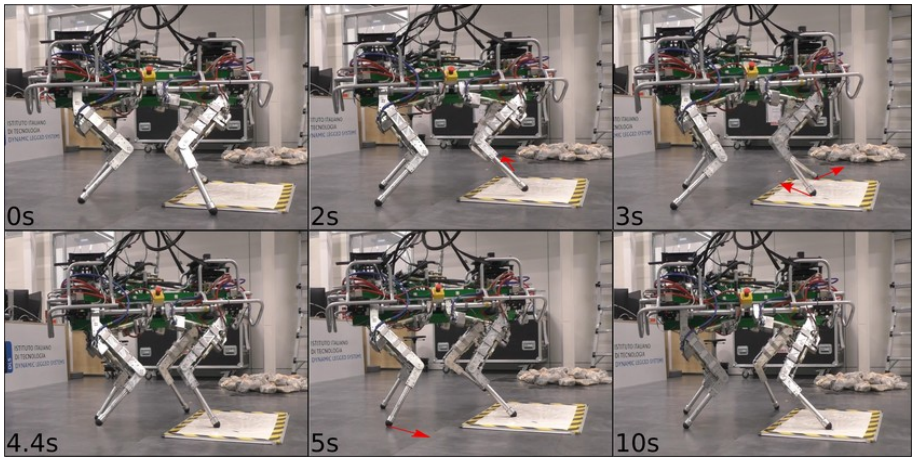


Figure 3.7: Experiment of the HyQ robot stepping from left to right on slippery terrain patches, using a crawl gait (low friction patches obtained by sprinkling a whiteboard with soap). The image sequence starts from the top left to right and continues at the bottom left. The red arrows indicate slipping feet.

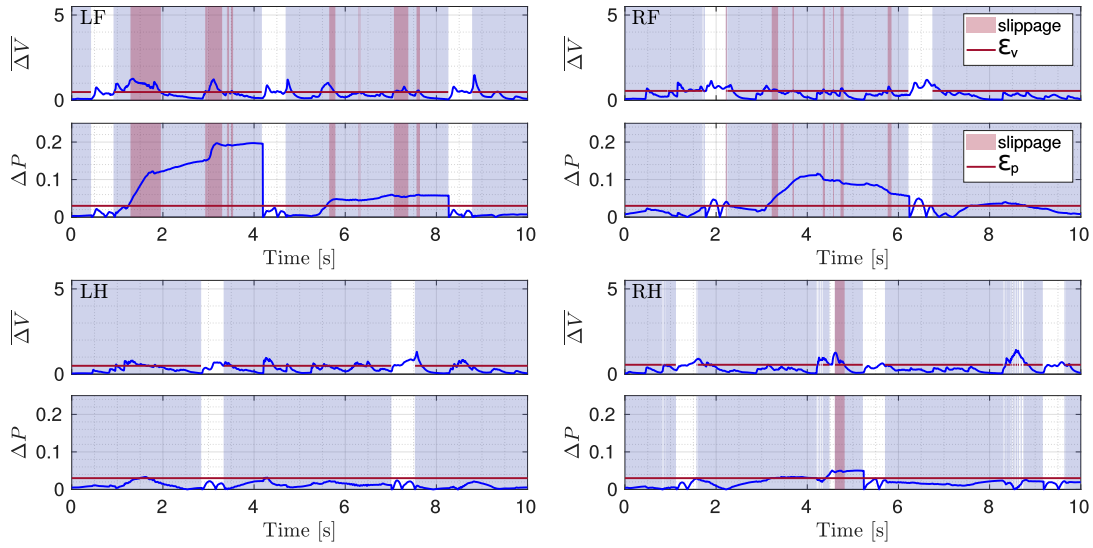


Figure 3.8: Plots of $\overline{\Delta V}$ and ΔP (blue) with respective thresholds (red line) illustrated for the four legs (LF, RF, LH, RH) during a crawling gait. The gray shaded area shows stance phases and the red shaded area marks slippage.

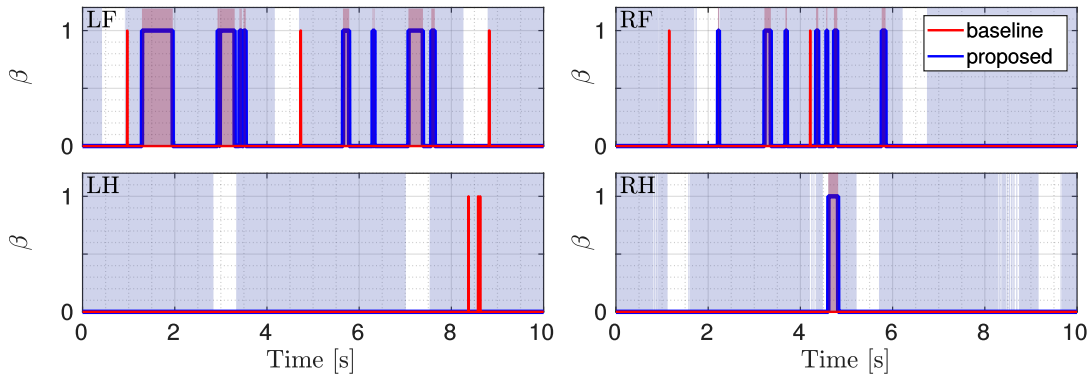


Figure 3.9: Comparison between the flags. The red one is obtained with the baseline approach, and the blue one with the proposed approach. The gray shaded area shows stance phases and the red shaded area indicates the ground truth.

3.10 Discussion

The proposed slip detection method for quadruped robots leverages foot velocities expressed in the body frame, ensuring independence from the robot's velocity and gait type. This design eliminates challenges typically associated with methods relying on the world frame, such as drift caused by inaccurate state estimation. The method was validated both in simulation and on the HyQ robot, demonstrating its versatility and effectiveness in detecting slippage across different gaits, including trotting and crawling. The results highlight the method's ability to outperform the baseline approach, offering accurate slippage detection. This is particularly valuable for maintaining stability during locomotion and discarding unreliable measurements in the estimation of the robot's state. The use of kinematics-based measurements, which rely directly on foot position and velocity data, ensures robustness against uncertainties such as ground reaction force (GRF) estimation errors often seen in force-based approaches. However, while the current results are promising, further research is needed to refine the method's adaptability to diverse terrains and environmental conditions. Addressing these areas will enhance the method's reliability in real-world applications.

3.10.1 Limitations

Despite its advantages, the proposed method has certain limitations:

- **Threshold Sensitivity:** The thresholds ϵ_v and ϵ_p require careful tuning for each specific robot and terrain. This tuning process can be labor-intensive and may reduce the method's generalizability.

- **Measurement Dependency:** The method's performance is heavily reliant on the accuracy of foot position and velocity measurements in the base frame, calculated using forward kinematics. Errors or noise in these measurements, due to sensor inaccuracies or external disturbances, can affect detection reliability.

Additionally, in our approach, a slipping leg is identified when its measured velocity and position deviate significantly from the commanded (desired) velocity and position in the body frame. This criterion is generally reliable under normal walking conditions, where the foot is expected to move as commanded. However, we acknowledge it may fail in some less common situations. For instance it can incorrectly classify a "trapped" foot (e.g., wedged under a rock) as slipping if the commanded velocity is high but the foot cannot actually move. In such edge cases, additional checks or complementary sensing strategies (e.g., contact force monitoring) would be necessary to discriminate between a legitimately slipping foot and one that is immovable.

3.11 Conclusion

In this chapter, we presented a kinematics-based slip detection approach for legged robots that relies on velocity and position measurements at ground contacts. Unlike force-based methods, which require six-axis force/torque sensors at the feet, this approach uses kinematic data, making it simpler and more adaptable. By expressing foot velocities in the body frame, the method avoids drift issues typically encountered when using the world frame, ensuring reliable performance across various locomotion types and during velocity transitions.

The method was validated through simulations and physical experiments on the HyQ robot, demonstrating its ability to detect slippage quickly and effectively. Results showed that the proposed method is more robust than the baseline approach, accurately detecting slippage events without false positives, even in challenging scenarios. This robustness makes it suitable for diverse terrains and gaits, including dynamic tasks where the robot's velocity changes frequently.

Future research will focus on expanding the applicability and reliability of the approach. Key areas include:

- **Tolerable Slippage Analysis:** Investigating the maximum slippage allowable while preserving locomotion stability.
- **Terrain Friction Estimation:** Developing methods to estimate terrain friction

properties during locomotion, enabling the robot to adjust its gait and cautiousness dynamically based on the situation.

- **Recovery Strategies:** Implementing strategies for recovering from slippage on extreme terrains, such as icy surfaces or slopes with incorrectly estimated inclinations.
- **Integration with Vision:** Incorporating visual data to estimate terrain friction coefficients and roughness, providing additional information to enhance detection accuracy.

These advancements will enable more resilient and adaptive locomotion, ensuring the robot's ability to operate effectively in complex and dynamic environments.

Finally, the proposed method has been used in the multi-sensor state estimator for quadruped robots presented in Chapter 4. The slip detection algorithm is integrated into the state estimator to improve the robustness of the state estimation in challenging terrains.

Chapter 4

The Real-Time Multi-Sensor State Estimator MUSE

4.1 Preface

This chapter introduces the Multi-Sensor State Estimator for quadruped robots MUSE, a real-time system that integrates data from multiple (proprioceptive and exteroceptive) sensors to deliver precise and dependable state estimates. Designed for on-board implementation, MUSE is capable of estimating the robot's pose, velocity, attitude, and contact forces. The framework combines Kalman filters with nonlinear observers to achieve robust performance. The effectiveness of MUSE is demonstrated through validation on the Aliengo and ANYmal quadruped robots, where we demonstrated that MUSE can consistently provide real-time state estimates with greater accuracy and reliability than other state-of-the-art state estimation methods.

The first findings of this research have been presented in the following publication:

> **Ylenia Nisticò**, João Carlos Virgolino Soares, Geoff Fink, and Claudio Semini, "*Multi-Sensor Fusion for Quadruped Robot State Estimation on Challenging Terrain*", *I-RIM 3D Conference 2024*

This work was conceptualized and developed by me and Geoff Fink, who also contributed to the software development. João Carlos Virgolino Soares assisted with the

development, validation, and formal analysis. The initial draft was prepared by me, with all authors contributing to review and editing. Supervision was provided by João Carlos Virgolino Soares, Geoff Fink, and Claudio Semini.

The main findings of this research have been presented in the following publication:

> **Ylenia Nisticò**, João Carlos Virgolino Soares, Lorenzo Amatucci, Geoff Fink, and Claudio Semini, "*MUSE: A Real-Time Multi-Sensor State Estimator for Quadruped Robots*", Under Review at *IEEE Robotics and Automation Letter*.

This work shares the same contributions as the previous one, with the addition of Lorenzo Amatucci, who participated in the experiment described in Section 4.7.2.1. The dataset used in Section 4.7.2.2 was provided by Prof. Maurice Fallon and Prof. Marco Camurri, whom we gratefully acknowledge.

The code utilized in this is available at the following link: `https://github.com/iit-DLSLab/muse`.

Additionally, a software framework named DLS2 has been employed to manage real-time communication between the robot sensors and the estimator. DLS2 is a real-time communication framework designed to simplify interfacing with robot sensors and actuators. While not directly related to state estimation, DLS2 is instrumental in facilitating the development of real-time applications on mobile robots. A detailed description of the DLS2 framework is provided in Appendix A.

4.2 Introduction

Perceptive information is essential for quadrupedal locomotion on unstructured terrain, where complex maneuvers are often required, as illustrated by Grandia et al. [103]. Accurate state estimation is a fundamental component for ensuring robust locomotion and perception in quadruped robots across diverse applications. It provides critical data on the robot's position, orientation, and velocity within its environment. In this context, real-time feedback is essential, because it enables the robot to adapt its gait, make prompt decisions, and plan locomotion strategies to maintain balance and stability. Moreover, state estimation is crucial for mitigating sensor failures or inaccuracies, allowing the robot to switch to alternative sensors or deploy recovery strategies to ensure safe and reliable navigation.

Research in state estimation has focused on integrating both proprioceptive sensors (e.g., IMUs, encoders, torque sensors) and exteroceptive sensors (e.g., cameras,

LiDARs). Exteroceptive sensors, primarily used for precise and low-drift robot pose estimation, have significantly advanced navigation as well as visual and LiDAR-based SLAM, as discussed by Cadena et al. [60]. However, as highlighted in Chapter 2, exteroceptive sensors face challenges in adverse conditions, such as failures due to environmental factors, frequency-related controller limitations, and delays in measurement acquisition.

To overcome these challenges, exteroceptive sensors are often complemented with proprioceptive sensors, which provide high-frequency data for rapid updates on the robot's state. Proprioceptive sensors excel in environments where exteroceptive sensors may struggle, such as poorly lit areas, occluded regions, or spaces lacking distinctive features, ensuring reliable state estimation across a broader range of operating conditions.

For legged robots, additional information can be obtained from leg kinematics, which provides precise details about the movement and positioning of each leg, further enhancing state estimation. Several notable studies have explored the fusion of proprioceptive sensors to improve state estimation. For instance, Bloesch et al. [7, 31], and Hartley et al. [36] have proposed methods that integrate data from proprioceptive sensors, including joint encoders, IMUs, and force sensors, while leveraging contact information to accurately estimate the robot's state. These studies have demonstrated the importance of incorporating contact information to enhance state estimation, particularly in challenging terrains.

Additionally, several studies have investigated the impact of terrain on state estimation for legged robots. Fahmi et al. [46] demonstrated that assuming static contact conditions can lead to inaccurate state estimates, especially when the robot traverses soft or uneven terrain. Similarly, Bloesch et al. [41] and Jenelten et al. [42] highlighted the challenges of maintaining accurate state estimation under dynamic and slippery conditions. These studies reveal that while avoiding the assumption of static contact can improve results, relying solely on proprioceptive state estimation is insufficient to provide a drift-free pose. For instance, in previous work by Santana et al. [49], the authors introduced an innovative InEKF designed specifically for legged robots, relying solely on proprioceptive sensors and incorporating robust cost functions in the measurement update. Although the use of these robust cost functions significantly reduced drift, they were not able to completely eliminate it. This underscores the necessity of incorporating exteroceptive sensor data for enhanced accuracy.

Over the years, various techniques for multi-sensor state estimation in legged robots have been developed. For instance, the Pronto state estimator (Camurri et al. [6]) employs an EKF to fuse data from an IMU and leg kinematics, while incorporating pose corrections derived from stereo vision and LiDAR inputs. Similarly, WALK-VIO, as

described in (Lim et al. [3]), combines data from an IMU, a camera, and joint encoders to estimate the robot's state. WALK-VIO uses a walking-motion-adaptive leg kinematic constraint that dynamically adjusts based on the robot's body motion.

However, both Pronto and WALK-VIO operate under a no-slip assumption, meaning they presume the robot maintains consistent contact with the ground without experiencing slippage or falls. While this assumption simplifies the state estimation process, it fails to account for slippage or loss of contact, which can significantly affect the accuracy and reliability of state estimation, particularly in challenging or unpredictable terrains.

The work by Teng et al. [90] utilizes an InEKF for state estimation in a bipedal robot navigating slippery terrain. This approach integrates vision-based velocity measurements from a Realsense T265 Tracking Camera [104] with data from an IMU and leg kinematics. To address potential measurement noise from the camera, an online noise parameter tuning method is introduced, allowing the system to adapt dynamically. In another study, Kim et al. [4] proposed the STEP state estimator, which estimates the end-effector's pose using pre-integrated foot velocity factors and body speed data obtained from a stereo camera. This method eliminates the need for contact detection and the assumption of non-slip conditions, offering a more flexible approach to state estimation.

The main drawback is that these two last methods rely heavily on camera inputs, which can sometimes be unreliable. This dependence may impact the accuracy and robustness of the estimated states, particularly in environments where visual inputs are degraded or unavailable.

VILENS, introduced by Wisth et al. [2], integrates data from IMU, kinematics, LiDAR, and cameras using factor graphs to provide robust state estimation, even when individual sensors encounter failures. That said, this state estimator has not been applied to close the loop with the controller during online experiments. As a result, it has yet to demonstrate its capability for providing real-time feedback in robot control scenarios.

Leg-KILO, introduced by Ou et al. [93], is a state estimator that integrates LiDAR odometry with kinematic and inertial measurements to determine a robot's pose, using loop closures to mitigate drift over time. Although loop closures can introduce abrupt corrections in the global map frame, systems like ROS typically address this by separating the local "`odom`" frame (for controller feedback) from the global "`map`" frame (for drift-free localization). With proper frame management, loop closures need not cause instability in real-time applications. However, in environments where loop opportunities are limited or delayed, such as extended corridors, drift may accumulate for longer periods before it is corrected at the global level. How real-time control is affected ultimately depends on how well the system keeps global corrections separate from the local control loop.

4.3 Contributions

In this chapter, we introduce MUSE, an innovative state estimator for legged robots, building on the earlier work of Fink and Semini [34]. In this prior research, the authors developed a nonlinear observer for attitude estimation and computed leg odometry based on a quadruped model, combining these through sensor fusion using a Kalman Filter (KF). Expanding on that foundation, this work presents a comprehensive state estimation pipeline. It incorporates exteroceptive sensors and integrates the slip detection module from Nisticò et al. [1] (see Chapter 3), which uses a kinematics-based approach to detect simultaneous slippage of one or more legs.

This new approach results in a low-drift state estimator that is robust to sensor failures and adaptable to uneven, unstructured environments.

In this context, the main contributions of this work are stated as follows:

- **Integration of a Slip Detection Module in State Estimation**: To the best of our knowledge, this is the first multi-sensor (proprioceptive and exteroceptive) state estimation pipeline to include a dedicated slip detection module, critical for navigating uneven and unstructured terrain.
- **Real-Time Feedback for Locomotion Control**: Unlike previous approaches such as WALK-VIO (Lim et al. [3]), STEP (Kim et al. [4]), and VILENS (Wisth et al. [2]), MUSE has been successfully employed to provide real-time feedback to the locomotion controller in experiments conducted on the Aliengo robot.
- **Extensive Online and Offline Evaluation Across Platforms and Scenarios**: Our work was validated on the Aliengo robot in indoor environments on difficult scenarios, and on the ANYmal B300 robot, in the Fire Service College (FSC) Dataset (Wisth et al. [2], Camurri et al. [6]). We demonstrated significant improvements in state estimation accuracy compared to state-of-the-art methods:
  - 67.6% reduction in the translational errors compared to Pronto (Camurri et al. [6]).
  - 26.67% reduction in the translational errors compared to VILENS (Wisth et al. [2]).
  - 45.9% reduction in the absolute trajectory error compared to Two-State Information Filter (TSIF) (Bloesch et al. [7]).
  - Superior rotational accuracy and frequency performance compared to DLIO (Chen et al. [8]), a LiDAR-inertial odometry system.

To support the research community, we will make MUSE's code publicly available under an open-source license. The code will be accessible at this link.

4.4 Outline

The remainder of this chapter is structured as follows: in Section 4.5, we provide a theoretical background on the Kalman filter, nonlinear observers, and the eXogeneous Kalman filter. In Section 4.6, we present the MUSE formulation, including the robot models, exteroceptive odometry, contact estimation, leg odometry, slip detection, attitude observer, and sensor fusion. In Section 4.7, we present the experimental results obtained with the Aliengo and ANYmal robots. Finally, in Section 4.8, we discuss the results, and Section 4.9 concludes the chapter.

4.5 Theoretical Background

This section provides a concise overview of the theoretical foundations of the Kalman filter, Nonlinear observers, and the eXogeneous Kalman filter, which form the basis of the MUSE formulation.

A detailed exploration of state estimation theory is beyond the scope of this work. For readers seeking a deeper understanding, we recommend consulting the following authoritative references: (Kalman et al. [105], Simon [106], Mahony et al. [107], Johansen and Fossen [108]).

4.5.1 Kalman Filter

The Kalman filter is an optimal estimator for linear systems with Gaussian noise, designed to minimize the mean squared error of the state estimate. It achieves this by combining two key steps:

- **Prediction**: Propagating the system dynamics to estimate the state at the next time step.
- **Correction**: Updating the state estimate based on new measurements.

The Kalman filter operates under the assumption that the system is linear and that the process and measurement noises are white, Gaussian, and mutually independent.

For discrete-time systems, the filter alternates between prediction and update steps. In continuous-time systems, it operates continuously by solving differential equations for

state estimation and covariance. In this section, we derive the equations for both discrete and continuous-time Kalman filters. Readers interested in a detailed derivation of the continuous-time equations from the discrete-time ones are referred to (Simon [106]), where the continuous-time Kalman filter is derived by taking the limit as the sample time approaches zero. Continuous-time Kalman filters are particularly useful when the system is continuously observed, requiring ongoing state estimation updates.

4.5.1.1 Linear Time-Varying Continuous-Time Kalman Filter

Considering a continuous-time linear system described by the following state-space model:

$$\dot{\mathbf{x}}(\mathbf{t}) = \mathbf{A}\mathbf{x}(\mathbf{t}) + \mathbf{B}\mathbf{u}(\mathbf{t}) + \mathbf{w}(\mathbf{t}), \quad \mathbf{w}(\mathbf{t}) \sim (\mathbf{0}, \mathbf{Q_c}) \tag{4.1a}$$

$$\mathbf{y}(\mathbf{t}) = \mathbf{C}\mathbf{x}(\mathbf{t}) + \mathbf{v}(\mathbf{t}), \quad \mathbf{v}(\mathbf{t}) \sim N(\mathbf{0}, \mathbf{R_c}) \tag{4.1b}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state, $\mathbf{u} \in \mathbb{R}^m$ is the input, $\mathbf{y} \in \mathbb{R}^p$ is the output, $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times m}$, and $\mathbf{C} \in \mathbb{R}^{p \times n}$ are the system matrices, and $\mathbf{w}$ and $\mathbf{v}$ are the process and measurement noise, respectively. The Kalman filter estimates the state $\mathbf{x}$ given the measurements $\mathbf{y}$ and the input $\mathbf{u}$. $\mathbf{Q_c}$ and $\mathbf{R_c}$ are the covariance matrices of the process and measurement noise, respectively. In this context, state estimation is updated continuously over time to accommodate real-time system dynamics.

Prediction of the State and Error Covariance The state evolution is governed by the following differential equation:

$$\dot{\hat{\mathbf{x}}}(\mathbf{t}) = \mathbf{A}\hat{\mathbf{x}}(\mathbf{t}) + \mathbf{B}\mathbf{u}(\mathbf{t}) \tag{4.2}$$

Meanwhile, the covariance of the estimation error evolves according to the Riccati differential equation:

$$\dot{\mathbf{P}}(\mathbf{t}) = \mathbf{A}\mathbf{P}(\mathbf{t}) + \mathbf{P}(\mathbf{t})\mathbf{A}^\top + \mathbf{Q_c} - \mathbf{P}(\mathbf{t})\mathbf{C}^\top\mathbf{R_c^{-1}}\mathbf{C}\mathbf{P}(\mathbf{t}) \tag{4.3}$$

This equation captures the evolution of the uncertainty in the state estimation over time, balancing the system's intrinsic dynamics, process noise, and the contribution of measurements to reduce the uncertainty.

Update with Continuous Measurements In the continuous-time scenario, the measurement update takes place continuously, and the Kalman gain is expressed as:

$$\mathbf{K(t)} = \mathbf{P(t)C}^\top \mathbf{R_c^{-1}} \tag{4.4}$$

The corresponding update for the state estimate is given by:

$$\dot{\hat{\mathbf{x}}}(\mathbf{t}) = \dot{\hat{\mathbf{x}}}(\mathbf{t}) + \mathbf{K(t)}(\mathbf{y(t)} - \mathbf{C}\hat{\mathbf{x}}(\mathbf{t})) \tag{4.5}$$

where $\mathbf{y(t)} - \mathbf{C}\hat{\mathbf{x}}(\mathbf{t})$ represents the innovation or measurement residual. The error covariance is updated according to:

$$\dot{\mathbf{P}}(\mathbf{t}) = \mathbf{P(t)} - \mathbf{K(t)CP(t)} \tag{4.6}$$

4.5.1.2 Linear Time-Varying Discrete-Time Kalman Filter

Now, if we discretize the system with a sample time T the discrete-time equations are obtained as:

$$\mathbf{x_{k+1}} = \mathbf{Fx_k} + \mathbf{Gu_k} + \mathbf{\Lambda w_k}, \quad \mathbf{w_k} \sim (\mathbf{0}, \mathbf{Q}) \tag{4.7a}$$

$$\mathbf{y_k} = \mathbf{Cx_k} + \mathbf{v_k}, \quad \mathbf{v_k} \sim N(\mathbf{0}, \mathbf{R}) \tag{4.7b}$$

The objective of the Kalman filter is to estimate the state $\mathbf{x_k}$ from the noisy measurements $\mathbf{y_k}$, while minimizing the covariance of the state estimation error.

Prediction of the State and Error Covariance The predicted state, based on the previous state estimate, is given by:

$$\hat{\mathbf{x}}_{\mathbf{k|k-1}} = \mathbf{F}\hat{\mathbf{x}}_{\mathbf{k-1|k-1}} + \mathbf{Gu_{k-1}} \tag{4.8}$$

where $\hat{\mathbf{x}}_{\mathbf{k-1|k-1}}$ is the previous state estimate, $\mathbf{u_{k-1}}$ is the input at the previous iteration, and $\hat{\mathbf{x}}_{\mathbf{k|k-1}}$ is the predicted state at time k, based on the information up to time $\mathbf{k-1}$. The error covariance prediction is:

$$\mathbf{P_{k|k-1}} = \mathbf{FP_{k-1|k-1}F}^\top + \mathbf{\Lambda Q \Lambda}^\top \tag{4.9}$$

Update Step with Discrete Measurements The Kalman filter updates the state estimate using the new measurement $\mathbf{y_k}$. The innovation (or measurement residual),

which quantifies the difference between the observed measurement and the predicted measurement, is given by:

$$\tilde{\mathbf{y}}_{\mathbf{k}} = \mathbf{y}_{\mathbf{k}} - \mathbf{H}\hat{\mathbf{x}}_{\mathbf{k|k-1}} \tag{4.10}$$

while the innovation covariance is:

$$\mathbf{S}_{\mathbf{k}} = \mathbf{H}\mathbf{P}_{\mathbf{k|k-1}}\mathbf{H}^{\top} + \mathbf{R} \tag{4.11}$$

The Kalman gain determines how much weight is given to the new measurement:

$$\mathbf{K}_{\mathbf{k}} = \mathbf{P}_{\mathbf{k|k-1}}\mathbf{H}^{\top}{\mathbf{S}_{\mathbf{k}}}^{-1} \tag{4.12}$$

The state is updated by incorporating the new measurement, as follows:

$$\hat{\mathbf{x}}_{\mathbf{k|k}} = \hat{\mathbf{x}}_{\mathbf{k|k-1}} + \mathbf{K}_{\mathbf{k}}\tilde{\mathbf{y}}_{\mathbf{k}} \tag{4.13}$$

while the error covariance is updated as:

$$\mathbf{P}_{\mathbf{k|k}} = (\mathbf{I} - \mathbf{K}_{\mathbf{k}}\mathbf{H})\mathbf{P}_{\mathbf{k|k-1}} \tag{4.14}$$

where $\mathbf{I}$ is the identity matrix.

4.5.2 Nonlinear Kalman Filters

Up to this point, our discussion has focused on linear filters for linear systems. However, in reality, purely linear systems are an idealization since most systems are inherently nonlinear. For such systems, extensions of the Kalman filter, collectively referred to as Nonlinear Kalman Filters, are employed.

In a general nonlinear system, the dynamics and measurements are governed by the following nonlinear equations:

Continuous-Time Nonlinear System:

$$\dot{\mathbf{x}}(\mathbf{t}) = f(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t})) + \mathbf{w}(\mathbf{t}), \quad \mathbf{w}(\mathbf{t}) \sim (\mathbf{0}, \mathbf{Q}_{\mathbf{c}}) \tag{4.15a}$$

$$\mathbf{y}(\mathbf{t}) = \mathbf{h}(\mathbf{x}(\mathbf{t})) + \mathbf{v}(\mathbf{t}), \quad \mathbf{v}(\mathbf{t}) \sim N(\mathbf{0}, \mathbf{R}_{\mathbf{c}}) \tag{4.15b}$$

Discrete-Time Nonlinear System:

$$\mathbf{x_{k+1}} = f(\mathbf{x_k}, \mathbf{u_k}) + \mathbf{\Lambda w_k}, \quad \mathbf{w_k} \sim (\mathbf{0}, \mathbf{Q}) \tag{4.16a}$$

$$\mathbf{y_k} = h(\mathbf{x_k}) + \mathbf{v_k}, \quad \mathbf{v_k} \sim N(\mathbf{0}, \mathbf{R}) \tag{4.16b}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state, $\mathbf{u} \in \mathbb{R}^m$ is the input, $\mathbf{y} \in \mathbb{R}^p$ is the output, f and h are the nonlinear state transition function and nonlinear measurement function, respectively; $\mathbf{w}$ and $\mathbf{v}$ are the gaussian and zero-mean process and measurement noise, respectively, and $\mathbf{Q_c}$ and $\mathbf{R_c}$ are the covariance matrices of the process and measurement noise, respectively.

In nonlinear systems, the state distributions (e.g. $p(\mathbf{x_k}|\mathbf{y_{1:k}})$) may no longer remain Gaussian due to the nonlinearities in the system dynamics and measurement functions. These nonlinearities in the $f(\cdot)$ and $h(\cdot)$ functions require approximations or numerical methods to handle the transformations of both the state and the error covariance. Moreover, unlike linear systems, there is no closed-form analytical solution for state estimation in nonlinear systems, making approximations a necessity. These approximations are typically designed to strike a balance between computational efficiency and estimation accuracy.

To address these challenges, several extensions of the Kalman filter have been developed, including:

- **Extended Kalman Filter (EKF)** (McGee and Schmidt [109]): The EKF linearizes the system dynamics and measurement equations around the current state estimate using a first-order Taylor expansion. It is widely used in practice due to its simplicity and computational efficiency. However, the EKF has notable limitations, including the need for a good initial guess, the requirement for the system to remain approximately linear, and the risk of divergence when the linearization is inadequate.

- **Unscented Kalman Filter (UKF)** (Wan and Van Der Merwe [110]): The UKF employs a deterministic sampling technique to generate a minimal set of sample points (sigma points) that accurately capture the mean and covariance of the state distribution. Unlike the EKF, the UKF avoids linearization errors, making it more accurate for highly nonlinear systems. However, this accuracy comes at the cost of increased computational complexity due to the need for sigma point propagation.

- **Particle Filter (PF)** (Salmond and Gordon [111]): The PF represents the state

distribution using a set of weighted particles, making it a non-parametric filter capable of approximating arbitrary distributions. This flexibility makes the PF suitable for highly nonlinear and non-Gaussian systems. However, it is computationally intensive and prone to sample degeneracy, particularly in high-dimensional state spaces.

In the following section (Section 4.5.2.1), we will focus on the EKF, as it serves as the foundation for understanding the eXogeneous Kalman Filter (XKF). The theoretical framework for the XKF is presented in Section 4.5.4, and its implementation is applied in the Attitude Observer formulation discussed in Section 4.6.6.

4.5.2.1 Extended Kalman Filter

The EKF enables state estimation for nonlinear models by linearizing the system dynamics around the current state estimate. In this section, we derive the EKF equations step by step and explain the underlying theory in detail.

Continuous-Time EKF Similar to the Linear Time-Varying (LTV)–KF, the EKF operates by predicting the state and error covariance based on the system dynamics, and then updating the state and covariance using new measurements. The primary distinction is that the EKF linearizes the system dynamics and measurement functions around the current state estimate. Starting from the continuous-time nonlinear system described in Equation (4.15), the nonlinear functions $f(\mathbf{x}, \mathbf{u})$ and $h(\mathbf{x})$ are linearized around the current estimate using a first-order Taylor expansion:

Prediction step: The prediction step advances the state and covariance forward in time based on the system dynamics. The continuous-time nonlinear system equations are used to compute:

$$\dot{\hat{\mathbf{x}}}(\mathbf{t}) = f(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{u}(\mathbf{t})) + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}(\mathbf{t})} (\mathbf{x}(\mathbf{t}) - \hat{\mathbf{x}}(\mathbf{t})) + \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\hat{\mathbf{u}}(\mathbf{t})} (\mathbf{u}(\mathbf{t}) - \hat{\mathbf{u}}(\mathbf{t})) \tag{4.17}$$

where $f(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{u}(\mathbf{t}))$ is the predicted evolution of the state estimate, $\left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}(\mathbf{t})} (\mathbf{x}(\mathbf{t}) - \hat{\mathbf{x}}(\mathbf{t}))$ accounts for the deviations in the state $\mathbf{x}(\mathbf{t})$, around $\hat{\mathbf{x}}\mathbf{x}(\mathbf{t})$, while $\left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\hat{\mathbf{u}}(\mathbf{t})} (\mathbf{u}(\mathbf{t}) - \hat{\mathbf{u}}(\mathbf{t}))$ accounts for the deviations in the input $\mathbf{u}(\mathbf{t})$, around $\hat{\mathbf{u}}(\mathbf{t})$. The error covariance evolves according to:

$$\dot{\mathbf{P}}(\mathbf{t}) = \frac{\partial f}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}(\mathbf{t})} \mathbf{P}(\mathbf{t}) + \mathbf{P}(\mathbf{t}) \frac{\partial f}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}(\mathbf{t})}^{\top} + \mathbf{Q_c} - \mathbf{P}(\mathbf{t}) \frac{\partial h}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}(\mathbf{t})}^{\top} \mathbf{R_c^{-1}} \frac{\partial h}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}(\mathbf{t})} \mathbf{P}(\mathbf{t}) \tag{4.18}$$

where $\mathbf{P}(\mathbf{t})$ is the error covariance matrix. If we call $\mathbf{F} = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}(\mathbf{t})}$, $\mathbf{H} = \frac{\partial h}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}(\mathbf{t})}$, and $\mathbf{Q} = \mathbf{Q_c}$, the prediction step can be written as:

$$\dot{\hat{\mathbf{x}}}(\mathbf{t}) = \mathbf{F}\hat{\mathbf{x}}(\mathbf{t}) + \mathbf{G}\mathbf{u}(\mathbf{t}) \tag{4.19a}$$

$$\dot{\mathbf{P}}(\mathbf{t}) = \mathbf{F}\mathbf{P}(\mathbf{t}) + \mathbf{P}(\mathbf{t})\mathbf{F}^{\top} + \mathbf{Q} - \mathbf{P}(\mathbf{t})\mathbf{H}^{\top}\mathbf{R_c^{-1}}\mathbf{H}\mathbf{P}(\mathbf{t}) \tag{4.19b}$$

Update step: The update step corrects the state estimate when a new measurement $\mathbf{y}(\mathbf{t})$ becomes available. The innovation, which represents the discrepancy between the observed measurement and the predicted measurement, is given by:

$$\tilde{\mathbf{y}}(\mathbf{t}) = \mathbf{y}(\mathbf{t}) - h(\hat{\mathbf{x}}(\mathbf{t})) \tag{4.20}$$

while the innovation covariance is:

$$\mathbf{S}(\mathbf{t}) = \mathbf{H}\mathbf{P}(\mathbf{t})\mathbf{H}^{\top} + \mathbf{R_c} \tag{4.21}$$

where $\mathbf{H} = \frac{\partial h}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}(\mathbf{t})}$ is the Jacobian of the measurement function, and $\mathbf{R_c}$ is the measurement noise covariance.

The Kalman gain, that balances the uncertainty in the prediction and measurement, is:

$$\mathbf{K}(\mathbf{t}) = \mathbf{P}(\mathbf{t})\mathbf{H}^{\top}\mathbf{S}(\mathbf{t})^{-\mathbf{1}} \tag{4.22}$$

In the end, the state and the covariance updates are:

$$\hat{\mathbf{x}}(\mathbf{t}) = \hat{\mathbf{x}}(\mathbf{t}) + \mathbf{K}(\mathbf{t})\tilde{\mathbf{y}}(\mathbf{t}) \tag{4.23a}$$

$$\mathbf{P}(\mathbf{t}) = (\mathbf{I} - \mathbf{K}(\mathbf{t})\mathbf{H})\mathbf{P}(\mathbf{t}) \tag{4.23b}$$

where $\mathbf{I}$ is the identity matrix.

Discrete-Time EKF For the discrete-time case, we begin with the discrete-time system described in Equation (4.16). Similar to the continuous-time case, the functions $f(\mathbf{x_k}, \mathbf{u_k})$ and $h(\mathbf{x_k})$ are nonlinear and are therefore linearized around the current state

estimate using a first-order Taylor expansion:

$$f(\mathbf{x_k}, \mathbf{u_k}) \approx f(\hat{\mathbf{x}}_{\mathbf{k|k-1}}, \mathbf{u_k}) + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{\mathbf{k|k-1}}} (\mathbf{x_k} - \hat{\mathbf{x}}_{\mathbf{k|k-1}}) + \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\hat{\mathbf{u}}_{\mathbf{k}}} (\mathbf{u_k} - \mathbf{u_k}) \tag{4.24}$$

The measurement function is linearized as:

$$h(\mathbf{x_k}) \approx h(\hat{\mathbf{x}}_{\mathbf{k|k-1}}) + \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{\mathbf{k|k-1}}} (\mathbf{x_k} - \hat{\mathbf{x}}_{\mathbf{k|k-1}}) \tag{4.25}$$

We define the matrices $\mathbf{F_k}$ and $\mathbf{H_k}$ as:

$$\mathbf{F_k} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{\mathbf{k|k-1}}}, \quad \mathbf{H_k} = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{\mathbf{k|k-1}}} \tag{4.26}$$

These matrices represent the Jacobians of the system dynamics and measurement functions, respectively.

Prediction step: The prediction step advances the state forward in time using the nonlinear state function:

$$\hat{\mathbf{x}}_{\mathbf{k|k-1}} = f(\hat{\mathbf{x}}_{\mathbf{k-1|k-1}}, \mathbf{u_{k-1}}) \tag{4.27}$$

The error covariance prediction is:

$$\mathbf{P_{k|k-1}} = \mathbf{F_k} \mathbf{P_{k-1|k-1}} \mathbf{F_k^\top} + \mathbf{\Lambda Q \Lambda^\top} \tag{4.28}$$

where $\mathbf{\Lambda}$ is the Jacobian of the process noise function.

Update step: The update step refines the state estimate by incorporating the new measurement y_k. The innovation is given by:

$$\tilde{\mathbf{y}}_\mathbf{k} = \mathbf{y_k} - h(\hat{\mathbf{x}}_{\mathbf{k|k-1}}) \tag{4.29}$$

while the innovation covariance is:

$$\mathbf{S_k} = \mathbf{H_k} \mathbf{P_{k|k-1}} \mathbf{H_k^\top} + \mathbf{R} \tag{4.30}$$

where $\mathbf{H} = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{\mathbf{k|k-1}}}$ is the Jacobian of the measurement function, and $\mathbf{R}$ is the measurement noise covariance. The Kalman gain is computed as:

$$\mathbf{K_k} = \mathbf{P_{k|k-1}} \mathbf{H_k}^\top \mathbf{S_k^{-1}} \tag{4.31}$$

The state update is:

$$\hat{\mathbf{x}}_{\mathbf{k}|\mathbf{k}} = \hat{\mathbf{x}}_{\mathbf{k}|\mathbf{k-1}} + \mathbf{K}_\mathbf{k}\tilde{\mathbf{y}}_\mathbf{k} \tag{4.32}$$

while the error covariance update is:

$$\mathbf{P}_{\mathbf{k}|\mathbf{k}} = (\mathbf{I} - \mathbf{K}_\mathbf{k}\mathbf{H}_\mathbf{k})\mathbf{P}_{\mathbf{k}|\mathbf{k-1}} \tag{4.33}$$

where $\mathbf{I}$ is the identity matrix.

Summary of the EKF Theory and Interpratation The EKF assumes that the system dynamics and measurement model are locally linear at each time step. This approximation enables the filter to apply linear Kalman filter theory to nonlinear systems. Additionally, the EKF operates recursively, updating the state estimate and its covariance incrementally at each time step, which is computationally efficient compared to batch processing all measurements. By linearizing around the current state estimate, the EKF effectively handles systems with mild nonlinearities.

However, in highly nonlinear systems, the linearization may introduce significant errors, rendering the filter suboptimal. To address this issue, we explored the XKF, which is used in combination with the Nonlinear Observer (NLO) for attitude estimation. The theoretical foundations of these observers are detailed in the following sections.

4.5.3 Nonlinear Observer

In dynamic systems, particularly those involving rigid-body mechanics, attitude estimation refers to determining the orientation of a body in space relative to a reference frame. Nonlinear observers play a crucial role in such problems, as the dynamics and measurements are inherently nonlinear, and the state space (e.g., rotation matrices on $SO(3)$) exhibits a complex geometric structure. Mahony et al. [112], explored the design of nonlinear state observers for kinematic systems with symmetry, providing a robust theoretical framework for nonlinear observer design. Their approach leverages the geometric properties and inherent symmetry of these systems. Building on their work, we present a methodology for designing observers that utilize symmetry properties, resulting in autonomous error dynamics and strong convergence guarantees.

A kinematic system evolves according to:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{v}) \tag{4.34}$$

where $\mathbf{x} \in \mathbf{X}$ is the state of the system on a state space $\mathbf{X}$ (e.g. $SO(3)$, $SE(3)$, or their

homogeneous spaces), $\mathbf{v} \in \mathbf{V}$ is the velocity input, tipically measured or known, and $f(\mathbf{x}, \mathbf{v})$ is the state dynamics, often dependent on the symmetry properties of the system. The system outputs are given as:

$$\mathbf{y} = h(\mathbf{x}) \tag{4.35}$$

where $\mathbf{y} \in \mathbf{Y}$ represents the measurement model, mapping the state $\mathbf{x}$ to an output space Y.

Both the state dynamics $f(\mathbf{x}, \mathbf{v})$ and measurement functions $h(\mathbf{x})$ are nonlinear, necessitating the use of advanced estimation techniques. Additionally, the state space $\mathbf{X}$ often resides on a nonlinear manifold (e.g., rotation matrices in $SO(3)$), rendering conventional linear estimator techniques inadequate. Achieving global or almost-global convergence of the estimation error remains a significant challenge, particularly in rotation, and pose estimation problems.

Many mechanical systems exhibit symmetry properties due to their invariant behavior under specific transformations (e.g., rotations and translations). These properties can be leveraged to simplify observer design, particularly for kinematic models where the state evolves linearly with respect to velocity inputs. For instance, the attitude of a rigid body in 3D space can be represented as a rotation, commonly modeled using a unit quaternion. The unit quaternion space forms a 3D sphere, which is a Lie group. In such cases, the state space $\mathbf{X}$ is typically modeled as a homogeneous space $\mathbf{X} = \mathbf{G}/\mathbf{H}$ where $\mathbf{G}$ is the symmetry group, and $\mathbf{H}$ is the stabilizer subgroup.

Definition: A system is said to be equivariant if the state dynamics $f(\mathbf{x}, \mathbf{v})$ respect the group action of $\mathbf{G}$, and the output $\mathbf{y} = h(\mathbf{x})$ are consistent under the same symmetry. For instance, in attitude estimation, the state $\mathbf{x} \in SO(3)$ represents a rotation matrix describing the orientation of a body in three-dimensional space. The system exhibits invariance under changes in the reference frame, which is a symmetry of the rotation group $SO(3)$.

The observer design is based on the following principles:

- **Error Dynamics**: The error dynamics are defined on the tangent space of the Lie group, which is a vector space. This approach enables the error dynamics to remain linear, even though the overall system dynamics are nonlinear. This is a key advantage of leveraging the geometric properties of Lie groups for observer design. The error between the true state $\mathbf{X}_{\text{true}}$ and the estimated state $\mathbf{X}_{\text{observer}}$ is represented using two canonical error functions, both of which reside in the symmetry group $\mathbf{G}$:

  - **Right-Invariant Error**: $\mathbf{E} = \mathbf{X}_{\text{observer}}\mathbf{X}_{\text{true}}^{-1}$, commonly used when sensors are body-fixed (e.g., IMU measurements on a drone).
  - **Left-Invariant Error**: $\mathbf{E} = \mathbf{X}_{\text{true}}^{-1}\mathbf{X}_{\text{observer}}$, suitable for systems where sensors measure in the global frame.

- **Autonomous Dynamics**: The error dynamics are autonomous, meaning they evolve independently of the system's input.

$$\dot{\mathbf{E}} = f(\mathbf{E}) \tag{4.36}$$

  where $\mathbf{E}$ is the error. This property greatly simplifies stability analysis, as the error dynamics are determined solely by the error state itself, without being influenced by the specific trajectory of the system. By decoupling the error evolution from the input, the observer design can focus entirely on the convergence properties of the error dynamics. This ensures that stability and convergence guarantees are valid for any trajectory of the system, making the observer robust to variations in the input and system dynamics.

- **Convergence**: The observer is designed to ensure that the error dynamics converge to zero, which guarantees that the estimated state produced by the observer asymptotically approaches the true state. This is achieved by carefully constructing the observer's update laws to stabilize the error dynamics, leveraging the system's geometric properties and symmetry to ensure robustness and convergence across a wide range of operating conditions.

- **Symmetry**: The observer design leverages the inherent symmetries of the system to simplify the error dynamics and ensure convergence. By aligning the observer structure with the system's symmetry properties, the error dynamics are made more tractable, often leading to autonomous or linearized forms. This approach enhances the observer's robustness and facilitates the design of control laws that guarantee the convergence of the estimated state to the true state.

- **Global Convergence**: The observer is designed to ensure global convergence, meaning the error dynamics converge to zero regardless of the initial conditions. This guarantees that the observer's state estimate will asymptotically approach the true state for any starting point in the state space, providing robust and reliable performance across a wide range of scenarios.

- **Computational Efficiency**: The observers are designed to be computationally efficient, ensuring they can operate in real-time applications. This efficiency is achieved by simplifying the mathematical operations and leveraging system symmetries, enabling the observer to deliver accurate state estimates with minimal computational overhead. This makes them well-suited for time-critical tasks in dynamic systems.

The corrective mechanism in the observer dynamics is the symmetry-preserving innovation term $\Delta(\hat{\mathbf{X}}, \mathbf{y})$, which is designed as an equivariant function of the output error. This term ensures that the correction respects the system's symmetry properties, allowing the observer to remain consistent with the underlying geometric structure:

$$\Delta(\hat{\mathbf{X}}, \mathbf{y}) = k \sum_i w_i (\hat{\mathbf{X}}^\top \mathbf{v}_i - \mathbf{y}_i) \tag{4.37}$$

where k is a gain parameter, w_i are the weights for each reference vector, $\mathbf{v}_i$ are reference vectors in the global frame (e.g. gravity, magnetic field), and $\mathbf{y}_i$ are measured vectors in the local frame. By aligning the estimated vectors with the measured ones, the innovation ensures convergence while preserving the system's symmetry.

Additionally, the innovation term is designed as a gradient descent process on the error function, aiming to minimize a Lyapunov candidate function $V(\mathbf{E})$. This approach guarantees that the error decreases monotonically, contributing to the stability of the observer. Formally the innovation term $\Delta(\hat{\mathbf{X}}, \mathbf{y})$ is constructed to satisfy:

$$\Delta(\hat{\mathbf{X}}, \mathbf{y}) = -\nabla V(\mathbf{E}) \tag{4.38}$$

A typical Lyapunov candidate is:

$$V(\mathbf{E}) = \frac{1}{2} ||\text{vex}(P_a(\mathbf{E}))||^2 \tag{4.39}$$

where $P_a(\mathbf{E}) = \frac{1}{2}(\mathbf{E} - \mathbf{E}^\top)$ is the skew-symmetric part of the error matrix $\mathbf{E}$, and vex() maps skew-symmetric matrices to vectors. The Lyapunov function $V(\mathbf{E})$ is positive definite and monotonically decreasing, ensuring that the error converges to zero:

$$V(\mathbf{E}) > 0, \quad \forall \mathbf{E} \neq \mathbf{I} \tag{4.40a}$$

$$\dot{V}(\mathbf{E}) < 0 \tag{4.40b}$$

The benefits of leveraging symmetries in observer design are significant. By exploiting

these properties, the observer achieves global convergence of the error dynamics for many systems. In cases such as $SO(3)$, where topological constraints prevent full global stability, the observer can still achieve almost global convergence, meaning the error dynamics converge to zero for all initial conditions except for a set of measure zero (e.g., singularities).

Additionally, the symmetry-aware design inherently mitigates the effects of noise by ensuring that corrections are geometrically consistent with the system's structure. This approach distributes corrections evenly across measurements, preserving the system's integrity and improving robustness against disturbances and sensor noise. These properties make symmetry-based observers particularly effective for applications involving complex geometries and dynamic environments.

4.5.4 eXogeneous Kalman Filter

The eXogenous Kalman Filter (XKF) originally proposed by Johansen and Fossen [108] is a two-stage estimator that combines a global nonlinear observer (NLO) and a Linearized Kalman Filter (LKF). This approach seeks to leverage the complementary strengths of nonlinear observers and linearized Kalman filtering while mitigating their individual weaknesses, particularly in the context of state estimation for nonlinear systems.

The XKF is proposed to combine the global stability of NLOs with the optimality of Kalman filtering. By using the output of an NLO as an exogenous trajectory for the linearization in a second-stage LKF, the approach avoids the destabilizing feedback present in other nonlinear filters such as the EKF.

The XKF consists of two cascaded stages:

- **Nonlinear Observer**: The first stage is a global nonlinear observer that provides a globally stable state estimate $\hat{x}(t)$ and ensures boundedness of its estimation error, enabling robust initialization for the second stage.
- **Linearized Kalman Filter**: The second stage is a linearized Kalman filter that Uses the NLO's output as an exogenous trajectory for linearization and applies the Kalman filter to estimate deviations from the NLO trajectory.

The nonlinear system is described as:

$$\dot{\mathbf{x}}(\mathbf{t}) = f(\mathbf{x}(\mathbf{t}), \mathbf{t}) + \mathbf{G}(\mathbf{t})\mathbf{w}(\mathbf{t}), \tag{4.41a}$$

$$\mathbf{y}(\mathbf{t}) = h(\mathbf{x}(\mathbf{t}), \mathbf{t}) + \mathbf{e}(\mathbf{t}) \tag{4.41b}$$

where $\mathbf{x}(\mathbf{t})$ is the state vector, $\mathbf{y}(\mathbf{t})$ is the measurement vector, $f(\mathbf{x}(\mathbf{t}), \mathbf{t})$ is the nonlinear dynamics, $h(\mathbf{x}(\mathbf{t}), \mathbf{t})$ is the nonlinear measurement model, $\mathbf{G}(\mathbf{t})$ is the process noise matrix, $\mathbf{w}(\mathbf{t})$ is the process noise, and $\mathbf{e}(\mathbf{t})$ is the measurement noise.

In traditional methods like the EKF, the system is linearized about its own state estimate. This feedback loop can destabilize the filter, especially for systems with high nonlinearity. In the XKF, the system is linearized about the NLO's estimate $\hat{\mathbf{x}}(\mathbf{t})$, which is treated as an exogenous signal. This eliminates destabilizing feedback.

The XKF inherits the global stability properties of the NLO, which means that if the NLO achieves uniform global asymptotic stability or stronger properties (e.g., Globally Exponentially Stable (GES)), the cascade is globally stable. This is formalized through the cascade error dynamics, where the LKF operates on the error dynamics of the NLO.

4.5.4.1 Design of the XKF

The NLO generates a bounded estimate $\hat{\mathbf{x}}(\mathbf{t})$ using Lyapunov-based techniques. The system dynamic equations are:

$$\dot{\hat{\mathbf{x}}}(\mathbf{t}) = f(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t}) + \mathbf{L}(\mathbf{y}(\mathbf{t}) - h(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t})) \tag{4.42}$$

where L is the observer gain designed to ensure global stability of the error dynamics.

The error between the true state $x(t)$ and the observer estimate $\hat{\mathbf{x}}(\mathbf{t})$ is defined as $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$, and satisfies:

$$\dot{\tilde{\mathbf{x}}}(\mathbf{t}) = \mathbf{A}(\mathbf{t})\tilde{\mathbf{x}}(\mathbf{t}) + \mathbf{B}(\mathbf{t})\mathbf{w}(\mathbf{t}) + \mathbf{L}(\mathbf{t})\mathbf{e}(\mathbf{t}) \tag{4.43}$$

where $\mathbf{A}(\mathbf{t}) = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}(\mathbf{t})}$, $\mathbf{B}(\mathbf{t}) = \mathbf{G}(\mathbf{t})$, and $\mathbf{L}(\mathbf{t}) = \frac{\partial h}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}(\mathbf{t})}$ are the Jacobians of the system dynamics and measurement functions, respectively.

Then, the LKF operates on the error dynamics $\tilde{\mathbf{x}}(\mathbf{t})$, which represents the deviation from the NLO's trajectory. The LKF is a standard Kalman filter that estimates the error state $\tilde{\mathbf{x}}(\mathbf{t})$ and its covariance $\mathbf{P}(\mathbf{t})$ using the linearized system dynamics and measurement functions. The linearized dynamics are obtained by expanding $f(\mathbf{x}, \mathbf{t})$ and $h(\mathbf{x}, \mathbf{t})$ around the NLO's estimate $\hat{\mathbf{x}}(\mathbf{t})$:

$$\dot{\mathbf{x}} = f(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t}) + \mathbf{F}(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t})\tilde{\mathbf{x}} + \mathbf{G}(\mathbf{t})\mathbf{w}(\mathbf{t}) \tag{4.44a}$$

$$\mathbf{y} = h(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t}) + \mathbf{H}(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t})\tilde{\mathbf{x}} + \mathbf{e}(\mathbf{t}) \tag{4.44b}$$

where $F(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t}) = \frac{\partial f}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}(\mathbf{t})}$ and $\mathbf{H}(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t}) = \frac{\partial h}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}(\mathbf{t})}$ are the Jacobians of the system

dynamics and measurement functions, respectively. Then the correction is applied with the LKF updating the state deviation estimate $\tilde{\mathbf{x}}(\mathbf{t})$:

$$\dot{\tilde{\mathbf{x}}}(\mathbf{t}) = \mathbf{F}(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t})\tilde{\mathbf{x}}(\mathbf{t}) + \mathbf{G}(\mathbf{t})\mathbf{w}(\mathbf{t}) + \mathbf{K}(\mathbf{t})(\mathbf{y}(\mathbf{t}) - h(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t} - \mathbf{H}(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{t})\tilde{\mathbf{x}}(\mathbf{t}))) \quad (4.45)$$

where $\mathbf{K}(\mathbf{t})$ is the Kalman gain obtained from the Riccati equation.

In summary, the XKF demonstrates robustness by combining the NLO's ability to globally stabilize the system and the LKF's optimal noise rejection properties for small deviations. The XKF is particularly well-suited for systems with high nonlinearity, where the EKF may struggle due to linearization errors. The XKF is computationally efficient because the modular design of the XKF allows each stage (NLO and LKF) to be implemented independently, enabling parallel processing or optimized implementations suited to specific hardware, making it suitable for real-time applications. In fact, the NLO typically involves solving nonlinear differential equations, which can often be implemented using efficient numerical integration techniques (e.g., Euler or Runge-Kutta methods), while the LKF, operating on linearized deviations $\tilde{\mathbf{x}}$, has a computational effort similar to the standard Kalman filter, which is lightweight for real-time use.

4.5.5 Summary of the Theoretical Background

In this section, we have discussed part of the theoretical background of state estimation for nonlinear systems. We have discussed the Kalman filter, a powerful tool for linear systems, and its extensions for nonlinear systems, including the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), and Particle Filter (PF). We have also described the Nonlinear Observer, a global observer that ensures global stability of the error dynamics. Finally, we have presented the eXogenous Kalman Filter (XKF), a two-stage estimator that combines the global stability of the Nonlinear Observer with the optimality of the Kalman filter. Although EKF-based methods are widely adopted in robotic applications, they can struggle with strong nonlinearities and inaccurate initial conditions, sometimes leading to divergence or estimator "failure". For instance, in a previous work by Fink and Semini [34], the authors observed that an EKF-only approach could become unstable under large initialization errors, whereas the XKF maintained robust performance under the same conditions.

In the following sections, we will apply these theoretical concepts to the design of the MUSE state estimator for quadruped robots.

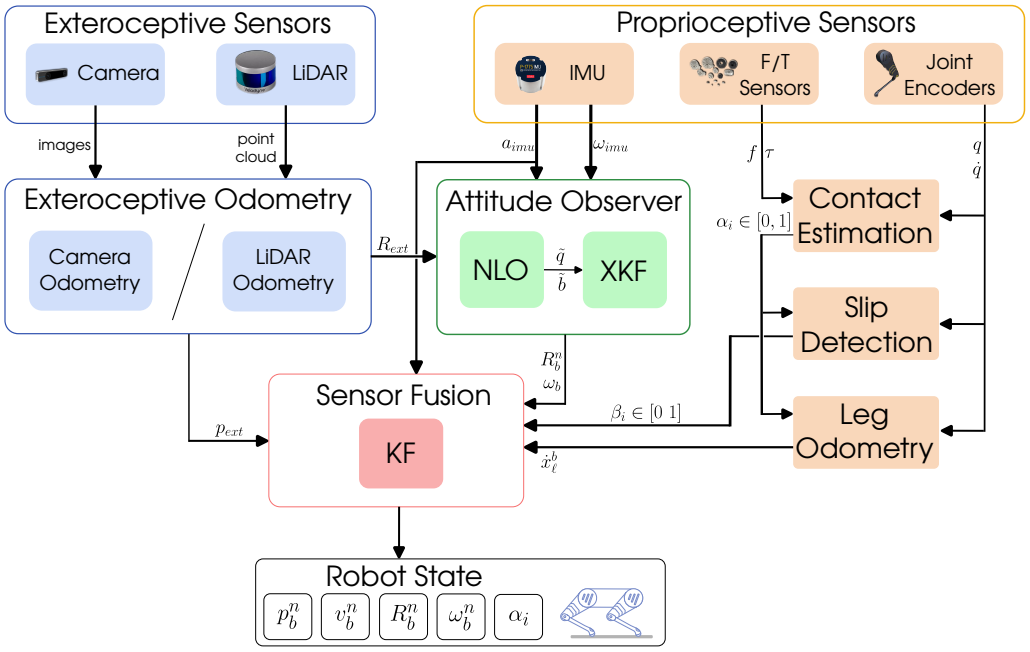


Figure 4.1: Overview of the MUSE state estimation pipeline.

4.6 MUSE Formulation

The goal of MUSE is to determine the pose, twist, and contact state of a quadruped robot with respect to an arbitrary inertial navigation frame. The robot is equipped with a suite of proprioceptive and exteroceptive sensors, including IMUs, force sensors, joint encoders, torque sensors, cameras, and LiDARs.

As illustrated in Fig. 4.1, MUSE combines the measurements from two exteroceptive sensors, Camera and LiDAR, with three proprioceptive sensors, IMU, force/torque sensors, and joint encoders. The system comprises several key components: exteroceptive odometry (Section 4.6.2), an attitude observer (Section 4.6.6), a contact estimation module (Section 4.6.3), a slip detection module (Section 4.6.5), a leg odometry module (Section 4.6.4), and a sensor fusion algorithm (Section 4.6.7).

- The attitude Observer includes a NLO and an XKF.
- The slip detection and leg odometry modules incorporates Joint State (JS), robot Kinematics/Dynamics (KD) model, and GRFs estimated from the dynamic model.
- The sensor fusion algorithm employs a KF to perform position and linear velocity estimation.

The following sections provide a detailed description of each component.

(a) The 21 kg **Aliengo robot** from Unitree [5].

(b) The 30 kg **ANYmal B300** robot from ANYbotics [43].

Figure 4.2: Robots used to test MUSE.

4.6.1 Robot Models

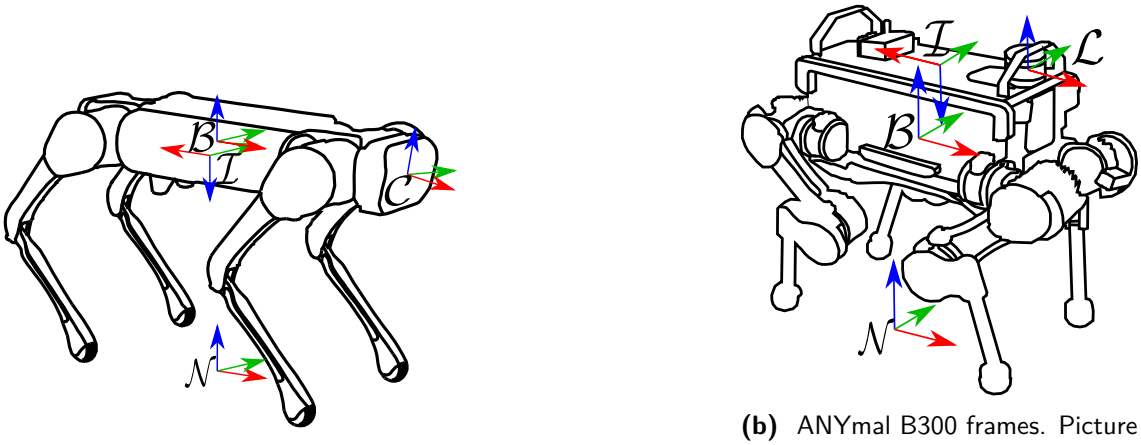
In this work, we utilized the dynamic and kinematic models of the 20 kg Aliengo robot, (Fig. 4.2a) and the 30 kg ANYmal B300, (Fig. 4.2b). However, the state estimator modules are designed to be generalizable and can be applied to any legged robot equipped with the proper sensors. The following reference frames are defined: the navigation frame $\mathcal{N}$, the body frame $\mathcal{B}$ which is located at the geometric center of the robot's trunk, with its basis oriented forward, left, and up, the IMU sensor frame $\mathcal{I}$, which is located at the origin of the accelerometer within the IMU mounted onto the trunk, the camera frame $\mathcal{C}$ for Aliengo (Fig. 4.3a), located at the optical center of the front-mounted camera, and the LiDAR frame $\mathcal{L}$ for ANYmal (Fig. 4.3b), located at the center of the sensor mounted on top of the robot.

The reference frame of a variable is indicated using a right superscript. For instance, $\mathbf{x}^n$, $\mathbf{x}^b$, $\mathbf{x}^i$, $\mathbf{x}^c$, and $\mathbf{x}^l$ represent the variable $\mathbf{x}$ expressed in the inertial frame $\mathcal{N}$, body frame $\mathcal{B}$, IMU frame $\mathcal{I}$, camera frame $\mathcal{C}$, and LiDAR frame $\mathcal{L}$, respectively.

The robots are equipped with a six-axis IMU mounted on the trunk, and each joint is outfitted with an absolute encoder. Additionally, Aliengo is equipped with front-mounted cameras, while ANYmal features torque sensors and a LiDAR.

The sensors provide measurements modeled as $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{b}_\mathbf{x} + \mathbf{n}_\mathbf{x}$, where $\mathbf{b}_\mathbf{x}$, and $\mathbf{n}_\mathbf{x}$ represent the bias and noise of $\mathbf{x}$, respectively. The biases are assumed to be either constant or slowly varying over time, while all noise components are modeled as Gaussian with zero mean (Simon [106]). The sensors are described as follows:

- Camera: for the indoor lab experiments with Aliengo, we used the Intel Realsense T265, a lightweight tracking camera. This device is equipped with an IMU and



(a) Aliengo frames.

(b) ANYmal B300 frames. Picture adapted from (Wisth et al. [113]).

Figure 4.3: Robot Reference Frames: the navigation frame $\mathcal{N}$, the body frame $\mathcal{B}$, the IMU sensor frame $\mathcal{I}$, the camera frame $\mathcal{C}$ for Aliengo, and the LiDAR frame $\mathcal{L}$ for ANYmal.

two fisheye lenses offering a 163° field of view. It is capable of delivering camera odometry at a frequency of up to 200 Hz, making it suitable for high-speed state estimation tasks.

- LiDAR: for experiments using the FSC Dataset with the ANYmal B300 robot, we employed the Velodyne VLP16 LiDAR as the sole external sensor. This LiDAR operates at a frequency of approximately 10 Hz, delivering reliable spatial data for state estimation.

- IMU: the IMU consists of a 3-DOFs gyroscope and 3-DOFs accelerometer. The accelerometer measures the specific force $\mathbf{f}_i^i = \mathbf{a}^i + \mathbf{g}^i \in \mathbb{R}^3$: where $\mathbf{a}^i \in \mathbb{R}^3$ is the acceleration of the body in $\mathcal{I}$ and $\mathbf{g}^i \in \mathbb{R}^3$ is the acceleration due to gravity in $\mathcal{I}$. The gyroscope measures angular velocity $\boldsymbol{\omega}^i \in \mathbb{R}^3$ in $\mathcal{I}$.

- Encoders and Torque sensors: the absolute encoders provide the joint position $\mathbf{q}_i \in \mathbb{R}$ and joint speed $\dot{\mathbf{q}}_i \in \mathbb{R}$, respectively. ANYmal is equipped with torque sensors that directly measure $\boldsymbol{\tau}_i \in \mathbb{R}^3$, while for Aliengo, the joint torque is estimated based on the motor current.

4.6.2 Exteroceptive Odometry

Since the primary goal of MUSE is to perform robust sensor fusion, we did not develop a dedicated exteroceptive odometry module. Instead, we leveraged odometry data provided by existing algorithms to determine sensor pose and velocity, which were then used to update the robot's state.

- **Lab Experiments with Aliengo:** Odometry data was obtained using the Intel Realsense T265 tracking camera. The T265 provides visual-inertial odometry at up to 200 Hz, allowing precise pose and velocity estimation in structured indoor environments.

- **FSC Dataset with ANYmal B300**: To obtain LiDAR odometry for the outdoor dataset, we used a loosely coupled method, relying on external LiDAR odometry that operates independently of the rest of the system. Specifically, we employed KISS-ICP (Vizzo et al. [9]), a LiDAR odometry algorithm renowned for its simplicity, efficiency, and robustness. KISS-ICP performs pose estimation by sequentially aligning LiDAR point clouds using a classical point-to-point Iterative Closest Point (ICP) algorithm. The method begins by applying a constant velocity motion model to deskew scans, correcting distortions caused by the sensor's motion during data capture. It then uses a voxel-based downsampling approach to reduce the computational load while preserving essential geometric features for alignment. To ensure accurate data associations, KISS-ICP incorporates an adaptive thresholding scheme that dynamically adjusts based on the system's observed motion profile. Finally, a robust optimization process refines the pose estimation, ensuring resilience to noise and outliers. This loosely coupled approach enables KISS-ICP to provide reliable pose updates, which integrate seamlessly into our system as an external source of LiDAR-based information.

By relying on these established odometry sources, we ensured accurate pose and velocity inputs for the state estimation pipeline without the need to implement additional odometry computation methods.

4.6.3 Contact Estimation

The contact estimation is executed exaclty as explained in Section 3.6, but some equations are reported here for clarity.

To estimate the foot contact with the ground, it is assumed that the contact point lies at a fixed location at the center of the foot. The contact state $\boldsymbol{\alpha} \in \mathbb{R}^4$ is estimated by computing the GRFs using the dynamics equation of motion:

$$\mathbf{M}(\bar{\mathbf{x}})\ddot{\bar{\mathbf{x}}} + \mathbf{h}(\bar{\mathbf{x}}, \dot{\bar{\mathbf{x}}}) = \bar{\boldsymbol{\tau}} + \mathbf{J}^\top \mathbf{F}_{\text{grf}} \tag{4.46}$$

where $\bar{\mathbf{x}} = [\mathbf{x}^\top \boldsymbol{\eta}^\top \mathbf{q}^\top]^\top \in \mathbb{R}^{18}$ is the generalized robot state, given by the position and attitude of the base, and the joint angles. Then $\dot{\bar{\mathbf{x}}} \in \mathbb{R}^{18}$ and $\ddot{\bar{\mathbf{x}}} \in \mathbb{R}^{18}$ are the

corresponding generalized velocities and accelerations, $\mathbf{M} \in \mathbb{R}^{18\times18}$ is the joint-space inertia matrix, $\mathbf{h} \in \mathbb{R}^{18}$ is the vector of Coriolis, centrifugal, and gravity forces, $\bar{\boldsymbol{\tau}} = [\underline{0}\ \boldsymbol{\tau}] \in \mathbb{R}^{18}$ where $\boldsymbol{\tau} \in \mathbb{R}^{12}$ is the vector of joint torques, and finally $\mathbf{F}_{\text{grf}} \in \mathbb{R}^{12}$ is the vector of GRFs, while $\mathbf{J} \in \mathbb{R}^{18\times12}$ is the floating base Jacobian.

Then, we solve for the GRFs $\mathbf{F}_{\text{grf}}$ of each leg using the actuated part of the dynamics:

$$\mathbf{F}_{\text{grf},\ell} = -\boldsymbol{\alpha}(\mathbf{J}^{\top}{}_{\ell}(\mathbf{q}_{\ell}))^{-1}(\boldsymbol{\tau}_{\ell} - \mathbf{h}_{\ell}(\bar{\mathbf{x}}_{\ell}, \dot{\bar{\mathbf{x}}}_{\ell})) \tag{4.47}$$

Finally, assuming that all external forces are applied to the feet during the *stance* phase, we first estimate the GRFs. Following this, the contact state $\boldsymbol{\alpha}_{\ell}$ for every leg ℓ is determined as:

$$\boldsymbol{\alpha}_{\ell} = \begin{cases} 1 & \text{if } \|\mathbf{F}_{\text{grf},\ell}\| > \mathbf{F}_{\text{min}} \\ 0 & \text{otherwise} \end{cases} \tag{4.48}$$

where $\mathbf{F}_{\text{min}} \in \mathbb{R}$ is the threshold value, and $\mathbf{F}_{\text{grf},\ell} \in \mathbb{R}^3$ is the GRF of the leg $\ell \in \mathbb{L}$.

4.6.4 Leg Odometry

Leg odometry estimates the incremental motion of the floating base using the forward kinematics of the legs that are in stable contact with the ground. This measurement can be expressed as either a relative pose or a velocity measurement. In our system, we specifically formulate linear velocity measurements. If there is no slippage, then the contribution of each leg $\ell \in \mathbb{L}$ to the overall velocity of the base is:

$$\mathbf{x}^{\boldsymbol{b}}_{\ell} = -\boldsymbol{\alpha}_{\ell}(\mathbf{J}_{\ell}(\mathbf{q}_{\ell})\dot{\mathbf{q}} - \boldsymbol{\omega}^{\boldsymbol{b}} \times \mathbf{x}^{\boldsymbol{b}}_{\ell}) \tag{4.49}$$

and the base velocity is:

$$\dot{\mathbf{x}}^{\boldsymbol{b}} = \frac{1}{\boldsymbol{n_s}} \sum_{\ell}^{\mathbb{L}} \dot{\mathbf{x}}^{\boldsymbol{b}}_{\ell} \tag{4.50}$$

where $\boldsymbol{n_s} = \sum_{\ell}^{\mathbb{L}} \boldsymbol{\alpha}_{\ell}$ is the number of stance legs.

4.6.5 Slip Detection

Leg odometry is prone to drift when the robot walks on slippery surfaces. To address this, we employ the slip-detection algorithm presented in Chapter 3 to identify when leg odometry measurements become unreliable due to slippage and compensate for the resulting drift.

For each leg ℓ, we use the flag $\boldsymbol{\beta}_{\ell} \in [0, 1]$, which is set to $\mathbf{1}$ if slippage is detected and

$\mathbf{0}$ otherwise. Once a single slippage or multiple slippages are detected, we increase the leg odometry covariance $\mathbf{R_1}$ in the sensor fusion algorithm (Section 4.6.7, Equation (4.60)) to reduce the influence of the leg odometry measurements. This adjustment ensures that errors in leg odometry do not adversely affect the base pose or velocity estimates.

4.6.6 Attitude Observer

To estimate the attitude, we implemented a cascaded structure composed of a NLO (Section 4.5.3) and an XKF (Section 4.5.4), where the XKF linearizes around a globally stable exogenous signal from the NLO. This cascade structure preserves the global stability properties of the NLO, while benefiting from the near-optimal properties of the Kalman Filter. The proof of stability is provided in (Johansen and Fossen [108]).

More in detail, for attitude estimation we specifically chose to represent the rotation using a quaternion, as it avoids singularities. The state is defined as $\mathbf{x} = [\mathbf{q}^\top \ \mathbf{b}^\top]^\top \in \mathbb{R}^7$ where $\mathbf{q} \in \mathbb{R}^4$ is the quaternion, and $\mathbf{b} \in \mathbb{R}^3$ represents the IMU's bias. The input to the system is $\mathbf{u} = \boldsymbol{\omega}^b \in \mathbb{R}^3$ which corresponds to the 3-axis gyroscope readings from the IMU. The dynamics of the filter are described by:

$$\dot{\mathbf{q}}_b^n = \frac{1}{2} \begin{bmatrix} \mathbf{0} & -(\boldsymbol{\omega}^\mathbf{b} - \boldsymbol{b}^b)^\top \\ (\boldsymbol{\omega}^b - \mathbf{b}^b)^\top & -\mathbf{S}(\boldsymbol{\omega}^b - \mathbf{b}^b) \end{bmatrix} \mathbf{q}_b^n \tag{4.51a}$$

$$\dot{\mathbf{b}}^b = \mathbf{0} \tag{4.51b}$$

where $\mathbf{S}(\cdot)$ is the skew-symmetric matrix function. We use a multiplicative error function, to respect the quaternion norm constraint $\mathbf{e}_q = (\mathbf{q}_b^n)^{-1} \otimes \hat{\mathbf{q}}_b^n$, where $\otimes$ is quaternion multiplication (Markley and Crassidis [114]).

The general vector of measurements is given by $\mathbf{z} = \mathbf{R}_n^b \mathbf{y}^n$, where $\mathbf{y}^n$ are a set of $\boldsymbol{k}$ constant references vector in $\mathcal{N}$:

$$\mathbf{y}^b = [(\mathbf{y}_\mathbf{1}^b)^\top \dots (\mathbf{y}_\boldsymbol{k}^b)^\top]^\top \in \mathbb{R}^{\mathbf{3}\boldsymbol{k}} \tag{4.52a}$$

$$\mathbf{y}^n = [(\mathbf{y}_\mathbf{1}^n)^\top \dots (\mathbf{y}_\boldsymbol{k}^n)^\top]^\top \in \mathbb{R}^{\mathbf{3}\boldsymbol{k}} \tag{4.52b}$$

4.6.6.1 Nonlinear Observer

In (Mahony et al. [107]) the authors introduced a class of nonlinear observers for attitude estimation that leverages the symmetry properties of the group structure to achieve strong convergence properties (Mahony et al. [112]). Many extensions of this work have been made, particularly the observer introduced by Grip et al. [115], which we utilize in

this system. The NLO is designed to exploit these symmetry properties to improve the robustness and stability of attitude estimation in dynamic environments. The observer is defined as:

$$\dot{\hat{\mathbf{R}}}_b^n = \hat{\mathbf{R}}_b^n \mathbf{S}(\boldsymbol{\omega}^b - \hat{\mathbf{b}}^b) + \sigma \mathbf{K}_p^n \mathbf{J}(\hat{\mathbf{R}}_b^n) \tag{4.53a}$$

$$\dot{\hat{\mathbf{b}}}^b = \text{Proj}\left(\hat{\mathbf{b}}^b, -k \,\text{vex}\left(\mathbb{P}\left(\hat{\mathbf{R}}_b^n \mathbf{K}_p^n \mathbf{J}(\hat{\mathbf{R}}_{bs}^n)\right)\right)\right) \tag{4.53b}$$

where $\mathbf{K_p} \in \mathbb{R}^{3\times3}$ is a symmetric positive-definite gain matrix, $k > 0 \in \mathbb{R}$ is a scalar gain, $\sigma \geq 1 \in \mathbb{R}$ is a scaling factor. The term $\hat{\mathbf{R}}_{bs}^n = \text{sat}(\hat{\mathbf{R}}_n^s)$, where the function $\text{sat}(\mathbf{X})$ saturates every element of $\mathbf{X}$ to lie within the range $[-1, 1]$. The function $\text{Proj}(x, y)$ is a parameter projection that ensures that $\|\hat{\mathbf{b}}\| < M_b$, where $M_b > 0 \in \mathbb{R}$ is a known constant upper bound on the gyro bias. Additionally, $\mathbb{P}(\mathbf{X}) = \frac{1}{2}(\mathbf{X} + \mathbf{X}^\top)$ is the symmetrization of any square matrix $\mathbf{X}$, and $\mathbf{J}$ is the stabilizing injection term

$$\mathbf{J}(\hat{\mathbf{R}}_b^n, t) = \sum_{j=1}^{k} \left(\mathbf{y}_j^n - \hat{\mathbf{R}}_b^n \mathbf{y}_j^b\right) \mathbf{y}_j^{b\top} \tag{4.54}$$

The observer is Globally Exponentially Stable (GES) for all initial conditions, assuming that there exists $k > 1$ non-collinear vector measurements, i.e.,

$$\left|\mathbf{y}_i^n \times \mathbf{y}_j^n\right| > 0 \tag{4.55}$$

where $i, j \in 1, \ldots, k$, and $\mathbf{y}_i^n, \cdots \mathbf{y}_j^n$ are the vector measurements. Furthermore, if there is only one measurement the observer is still GES if the following Persistency of Excitation (PE) condition holds:
if there exist constants $T > 0 \in \mathbb{R}$ and $\gamma > 0 \in \mathbb{R}$ such that, for all $t \geq 0$

$$\int_t^{t+T} \mathbf{y}_1^n(\tau) \mathbf{y}_1^n(\tau)^\top \, d\tau \geq \gamma \mathbf{I} \tag{4.56}$$

holds then $\mathbf{y}_1^n$ is PE. The proof of stability is given in (Grip et al. [115]).

4.6.6.2 eXogeneous Kalman Filter

The eXogeneous Kalman Filter (XKF) introduced by Johansen and Fossen [108], is similar to an EKF in that it linearizes a nonlinear model about an estimate of the state and then applies the typical Linear Time-Varying (LTV) Kalman filter (KF) to the linearized model. If the estimate is close to the true state, then the filter performs near-optimally. However, if the estimate deviates significantly from the true state, the filter can quickly diverge. To overcome this problem, the XKF linearizes around a globally

stable exogenous signal from a NLO. The cascaded structure preserves the global stability properties of the NLO while maintaining the near-optimal properties from the KF. The equations of the XKF are:

$$\dot{\hat{\mathbf{x}}} = \boldsymbol{f_x} + \mathbf{F}(\hat{\mathbf{x}} - \bar{\mathbf{x}}) + \mathbf{K}(\mathbf{z} - \boldsymbol{h_x} - \mathbf{H}(\hat{\mathbf{x}} - \bar{\mathbf{x}})) \tag{4.57a}$$

$$\dot{\mathbf{P}} = \mathbf{FP} + \mathbf{PF}^\top - \mathbf{KHP} + \mathbf{Q} \tag{4.57b}$$

$$\mathbf{K} = \mathbf{PH}^\top \mathbf{P}^{-\mathbf{1}} \tag{4.57c}$$

where $\mathbf{F} = \frac{\partial f_x}{\partial x}|_{\bar{\mathbf{x}},\mathbf{u}}$, $\mathbf{H} = \frac{\partial h_x}{\partial x}|_{\bar{\mathbf{x}},\mathbf{u}}$, $\bar{\mathbf{x}} \in \mathbb{R}^n$ is the bounded estimate of $\mathbf{x}$ from the globally stable NLO. The measurement vector is $\mathbf{z} \in \mathbb{R}^6$, and further considerations are necessary to understand how we obtained it.

In ideal conditions, a 3-axis accelerometer and a 3-axis magnetometer provide the measurements (feedback) to the filter. While the magnetometer can be used to estimate the orientation of an object relative to the Earth's magnetic field, there are some limitations when using a magnetometer for determining the orientation of a quadruped robot. Magnetometers are sensitive to local magnetic fields, which can be influenced by nearby electric motors, high currents, and metallic objects (such as buildings) in the environment where the robots operate. To address these challenges, we adopted an alternative strategy to obtain the measurement vector. In a typical scenario, the magnetometer would provide a vector aligned with the Earth's magnetic north. However, instead of relying on the magnetometer, we propose a "*pseudo-north*" strategy using the exteroceptive sensors (LiDAR or camera) for external odometry. This approach provides the rotation from the sensor local frame ($\mathcal{S}$) to the navigation frame ($\mathcal{N}$). We used a constant vector, which remains fixed in the navigation frame $\mathcal{N}$, and rotated it by the amount given by the sensor's orientation. We then further rotated this measurement into the body frame $\mathcal{B}$. In the end, the measurement vector is obtained as:

$$\mathbf{z} = [\mathbf{f}_b^\top \mathbf{m}_b^\top]^\top \in \mathbb{R}^6 \tag{4.58}$$

where $\mathbf{f}_b = \mathbf{R}_i^b \mathbf{f}_n \in \mathbb{R}^3$ is the acceleration given by the accelerometer rotated in $\mathcal{B}$, and $\mathbf{m}_b = \mathbf{R}_s^b \mathbf{R}_n^s [\mathbf{1}\ \mathbf{0}\ \mathbf{0}]^\top \in \mathbb{R}^3$ is the "pseudo-magnetometer measure", in which $[\mathbf{1}\ \mathbf{0}\ \mathbf{0}]^\top$ is a constant vector in $\mathcal{N}$ pointing to a "pseudo" North, rotated in $\mathcal{B}$.

4.6.7 Sensor Fusion

The inertial measurements are fused with the leg odometry and the camera or LiDAR odometry. Decoupling the attitude from position and linear velocity provides a significant benefit: the resulting dynamics become LTV, ensuring inherent stability properties. In other words, the filter is designed in such a way that it will not diverge within a finite timeframe. The KF has the following dynamics:

$$\dot{\hat{\mathbf{x}}} = \boldsymbol{f}_{\boldsymbol{x}} + \mathbf{K}(\mathbf{z} - \boldsymbol{h}_{\boldsymbol{x}}) \tag{4.59a}$$

$$\dot{\mathbf{P}} = \mathbf{F}\mathbf{P} + \mathbf{P}\mathbf{F}^{\top} - \mathbf{K}\mathbf{H}\mathbf{P} + \mathbf{Q} \tag{4.59b}$$

$$\mathbf{K} = \mathbf{P}\mathbf{H}^{\top}\mathbf{R}^{-1} \tag{4.59c}$$

where the state $\mathbf{x} = \left[\mathbf{x}^{n\top}\mathbf{v}^{n\top}\right]^{\top} \in \mathbb{R}^6$ represents the position and velocity of the base, the input $\mathbf{u} = (\mathbf{R}_b^n\mathbf{f}_i^b - \mathbf{g}^n) \in \mathbb{R}^3$ correspond to the acceleration of the base, and the vector $\mathbf{z}$ denotes the measurement vector. The dimensions of $\mathbf{z}$ vary depending on the specific measurements. In the case of indoor experiments with Aliengo, where the T265 camera is the sole external sensor, $\mathbf{z}$ has a dimension of 9. This is because the T265 outputs both pose and twist. In this case that, $\mathbf{z} = \left[\mathbf{R}_b^n\dot{\mathbf{x}}_\ell^{b\top}\ \mathbf{R}_b^n\dot{\mathbf{x}}_c^{b\top}\ \mathbf{R}_b^n\mathbf{x}_c^{b\top}\right]^{\top} \in \mathbb{R}^9$, where $\mathbf{R}_b^n\dot{\mathbf{x}}_\ell^{b\top}$ represents the leg odometry (base velocity), and $\mathbf{R}_b^n\dot{\mathbf{x}}_c^{b\top}$ and $\mathbf{R}_b^n\mathbf{x}_c^{b\top}$ are the camera's velocity and position, respectively, rotated into the navigation frame $\mathcal{N}$. On the other hand, on the FSC Dataset, since KISS-ICP outputs only the pose, the measurement vector is $\mathbf{z} = \left[\mathbf{R}_b^n\dot{\mathbf{x}}_l^{b\top}\ \mathbf{R}_b^n\mathbf{x}_l^{b\top}\right]^{\top} \in \mathbb{R}^6$, where $\mathbf{x}_l^b$ is the position of the LiDAR in the body frame $\mathcal{B}$. The Kalman gain $\mathbf{K}$ is a matrix $\in \mathbb{R}^{6\times9}$ when all the measurements are available, or $\in \mathbb{R}^{6\times6}$ when the sensor velocity is not available.

$\mathbf{P} \in \mathbb{R}^{6\times6}$ is the covariance matrix, and $\mathbf{Q} \in \mathbb{R}^{6\times6}$ is the process noise. The measurement noise covariance matrix is a diagonal block matrix, assuming that the measurements are uncorrelated:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R_1} & \mathbf{0_3} & \mathbf{0_3} \\ \mathbf{0_3} & \mathbf{R_2} & \mathbf{0_3} \\ \mathbf{0_3} & \mathbf{0_3} & \mathbf{R_3} \end{bmatrix} \quad \text{or} \quad \mathbf{R} = \begin{bmatrix} \mathbf{R_1} & \mathbf{0_3} \\ \mathbf{0_3} & \mathbf{R_2} \end{bmatrix} \tag{4.60}$$

where $\mathbf{R_1} \in \mathbb{R}^{3\times3}$ is the covariance of the leg odometry and its values are updated in case of slippage. $\mathbf{R_2} \in \mathbb{R}^{3\times3}$ is the covariance of the exteroceptive sensor velocity measurement (when available), and $\mathbf{R_3} \in \mathbb{R}^{3\times3}$ is the covariance of the exteroceptive

sensor position measurement. Then

$$\boldsymbol{f}_x = \begin{bmatrix} \mathbf{v}^n \\ \mathbf{u} \end{bmatrix} \quad \text{and} \quad \mathbf{F} = \begin{bmatrix} \mathbf{0_3} & \mathbf{I_3} \\ \mathbf{0_3} & \mathbf{0_3} \end{bmatrix} \tag{4.61}$$

where $\mathbf{I_3} \in \mathbb{R}^{3\times3}$ and $\mathbf{0_3} \in \mathbb{R}^{3\times3}$ are the identity matrix and null matrix, respectively. For the same reason previously explained, the matrix $\mathbf{H} \in \mathbb{R}^{6\times9}$ or $\mathbf{H} \in \mathbb{R}^{6\times6}$ is:

$$\mathbf{H} = \begin{bmatrix} \mathbf{0_3} & \mathbf{I_3} \\ \mathbf{0_3} & \mathbf{I_3} \\ \mathbf{I_3} & \mathbf{0_3} \end{bmatrix} \quad \text{or} \quad \mathbf{H} = \begin{bmatrix} \mathbf{0_3} & \mathbf{I_3} \\ \mathbf{I_3} & \mathbf{0_3} \end{bmatrix} \tag{4.62}$$

4.6.8 Considerations about time execution

To maintain efficient computation despite the slower arrival of exteroceptive measurements, we rely on internal measurements, such as the IMU and joint states, for attitude estimation (Section 4.6.6) and sensor fusion (Section 4.6.7). The other corrections are applied only when exteroceptive data becomes available.

We analyzed and measured the execution time of each key block within the system. The results show an average execution time of 0.05 ms per block, demonstrating MUSE's ability to process and deliver state information with high speed and precision. The evaluated blocks include: ▪ Contact estimation (Section 3.6) ▪ Leg Odometry (Section 4.6.4) ▪ Slip Detection (Section 4.6.5) ▪ Attitude Estimation (Section 4.6.6) ▪ Sensor Fusion (Section 4.6.7). With the total of the blocks completing in just 0.05 ms, these results validate MUSE's capacity for real-time operation, making it ideal for environments that require low-latency and accurate state information.

Our approach is particularly suited for real-time performance because it avoids optimization over past timeframes, in contrast to optimization-based methods. Instead, the modular structure of our system allows for low-latency, efficient state estimation. For instance, the attitude observer primarily uses high-frequency IMU data, ensuring continuous state estimation. When slower exteroceptive data, such as from LiDAR or a camera, becomes available, it serves to refine the estimate without interrupting ongoing high-frequency operations. Meanwhile, the filter operates solely with IMU data, and the stability properties of our NLO and XKF ensure short-term stability even without exteroceptive updates. This modularity eliminates bottlenecks by prioritizing fast IMU measurements, maintaining stability via NLO and XKF, even when exteroceptive data is unavailable. Similarly, in the sensor fusion module, we rely on the high-frequency data

from the IMU and leg odometry (via encoders) to maintain real-time state estimation. When additional data from the camera or LiDAR arrives, it enhances the estimate without disrupting the high-frequency process. This approach ensures robust real-time performance, particularly crucial for closed-loop control, where speed and stability are critical.

Furthermore, the real-time capabilities of MUSE were demonstrated in a closed-loop experiment with the Aliengo robot in Section 4.7.2.1, where an MPC+PD controller operated at 100 Hz and 1000 Hz, respectively. MUSE successfully provided real-time feedback to the controller on the robot's linear velocity and orientation, enabling the robot to navigate challenging environments, including stairs, rocks, and slippery terrain.

4.7 Experimental Results

In this section, we present the experimental results obtained with the MUSE state estimator. We begin with an offline evaluation, where we analyze the performance of the state estimator in a controlled lab environment (Sections 4.7.1.1 and 4.7.1.2). Next, we move on to online evaluation, where we close the loop with a controller (Section 4.7.2.1). For this first set of experiments, ablation studies are done by comparing the performance of MUSE to those obtained with an Intel RealSense T265. Finally, to provide a comprehensive evaluation, we benchmark MUSE against state-of-the-art state estimators using the FSC dataset with the ANYmal B300 robot (Section 4.7.2.2). This ensures that MUSE's performance is rigorously compared with leading methods in the field.

The accuracy of the estimated trajectories is evaluated using the Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) statistics, which are computed with the EVO Python package (Grupp [116]). The ATE represents the average Euclidean distance between the estimated and ground truth trajectories, while the RPE measures the Euclidean distance and angular difference between the estimated and true poses at each time step, quantifying the system's accuracy in tracking changes in both position and orientation (Zhang and Scaramuzza [117]).

4.7.1 First results: Offline evaluation

In the first set of experiments, we assessed the offline performance of the MUSE state estimator in a controlled lab environment. The Aliengo robot performed various tasks, including walking up and down stairs (Section 4.7.1.1) and traversing rocky and slippery terrain (Section 4.7.1.2). For these experiments, we utilized visual odometry from an Intel Realsense T265 binocular visual-inertial tracking camera, which is already

embedded in the robot.

The performance of the state estimator was evaluated by comparing its estimates with ground truth data, which was collected using a Vicon motion capture system. This system provided the robot's true pose for evaluation purposes. The sensor frequencies were 250 Hz for the IMU and leg kinematics, and 200 Hz for the camera. The MUSE state estimator operates at a frequency of 250 Hz.

4.7.1.1 Aliengo walking up and down stairs

In the first experiment, the Aliengo robot used a crawling gait to navigate up and down stairs in our lab at IIT in Genoa (see Fig. 4.4). For this experiment, we utilized the IMU data and camera orientation as measurements for the XKF (Section 4.6.6), while linear velocity from the leg odometry, and both linear position and velocity data from the camera as measurements for the sensor fusion KF (Section 4.6.7).

We compared the results of our state estimator with the Vicon ground truth data, and we benchmarked the performance with those obtained using only the T265 camera, which is widely used as a standalone state estimator in other works (for instance by Bayer and Faigl [118]). The goal of this experiment was to demonstrate that by integrating leg odometry and an attitude observer with the camera odometry, we could improve the robot's state estimation while also increasing robustness through the use of multiple sensor modalities.

The results of the experiment are shown in Fig. 4.5, and Tab. 4.1. Fig. 4.5a is particularly significant as it highlights how MUSE corrected the substantial error in the z-position of the T265 estimate. Tab. 4.1 further supports these improvements, showing enhanced ATE and RPE performance.

The ATE and RPE statistics were calculated over a 5-meter trajectory, with the ATE evaluated over a 1-meter segment. The results clearly demonstrate that the MUSE state estimator outperforms the T265 camera in both ATE and RPE statistics. Notably, MUSE also achieves a lower RPE for orientation, with a value of 6.529° compared to 7.792° for the T265 camera. Additionally, the MUSE state estimator operates at a frequency of 250 Hz, while the T265 camera operates at 200 Hz.

Table 4.1: Aliengo climbing stairs: ATE and RPE over 1 m ($\sim$ 5 m trajectory)

	ATE [m]	RPE [m]	RPE [°]	Freq [Hz]
T265	0.089	0.081	7.792	200
MUSE	**0.045**	**0.077**	**6.529**	**250**

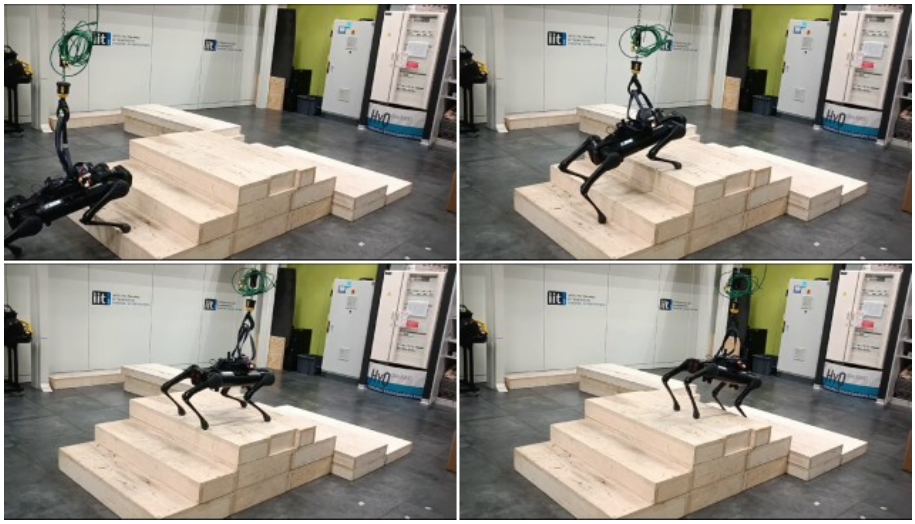
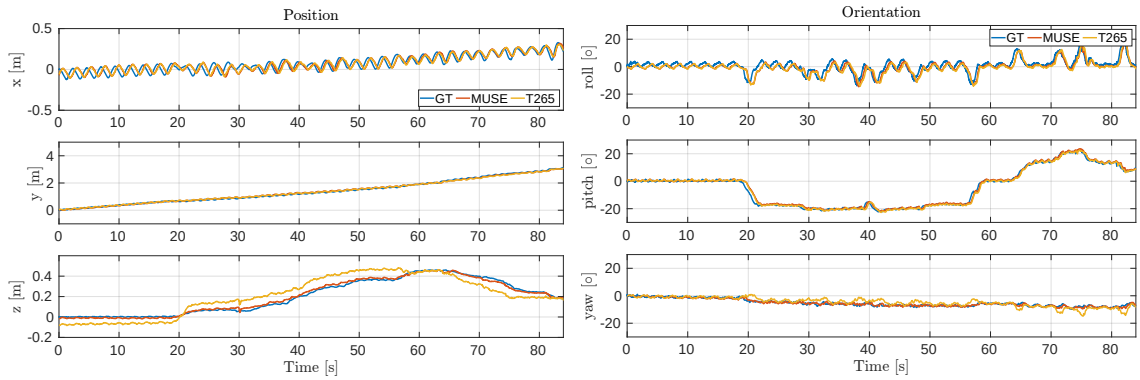


Figure 4.4: Aliengo climbing stairs



(a) Aliengo on stairs: base position

(b) Aliengo on stairs: base orientation

Figure 4.5: Aliengo climbing stairs: Comparison of the Ground Truth (GT) vs. position and orientation estimated by MUSE vs. the only T265 camera

4.7.1.2 Aliengo walking on uneven and slippery terrain

In the second experiment performed in our lab at IIT, the Aliengo robot used a crawling gait to traverse uneven terrain consisting of a pile of rocks, followed by a white plastic sheet coated with liquid soap (Fig. 4.6). The robot was equipped with the same sensors as in the previous experiment, and the same measurements were used for the XKF and the sensor fusion KF.

In this experiment, both the T265 camera odometry and the slip detection module contributed together to compensate for the drift in the leg odometry, making the improvement provided by the slip detection less noticeable (Fig. 4.7). The ATE and RPE statistics for both the T265 camera alone and the complete pipeline are shown in Tab. 4.2. To better highlight the benefits of using slip detection, we deactivated the camera and relied solely on proprioceptive measurements (leg odometry and IMU) along with the slip detection module (Slip Detection (SD)). The version of MUSE using only proprioceptive data is referred to as "P-MUSE". When the slip detection module was deactivated, this configuration was called "P-MUSE no SD". In this experiment, starting from approximately 6.5 seconds, at least one of the robot's legs was constantly in contact with uneven or slippery terrain. In Tab. 4.2, the upper rows display the comparison of estimates between the T265 camera and the complete MUSE pipeline. The lower rows show a comparison of the proprioceptive MUSE estimates, both with and without the slip detection module (P-MUSE and P-MUSE no SD, respectively). The results in Tab. 4.2 demonstrate that slip detection enhances estimation accuracy, especially when camera data is unavailable.

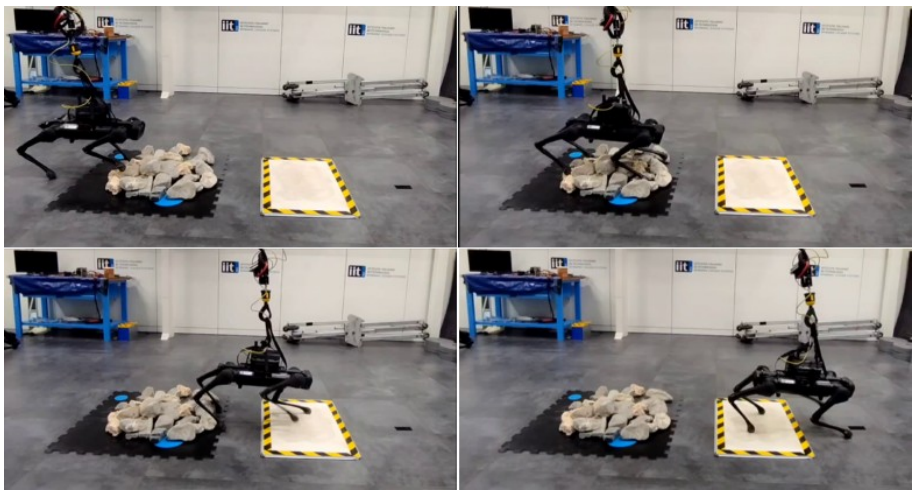
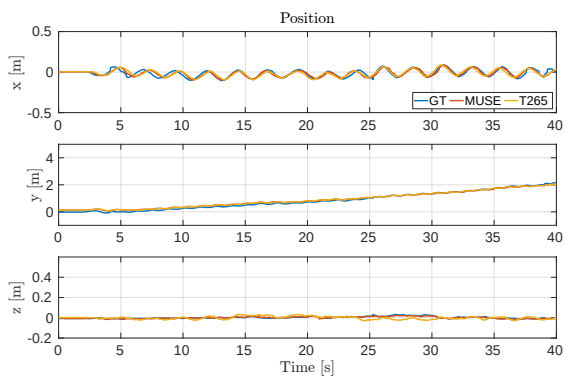
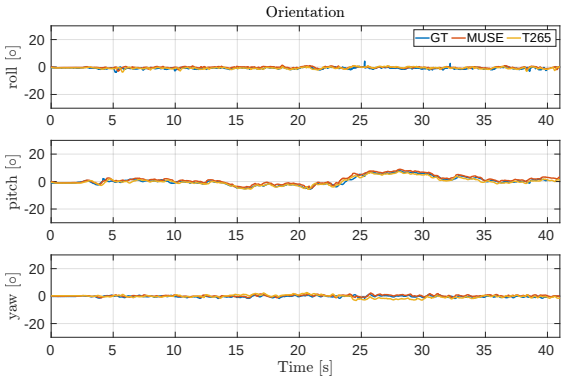


Figure 4.6: Aliengo walking on uneven and slippery terrain



(a) Aliengo on uneven and slippery terrain: base position



(b) Aliengo on uneven and slippery terrain: base orientation

Figure 4.7: Aliengo walking on uneven and slippery terrain: Comparison of the GT vs. position and orientation estimated by MUSE and the only T265 camera.

Table 4.2: Aliengo walking on uneven and slippery terrain: ATE and RPE over 1 m ($\sim$ 5 m trajectory)

	ATE [m]	RPE [m]	RPE [o]	Freq [Hz]
T265	0.097	0.100	2.151	200
MUSE	**0.096**	**0.092**	**1.897**	**250**
P-MUSE no SD	0.912	0.893	1.903	250
P-MUSE	0.425	0.120	1.899	250

4.7.2 Main results: Online evaluation and Benchmarking

In this section, we present the results obtained from two different robotic platforms: Aliengo tested in an online lab experiment walking a longer trajectory (Section 4.7.2.1), and ANYmal B300 evaluated on a pre-recorded outdoor dataset (Section 4.7.2.2). Since our state estimator functions as an odometry system, no loop closures (i.e., processes to recognize previously visited locations to mitigate drift) have been executed on the estimated trajectory.

4.7.2.1 Online evaluation: Closing the loop with the controller

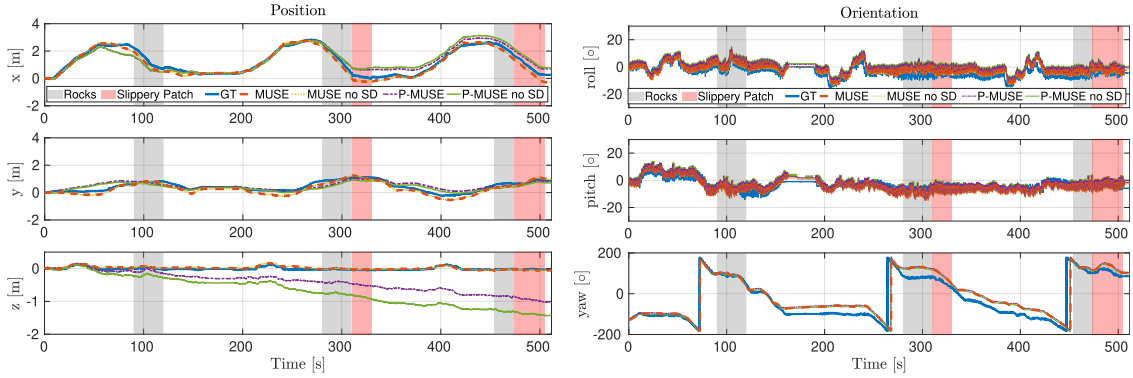
Figure 4.8: Aliengo in a closed-loop experiment: During the closed-loop experiment, Aliengo walked up and down the stairs, then on rocks and slippery terrain, repeating these tasks three times.

This test involved a closed-loop experiment with the Aliengo robot, where it navigated difficult terrain using a crawl gait, over a trajectory of approximately 45 meters, all while being commanded by a joystick. During this experiment, the robot completed three laps around the lab, traversing stairs, rocks, and slippery terrain (Fig. 4.8).

The controller used in this experiment is the Model Predictive Controller (MPC) described in (Amatucci et al. [119]). The MPC receives base pose and velocity inputs from MUSE and generates torque commands for the joint Proportional-Derivative (PD) controllers of the robot. The MPC operates at a frequency of 100 Hz, while the PD controllers run at 1000 Hz. Real-time communication was managed using the software framework described in (Bertol et al. [120]), and in Appendix A. The pipeline was executed on an Intel NUC i7 with 32 GB of memory. In this setup, the IMU and leg kinematics have an acquisition frequency of 1000 Hz, while the camera odometry runs at 200 Hz, and MUSE operates at 1000 Hz. The average execution time for each

module within MUSE is 0.05 milliseconds, ensuring efficient processing and real-time state updates.

For MUSE, the inputs to the XKF included the camera orientation $\mathbf{R}_c^n \in \mathbb{R}^{3\times3}$ and IMU acceleration $\mathbf{f}_b \in \mathbb{R}^3$. The linear velocity $\dot{\mathbf{x}}_b^n \in \mathbb{R}^3$ from the leg odometry, linear position $\mathbf{x}_c^n \in \mathbb{R}^3$ and velocity $\dot{\mathbf{x}}_c^n \in \mathbb{R}^3$ from the camera, along with the estimated orientation $\mathbf{R}_b^n \in \mathbb{R}^{3\times3}$ from the XKF were used as inputs for Sensor Fusion (SF) module.



(a) Aliengo in the closed-loop experiment: base position.

(b) Aliengo in the closed-loop experiment: base orientation.

Figure 4.9: Aliengo on uneven and slippery terrain: Comparison of position and orientation estimations between the GT and MUSE, MUSE without the slip detection module (MUSE with no SD), Proprioceptive MUSE (P-MUSE), and Proprioceptive MUSE without the slip detection module (P-MUSE with no SD). The grey shaded areas indicate that the robot is walking on rocks, while the red ones indicate when the robot is walking on the slippery patch. The position plot (left) clearly shows that the drift is higher when SD is not active.

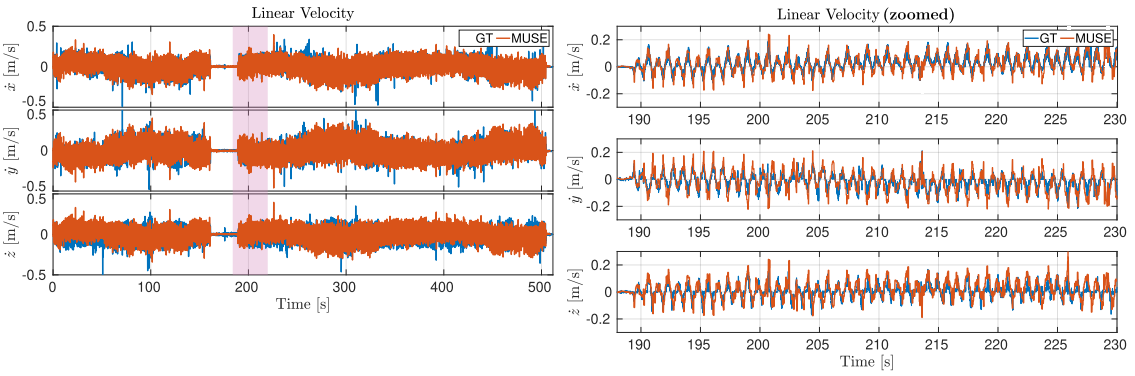


Figure 4.10: Aliengo on uneven and slippery terrain: GT vs. Linear Velocity estimated by MUSE during the closed-loop experiment. Left: linear velocity during the entire experiment. Right: Zoom on the time interval [185-230] seconds (pink shaded area in the left plot).

Table 4.3: Aliengo in a closed loop experiment: ATE and RPE over 1 m (∼ 45 m trajectory)

	T265	MUSE	MUSE no SD	P-MUSE	P-MUSE no SD
ATE [m]	**0.24**	**0.24**	0.25	0.57	0.67
RPE [m]	0.10	**0.08**	0.09	0.10	0.12
RPE [○]	0.35	**0.25**	0.26	0.27	0.27
Freq [Hz]	200	**1000**	**1000**	**1000**	**1000**

As shown in Figs. 4.9a and 4.9b, the position and orientation estimates provided by MUSE closely match the ground truth. Exteroceptive measurements improved the robot's state estimation, particularly in correcting drift, which is commonly observed along the vertical direction (z-axis) when relying solely on proprioception. Camera odometry, in particular, compensates for drift when the robot is walking on uneven terrain (Fig. 4.9a), making the benefits of the slip detection (SD) module less pronounced. However, the importance of SD becomes evident when using the proprioceptive-only version of MUSE (P-MUSE), as shown in Fig. 4.9a. In these cases, the SD module helps partially correct position drift during slippage, as leg odometry becomes unreliable in such situations.

Additionally, since the MPC controller requires the robot's linear velocity as feedback, Fig. 4.10 shows that the estimated linear velocity tracks the ground truth effectively. While the signal may appear noisy at first glance, this is due to the natural swaying motion of the robot during slow walking. The variations reflect the true velocity signal, not noise, capturing the dynamics of the gait. The zoomed-in view (highlighted in yellow) provides a closer look at these variations, illustrating that the tracking performance remains accurate despite the robot's swaying motion.

Tab. 4.3 presents the ATE and RPE statistics over 1 meter segment of the 60 meters trajectory. These results confirm that the MUSE pipeline delivers accurate position estimates. For this long trajectory, the ATE is comparable to the T265 camera, but MUSE operates at a higher frequency and shows lower RPE in both translation and orientation. The advantage of the SD module is evident in P-MUSE, where the ATE increases when SD is not utilized. Furthermore, it is important to highlight that the yaw angle is accurately estimated even when using only proprioceptive sensors, thanks to the globally stable Attitude Observer, which ensures bounded orientation errors and prevents filter divergence within a finite time frame.

4.7.2.2 Offline evaluation and benchmarking: FSC Dataset with ANYmal B300

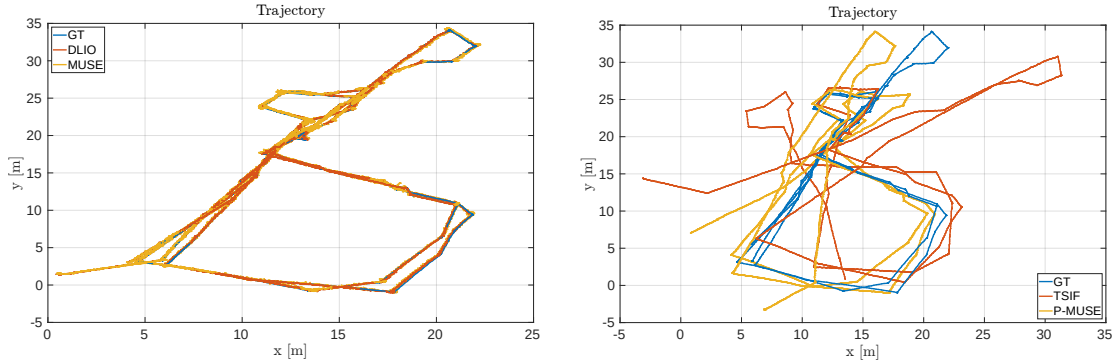
This section presents the results obtained by running MUSE on the Fire Service College Dataset (FSC) (Wisth et al. [2]). The Fire Service College, located in the United Kingdom, is a firefighting training facility, and one of its test areas represents a simulated industrial oil rig with a total dimension of 32.5m $\times$ 42.5m.

In the experiment, ANYmal trotted at a speed of 0.3 m/s, completing three laps before returning to the initial position, covering a total distance of 240 meters over 33 minutes. The environment posed significant challenges due to the presence of standing water, oil residue, gravel, and mud. For this experiment, results were obtained using LiDAR as the exteroceptive sensor. The orientation from LiDAR odometry was used as an external measurement for the attitude estimate (Section 4.6.6), while the position estimate was used in the sensor fusion KF (Section 4.6.7).

The ground truth (GT) trajectory was obtained with millimeter accuracy by combining the absolute positions taken from a *Leica Total Station T16*, with a SLAM system based on ICP registration and IMU data.

We computed the ATE over the entire 240 m trajectory, and RPE every 10 meters segments. The performance in terms of ATE and RPE was benchmarked against other state-of-the-art state estimators: DLIO (Chen et al. [8]), a LiDAR-inertial odometry algorithm, and three state estimators tailored for quadruped robots, Pronto (Camurri et al. [6]), VILENS (Wisth et al. [2]) and Two-State Information Filter (TSIF) (Bloesch et al. [7]). The values for Pronto and VILENS on the FSC Dataset are taken from their respective papers, while the data for TSIF was provided along with the dataset. Pronto and VILENS fuse exteroceptive and proprioceptive measurements, while TSIF relies solely on proprioceptive data. All of these systems are odometry systems that do not utilize loop closures. Results are shown in Tab. 4.4.

Compared to DLIO, MUSE achieved a similar ATE and translational RPE, with only a 3 cm and 2 cm difference, respectively. However, MUSE demonstrated a lower rotational RPE and operated at a higher frequency. Specifically for this experiment, MUSE runs at 400 Hz, as both leg kinematics and the IMU run at that frequency, whereas DLIO operates at an average of 100 Hz. Although incorporating leg kinematics introduces slightly higher ATE due to noisy leg odometry, it enhances the estimator's robustness and speed in terms of frequency. Notably, the fusion of different sensor modalities helps compensate for the limitations of individual sensors. While fusion does not always guarantee more accuracy than using a single sensor, it provides a more robust estimation



(a) FSC-Dataset: Ground-truth trajectory (blue) vs. estimated trajectories using DLIO and MUSE.

(b) FSC-Dataset: Ground-truth trajectory (blue) vs. estimated trajectory using TSIF and P-MUSE.

Figure 4.11: Trajectory of the FSC Dataset: Comparison of the trajectory estimated using MUSE, P-MUSE, and two state-of-the-art state estimators: DLIO and TSIF.

process by allowing each sensor to compensate for potential failures or inaccuracies in others. Additionally, sensor fusion enables the estimator to achieve higher frequencies by relying on high-frequency inputs.

When compared to Pronto and VILENS, MUSE outperforms both in terms of translational RPE, with improvements of 67.6% and 26.7%, respectively. The rotational RPE is similar to that of VILENS, although Pronto paper (Camurri et al. [6]) does not provide a rotational error metric, nor do either of these systems provide ATE data. When comparing proprioceptive-only state estimators, P-MUSE and TSIF, our algorithm demonstrates greater accuracy in terms of ATE, reducing the mean error by nearly 50%. This is the most significant metric for evaluating overall trajectory discrepancy, reflecting global accuracy. The rotational RPE is similar between P-MUSE and TSIF, but P-MUSE shows a slightly higher translational RPE. This indicates that while TSIF captures short-term movements with higher precision, small errors accumulate over time, resulting in inferior global accuracy compared to P-MUSE.

	LiDAR-inertial	**Legged robots (multi-sensor)**				**Legged robots (proprio)**		
	DLIO	Pronto	VILENS	MUSE	MUSE no SD	TSIF	P-MUSE	P-MUSE no SD
ATE [m]	0.14	N.A	N.A.	**0.17**	0.18	4.40	**2.38**	2.57
RPE [m]	0.09	0.34	0.15	**0.11**	0.12	**0.05**	0.12	0.15
RPE [◦]	1.9	N.A.	**1.14**	1.78	1.85	1.96	**1.93**	1.96
Freq [Hz]	100	**400**	**400**	**400**	**400**	**400**	**400**	**400**

Table 4.4: FSC Dataset: ATE and RPE over 10 m ($\sim$ 240 m trajectory). The first column is DLIO, then we have multi-sensor state estimators tailored for legged robots (Pronto, VILENS and MUSE), and on the right part of the table, there are proprioceptive state estimators for legged robots (TSIF and MUSE). The bold values indicate the best performance achieved by the legged robot state estimators.

Visual comparisons of the ground truth and estimated trajectories are shown in Fig. 4.11. In Fig. 4.11a, we can see that DLIO and MUSE trajectories closely overlap with the ground truth, while in Fig. 4.11b, it is evident that our proprioceptive pipeline outperforms TSIF in terms of global accuracy.

4.8 Discussion

MUSE is a modular state estimator for legged robots that fuses proprioceptive and exteroceptive sensor data. The results show that MUSE is capable of providing accurate and robust state estimation for legged robots in various environments. The modular structure of MUSE allows for the integration of different sensor modalities, enabling the system to compensate for the limitations of individual sensors and improve overall estimation accuracy. The real-time capability of MUSE was demonstrated in the closed-loop experiment with the Aliengo robot, where MUSE provided real-time feedback to the controller on the robot's linear velocity and orientation. The results of the benchmarking experiment on the Fire Service College dataset show that MUSE outperforms other state estimators in terms of global and local accuracy, demonstrating the effectiveness of the proposed approach.

We want to highlight that the primary contribution of our approach is not just the inclusion of additional odometry from the camera (T265) or the LiDAR-based algorithm (KISS-ICP), but rather the fusion of multiple sensor modalities into a comprehensive state estimator specifically designed for legged robots, which can handle challenging environments, such as uneven and slippery terrain. While previous works such as Pronto (Camurri et al. [6]) or STEP (Kim et al. [4]) also utilized exteroceptive sensors (cameras or LiDAR) for state estimation, they lack a dedicated slip detection module. These systems address the problem of slippage indirectly through their exteroceptive sensors, but our method provides a more robust solution by directly integrating a slip detection module into the state estimator. The integration of slip detection is crucial for scenarios where exteroceptive sensors (camera or LiDAR) may fail, be obstructed, or become unavailable (e.g., in low-visibility or cluttered environments). In such cases, relying solely on exteroceptive sensors can lead to drift, particularly in the z-position and yaw, which are typically difficult to estimate accurately without external observations. By incorporating slip detection, our method can maintain accurate state estimation even when exteroceptive data is missing or unreliable, providing redundancy and robustness that previous approaches do not offer. This can be seen in Tabs. 4.2–4.4, where we show that adding slip detection improves the performance in terms of ATE and RPE, and in Fig. 4.9a,

where it is clear that adding slip detection helps reduce the drift in the z-position.

Furthermore, we demonstrated that our MUSE framework has real-time capabilities by showcasing an experiment where MUSE provides velocity and orientation feedback to a high-frequency locomotion controller. This is a significant advancement, as other state-of-the-art methods, such as VILENS (Wisth et al. [2]), only provide feedback to mapping policies that do not require high-frequency operation. Although we recognize that exteroceptive measurements are not strictly required for a closed-loop state estimator, and that they may introduce discontinuities if used for control feedback, we experimented that leg odometry measurement alone is not immune to abrupt changes, since it relies on discrete foot contact detection. In challenging environments, misdiagnosed contacts and slip events can cause leg-odometry velocity to be unreliable, leading to drift in the state estimation. Incorporating exteroceptive odometry, such as velocity estimates from the T265 tracking camera, helps the state estimator to correct or reduce the velocity errors. Furthermore, reliable pose feedback is essential for fully autonomous tasks, such as navigation and path following, where even small drift can accumulate over long traverses or in complex terrain. While high-frequency pose estimates may not always be strictly necessary, having access to them can help capture and react to rapid changes in the robot's motion. Consequently, by fusing both leg and exteroceptive odometry, MUSE delivers more robust pose estimation, balancing the respective limitations of each sensor modality while preserving real-time control performance. Online feedback to a locomotion policies is critical in dynamic environments and in scenarios where quick reaction times are needed, and we believe this is a key aspect of our contribution.

4.8.1 Limitations

Although MUSE has demonstrated strong capabilities, it has some limitations, particularly in the following areas:

- **High friction in the Aliengo's joints**: The contact estimation module occasionally experiences inaccuracies due to high friction in the Aliengo's joints, which affects its dynamics. This can lead to errors in the slip detection module, which depends on precise contact estimation to detect slippage, and also in the leg odometry measurements, which rely on accurate contact information. Improving the contact estimation module to better account for high friction in the joints is a potential direction for future work.
- **Slip detection module limitations**: The slip detection module does not always capture 100% of slippage events. In some cases, it may fail to detect slippage,

resulting in errors in the state estimation. Enhancing the slip detection module to capture a broader range of slippage events is another potential area for improvement in future work.

- **Violation of the assumption on the fixed location of the contact point**: To estimate the foot contact with the ground, it is typically assumed that the contact point lies at a fixed location at the center of the foot. However, this assumption does not fully reflect reality for many legged robots, where feet are often spherical in shape. In such cases, the actual contact point is located on the surface of the sphere rather than at its center. Depending on the sphere's radius, this discrepancy can introduce an error of up to a few centimeters (e.g., 2 cm for the HyQ robot). Additionally, during locomotion, the spherical foot may roll slightly, causing the contact point to shift dynamically. While the slip detection module helps discard unreliable measurements during slip events, these deviations in the contact point can still introduce errors in contact state estimation and inaccuracies in the computed GRFs. Therefore, developing methods to detect and compensate for these variations in the contact point can significantly improve robustness.

4.9 Conclusion

This chapter presented MUSE, a state estimator designed to improve accuracy and real-time performance in quadruped robot navigation. By integrating camera and LiDAR odometry with foot-slip detection, MUSE fuses data from multiple sources, including IMU and joint encoders, to provide reliable pose and motion estimates, even in complex environments.

Ablation studies conducted on the Aliengo robot, along with benchmarking against other state-of-the-art estimators using the FSC Dataset of ANYmal B300 platform, validate the robustness and adaptability of MUSE across different scenarios. The results demonstrate the estimator's capability to handle dynamic and challenging conditions effectively, ensuring reliable performance during locomotion and navigation.

Future work includes the improvements of the contact estimation module which occasionally experiences inaccuracies due to high friction in the Aliengo's joints, affecting its dynamics, as well as developing camera and LiDAR odometry modules for dynamic environments where moving objects, people or animals introduce additional challenges.

Chapter 5

Invariant State Estimation on Lie-Groups

5.1 Preface

This chapter describes two state estimation frameworks, built upon the Invariant Extended Kalman Filter (InEKF) and the Invariant Smoother (IS), whose design is based on Lie and Invariant Error Theory. Developed in collaboration with the Korean Advanced Institute of Science and Technology (KAIST), in Daejeon, South Korea, where I conducted research during my Ph.D. secondment in 2024, the frameworks target state estimation for quadruped robots equipped with a LiDAR, IMU, joint encoders, force/torque sensors, and a GPS receiver. The effectiveness of the proposed frameworks was demonstrated through real-world experiments on the Hound (Shin et al. [10]) and Hound2 robots, developed by KAIST. The results showed that the proposed frameworks outperformed traditional state estimation methods, and they were able to correct the drift of the z-position, unobservable with only proprioception.

The frameworks were developed by me, based on the original implementation of the IS and InEKF developed by Hartley et al. [36] and Yoon et al. [50], while data collection was carried out by my colleague from KAIST, Hajun Kim. The analysis of the results and the manuscript preparation were collaboratively conducted with Hajun, who is the equal-contribution first author of the forthcoming publication. Supervision was provided by João Carlos Virgolino Soares, Geoff Fink, Hae-Won Park, and Claudio Semini.

Ylenia Nisticò*, Hajun Kim*, João Carlos Virgolino Soares, Geoff Fink, Hae-Won Park, and Claudio Semini, "*Multi-Sensor Fusion for Quadruped Robot State Estimation using Invariant Filtering and Smoothing*", Under review at *IEEE Robotics and Automation Letter*. * Equal contribution.

5.2 Introduction

As mentioned in the previous chapters, several well-known frameworks have been developed in the field of multi-sensor state estimators for legged robots. The Pronto state estimator (Camurri et al. [6]) utilizes an EKF to fuse data from an IMU, leg kinematics, stereo vision, and LiDAR for pose corrections. Similarly, WALK-VIO (Lim et al. [3]) integrates IMU, camera, and joint encoder data to estimate the robot's state, adjusting leg kinematics dynamically based on body motion. The STEP state estimator presented by Kim et al. [4] relies on pre-integrated foot velocity factors and stereo camera data for pose estimation, eliminating the need for contact detection and non-slip conditions, addressing some limitations of earlier methods, including Pronto and WALK-VIO, which assume constant ground contact. However, STEP relies heavily on camera inputs, which can be unreliable in featureless environments or areas with reflections, affecting accuracy and robustness. In (Wisth et al. [2]), the VILENS state estimator combines IMUs, kinematics, LiDAR, and cameras using factor graphs for reliable estimation even when individual sensors fail. Cerberus (Yang et al. [121]), instead fuses data from stereo cameras, IMU, joint encoders, and contact sensors to form a visual-inertial-leg odometry estimator, where visual information is used to estimate the kinematic parameters. Factor graphs are used for non-linear optimization and precise state estimation. More recently, Leg-KILO was presented in (Ou et al. [93]). Using graph optimization, leg-KILO tightly integrates leg odometry, lidar odometry, and loop closure.

The aforementioned studies utilized linearized dynamics dependent on the current estimate via EKF or factor graphs, which can lead to inaccuracies in highly nonlinear systems (Barrau and Bonnabel [122]). In contrast, Lie group-based methods naturally accommodate nonlinearities by representing the state directly on a manifold, reducing linearization errors and improving estimation accuracy (Sola et al. [123]). Additionally, by leveraging the group-affine property, Lie group-based estimators such as the InEKF (Barrau and Bonnabel [124]) ensure that the error dynamics remain log-linear. This property enhances the filter's convergence properties and stability, leading to more reliable state estimation. Furthermore, invariant error representation allows the estimation process to

be independent of the choice of coordinates, making it more robust to variations in the robot's pose and the environment (Barfoot [125]).

Specifically for legged robots, Lie group-based state estimation can dynamically manage contact points, removing or adding them as needed based on the robot's interactions with the environment. This dynamic handling is crucial for legged robots, which frequently experience changes in contact points due to walking or running on varied terrains. For instance, Hartley et al. [36] presented an InEKF that incorporates contact points as part of the state, allowing the robot to adapt to changes in contact conditions and maintain accurate state estimates.

The system proposed by Lin et al. [126], introduces a learning-based contact estimator for legged robots that use a network to estimate contact events across various terrains. Experiments demonstrate that a contact-aided InEKF that uses these estimated contacts, generates accurate odometry trajectories.

To further enhance the robustness of state estimation when walking on difficult terrains, in (Teng et al. [127]), the authors propose a state estimator for legged robots in slippery environments using an InEKF. It fuses inertial and velocity measurements from a tracking camera with leg kinematic constraints and auto-calibrates camera pose. Furthermore, Gao et al. [128] presents an InEKF that estimates pose and velocity using sensors such as IMUs, joint encoders, and RGB-D cameras, while the robot is walking on a dynamic surface.

In (Santana et al. [49]) the authors developed a novel InEKF for legged robots using only proprioceptive sensors and robust cost functions in the measurement update. Tested on quadruped robots, their approach reduces pose drift by up to 40% over trajectories longer than 450 meters compared to a state-of-the-art method, improving performance in challenging terrains.

When constructing state estimators, a key decision is choosing between filtering and smoothing approaches. Filtering methods sequentially fuse measurements without considering the entire history of states, keeping the process efficient as all previous states are marginalized. In contrast, smoothing methods perform batch optimization by considering the complete state trajectory, selected keyframes of past states, or a sliding window of previous states (Absil et al. [129]). Invariant smoothers (IS), are estimation algorithms that utilize group-affine properties in a smoother framework. One of the first works was proposed by Chauchat et al. [130]. The algorithm, based on a maximum a posteriori estimator, minimizes the need for re-linearization by leveraging the system's group-affine properties for observation. Its effectiveness is demonstrated through localization on a wheeled robot, equipped with gyroscopes, velocity odometry,

and GNSS measurements. Also, Walsh et al. [131], tested an IS for attitude and heading reference system with gyroscope bias in simulation.

Specifically for legged robots, there is limited literature on implementing invariant smoothing frameworks for state estimation. One notable work is (Yoon et al. [50]), where the authors developed and tested an invariant smoother that, similar to the approach in (Hartley et al. [36]), incorporated foot position into the state matrix. This method utilizes IMU measurements and leg kinematics while assuming static foot contact.

5.3 Contributions

Some of the aforementioned studies (e.g. Pronto, WALK-VIO, STEP, VILENS, Leg-KILO) utilized linearized dynamics dependent on the current estimate via EKF or factor graphs, which can lead to inaccuracies in highly nonlinear systems (Barrau and Bonnabel [122]). In contrast, Lie group-based methods handle nonlinearities by representing the state on a manifold and error dynamics in a log-linear form, which leads to improving estimation accuracy and convergence stability (Barrau and Bonnabel [124], Chauchat et al. [130]). This invariant error representation enables the estimation to be performed independently of the choice of coordinates, providing more robust performance in the robot's pose and the environment (Barfoot [125]).

In this chapter, we develop two multi-sensor-fused state estimation frameworks for legged robots, extending InEKF (Hartley et al. [36]) and IS (Yoon et al. [50]) using the group-affine properties. These frameworks combine data from kinematics, IMU, LiDAR, and GPS to mitigate position drift inherent in proprioceptive-only methods. The main contributions of this work are as follows:

- We propose two frameworks that fuse kinematics, IMU, LiDAR odometry, and GPS, mitigating position drift. To the best of the authors' knowledge, this is the first work to incorporate LiDAR odometry and GPS in an InEKF for quadruped robots, and also the first work to incorporate exteroceptive measurements in an IS framework. Also, to manage LiDAR's low frequency of approximately 10 Hz, we obtain the LiDAR odometry in a parallel thread by using the ICP registration of [9], enabling the estimator thread to maintain fast computation times.

- Our algorithms were verified on the Hound and Hound2 robots [10] in indoor and outdoor environments, while also providing a comparison between the performance of the proposed methods. We also benchmark the obtained results against two state-of-the-art LiDAR-based odometry systems [9, 11].

5.4 Outline

The remainder of this chapter is organized as follows. Section 5.5 provides a brief overview of Lie theory, and state estimation on manifolds. Section 5.6 describes the models of the robots used in this work, and the definition of the state. Sections 5.7 and 5.8 presents the formulation of the InEKF and IS frameworks. Section 5.9 briefly described the slip rejection method used in both the state estimation frameworks. Section 5.10 presents the experimental results, and Section 5.11 discusses the results. Finally, Section 5.12 concludes the chapter.

5.5 Theoretical Background

This section provides a brief overview of Lie theory and state estimation on manifolds, to introduce the necessary mathematical background for understanding the proposed state estimation frameworks. We will first focus on Lie theory, which is essential for understanding the structure of the state space of the robot. Then, we will review some group-affine properties, which are crucial for developing state estimation algorithms that can handle the nonlinearities of the robot's dynamics. For more details on the material discussed in the following section, we refer the interested reader to (Hartley et al. [36], Yoon et al. [50], Barrau and Bonnabel [122], Sola et al. [123], Barrau and Bonnabel [124], Absil et al. [129], Chauchat et al. [130], Barrau [132]).

5.5.1 Lie Theory

Lie theory examines the intricate relationship between algebra and geometry. Named after the Norwegian mathematician Sophus Lie, this field primarily investigates Lie groups, Lie algebras, and their representations, which serve as a foundation for understanding continuous symmetries.

In this context, we consider a matrix Lie group denoted as $\mathcal{G}$ and its associated Lie algebra $\mathfrak{g}$. When $\mathcal{G}$ consists of $n \times n$ matrices, its corresponding Lie algebra $\mathfrak{g}$ also comprises $n \times n$ matrices. For computational purposes, it is convenient to define the map:

$$(\cdot)^{\wedge} : \mathbb{R}^{\dim \mathfrak{g}} \to \mathfrak{g} \tag{5.1}$$

which transforms elements of the tangent space of $\mathcal{G}$ at the identity into their respective matrix representations.

In robotics, much of the focus on state estimation involves determining the orientation of the robot's body. However, representing the robot's orientation as a 3D rotation matrix introduces challenges, as such matrices do not reside in vector spaces but on manifolds. Specifically, the rotation matrix belongs to a matrix Lie group known as the special orthogonal group, denoted $SO(3)$ (Dellaert et al. [133], Chirikjian [134, 135], Eade [136]). Similarly, incorporating the robot's position and velocity often requires the special Euclidean group, denoted $SE(3)$.

Thus, this section explores the properties of the matrix Lie groups $SO(3)$ and $SE_k(3)$, which is an extension of the $SE(3)$ group, offering insights into the manifold nature of these variables and their relevance to robotic systems.

Special Orthogonal Group $SO(3)$: In most robotics applications, the robot's state does not reside in a simple vector space but rather on a manifold. For example, orientation is typically represented by the Special Orthogonal Group $\mathbf{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} \mid \det(\mathbf{R}) = 1, \mathbf{R}^\top \mathbf{R} = \mathbb{I}_3\}$, where $\mathbb{I}_{\mathrm{nd}} \in \mathbb{R}^{\mathrm{nd}\times \mathrm{nd}}$ is the identity matrix.

Special Euclidean Group of k-Direct Isometries $SE_k(3)$: To simultaneously estimate position, velocity, and orientation, one can leverage the group of k-Direct isometries $\mathbf{SE}_k(3)$ (Barrau and Bonnabel [124]).

Each element of $\mathbf{SE}_k(3)$ is a $(3+k)$ by $(3+k)$ square matrices:

$$\mathbf{X} \triangleq \begin{bmatrix} \mathbf{R} & {}^1p & \dots & {}^Np \\ \mathbf{0}_{k,3} & & \mathbb{I}_k & \end{bmatrix}, \tag{5.2}$$

where $\mathbf{R} \in \mathbf{SO}(3)$ represents the rotation matrix, and ${}^ip \in \mathbb{R}^3$, for $i = 1, 2, \dots, k$, is a vector.

A state $\mathbf{X} \in \mathbf{SE}_k(3)$ can be mapped to a corresponding vector $\xi \in \mathbb{R}^{3+3k}$ via logarithmic and exponential operations. This mapping provides a way to move between matrix Lie group representations and their corresponding vector forms:

$$\mathbf{X} \in \mathbf{SE}_k(3) \to \mathrm{Log}(\mathbf{X}) \in \mathbb{R}^{3+3k}, \tag{5.3}$$

$$\xi \in \mathbb{R}^{3+3k} \to \mathrm{Exp}(\xi) \in \mathbf{SE}_k(3). \tag{5.4}$$

The exponential map in Equation (5.4) is given by $\mathrm{Exp}(\xi) = \exp(\xi^\wedge)$, and it is defined as follows:

$$\mathrm{Exp}(\xi) = \begin{bmatrix} \exp(\phi^\wedge) & \mathbf{J}_l(\phi)\,{}^1\xi & \cdots & \mathbf{J}_l(\phi)\,{}^k\xi \\ \mathbf{0}_{k,3} & & \mathbb{I}_k & \end{bmatrix}, \tag{5.5}$$

where $\mathbf{J}_l(\cdot)$ denotes the left Jacobian on the $\mathbf{SO}(3)$ manifold. The *hat* operator $(\cdot)^\wedge$, which is defined as the inverse mapping of the *vee* operator $(\cdot)^\vee$, facilitating operations

within the Lie algebra. It is defined as

$$\phi^{\wedge} = \begin{bmatrix} \phi_{\mathrm{x}} \\ \phi_{\mathrm{y}} \\ \phi_{\mathrm{z}} \end{bmatrix}^{\wedge} = \begin{bmatrix} 0 & -\phi_{\mathrm{z}} & \phi_{\mathrm{y}} \\ \phi_{\mathrm{z}} & 0 & \phi_{\mathrm{x}} \\ -\phi_{\mathrm{y}} & -\phi_{\mathrm{x}} & 0 \end{bmatrix}. \tag{5.6}$$

The adjoint matrix is defined as follows:

$$\mathrm{Ad}_{\mathbf{X}} = \begin{bmatrix} \mathbf{R} & \mathbf{0}_{3\times 3} & \cdots & \mathbf{0}_{3\times 3} \\ {}^{1}p^{\wedge}\mathbf{R} & \mathbf{R} & \cdots & \mathbf{0}_{3\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ {}^{k}p^{\wedge}\mathbf{R} & \mathbf{0}_{3\times 3} & \cdots & \mathbf{R} \end{bmatrix} \tag{5.7}$$

5.5.2 Group-Affine Properties

The group-affine properties are essential for defining system models on matrix Lie groups that lead to state-independent error dynamics, as shown in (Barrau and Bonnabel [124]) and (Chauchat et al. [130]). These properties arise from specific propagation and observation models designed on matrix Lie groups.

Definition 1 (Right Invariant Error). Let $\mathbf{X}_{\mathrm{t}}$ be the true state at time t and $\bar{\mathbf{X}}_{\mathrm{t}}$ be the estimate in the Lie Group $\mathbb{G}$. The right-invariant error between the true state $\mathbf{X}_t$ and its estimate $\bar{\mathbf{X}}_t$ is defined as:

$$\eta_{\mathrm{t}}^{\mathrm{r}} \triangleq \bar{\mathbf{X}}_{\mathrm{t}}^{-1}\mathbf{X}_{\mathrm{t}} \tag{5.8}$$

and the right log-invariant error is obtained by applying the logarithm to this quantity:

$$\xi_{\mathrm{t}}^{\mathrm{r}} \triangleq \mathrm{Log}(\eta_{\mathrm{t}}^{l}) \tag{5.9}$$

Theorem 1 (Autonomous Error Dynamics (Barrau and Bonnabel [124])): *A system is called* ***group affine*** *if its propagation function $f(\cdot)$ satisfies the following condition*:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{X}\mathrm{t} = f(\mathbf{X}_{\mathrm{t}}), \\ &f(\mathbf{X}\mathbf{Y}) = \mathbf{X}f(\mathbf{Y}) + f(\mathbf{X})\mathbf{Y} - \mathbf{X}f(\mathbb{I}_{\mathsf{dim}(\mathbf{Y})})\mathbf{Y}, \\ &\qquad \text{for} \quad \forall \mathrm{t} \geq 0 \quad \text{and} \quad \forall \mathbf{X}, \mathbf{Y} \in \mathbb{G}. \end{aligned} \tag{5.10}$$

Equation (5.10) ensures that the log-invariant error $\xi_{\mathrm{t}}^{\mathrm{r}}$ has a linear propagation model

with a constant matrix $\mathbf{G}$:

$$g(\eta_t^r) = (\mathbf{G}\xi_t^r)^\wedge + \mathcal{O}(\|\xi_t^r\|^2), \frac{d}{dt}\xi_t^r = \mathbf{G}\xi_t^r. \tag{5.11}$$

This property significantly simplifies the analysis and design of nonlinear estimators on manifolds.

Observation Model Either in filtering or smoothing, the error is updated by incorporating sensor measurements. If the measurements $\mathbf{Y_t}$ adhere to specific forms (Equation (5.12)), the linearized observation model and innovation become autonomous (Barrau and Bonnabel [124]). The observations take the form:

$$\mathbf{Y_t} = \mathbf{X_t^{-1}b} + \mathbf{V_t} \quad \text{(Right-Invariant Observation)} \tag{5.12}$$

Here $\mathbf{b}$ is a constant vector, and $\mathbf{V_t}$ is the observation noise.

Definition 2 (Adjoint Map): The adjoint map, fundamental to Lie group theory, captures the non-commutative structure of a Lie group. For a matrix Lie group $\mathcal{G}$ with Lie algebra $\mathfrak{g}$, the adjoint map $\mathrm{Ad} : \mathcal{G} \to \mathrm{GL}(\mathfrak{g})$ is defined as:

$$\mathrm{Ad}_{\xi^\wedge} = \mathbf{X}\xi^\wedge\mathbf{X}^{-1} \tag{5.13}$$

where $\mathbf{X} \in \mathcal{G}$ and $\xi \in \mathfrak{g}$. The matrix representation of this map is denoted as $\mathrm{Ad}_\mathbf{X} \in \mathbb{R}^{n\times n}$.

5.5.3 Invariant Filtering vs. Invariant Smoothing

The choice between filtering and smoothing methods is a crucial consideration when designing state estimation algorithms. The main differences between these approaches can be summarized as follows:

- **Objective**: The InEKF estimates the state at the **current time step**, based on measurements available up to that point. In contrast, the IS computes a MAP estimate of the entire trajectory by utilizing all measurements over the entire time horizon.
- **Use of measurements**: The InEKF incorporates only **current** measurements at each time step to update the current state estimate. On the other hand, the IS leverages also **past and future** measurements to refine past state estimates, leading to improved accuracy over the entire trajectory.

- **Computational complexity**: The InEKF is computationally more efficient than the IS, as it processes only the current state and measurements, making it suitable for real-time applications. The IS, however, requires **batch optimization**, where the state estimation problem is formulated as a single optimization problem over the entire trajectory. This involves simultaneously estimating all states by minimizing a cost function that incorporates all measurements and system dynamics. While this approach improves estimation accuracy, it is computationally intensive because it involves solving large-scale optimization problems. To enable real-time application, the IS can employ **fixed-lag smoothing**, which limits the optimization to a time window that incorporates only a subset of future measurements, balancing computational demand and estimation accuracy.

5.5.4 Summary of the Theoretical Background

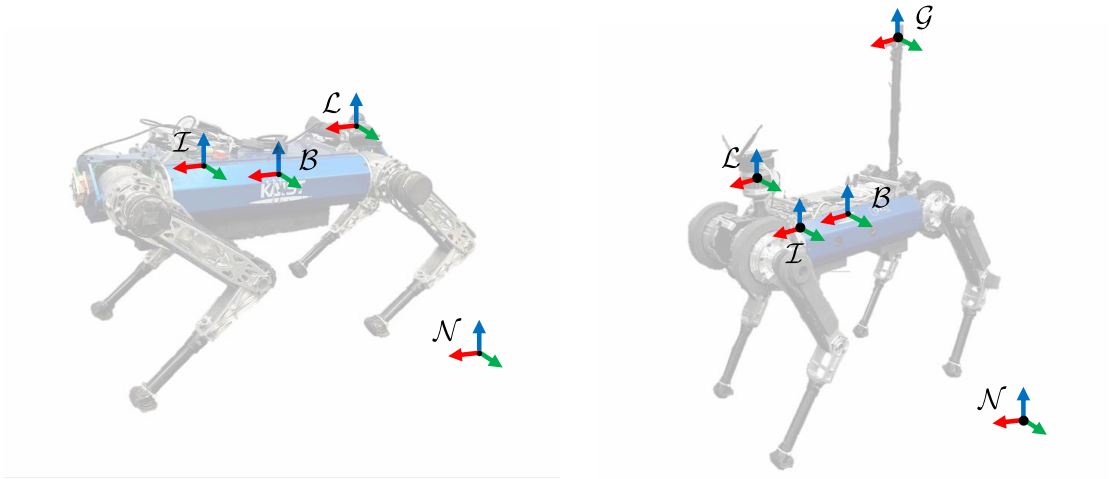
This section provided a brief overview of Lie theory and state estimation on manifolds, essential for understanding the proposed state estimation frameworks. We reviewed the concept of Lie groups, focusing on the special orthogonal group $SO(3)$ and on $SE_k(3)$ the special Euclidean group of k-Direct isometries, which are crucial for representing the orientation and position of robotic systems. We also discussed the group-affine properties that underpin the development of state estimation algorithms on matrix Lie groups, such as the InEKF and the IS.

With the core theory and key differences between filtering and smoothing methods outlined, we now present the formulations of the InEKF and IS frameworks for state estimation.

5.6 Robot Models and State Definitions

In this study, we employed the dynamic and kinematic models of the Hound (Shin et al. [10]) and Hound2 robots. We defined the following reference frames: the Navigation frame ($\mathcal{N}$), considered an inertial frame, the Body frame ($\mathcal{B}$), located at the geometric center of the robot's trunk, the IMU frame ($\mathcal{I}$), originating from the accelerometer within the IMU mounted on the trunk, the LiDAR frame ($\mathcal{L}$), centered on the sensor positioned atop the robot, and the GPS frame ($\mathcal{G}$), situated on the GPS antenna receiver on Hound2. The body frame's axes are oriented as follows: forward, left, and upward. The robots and their respective reference frames are depicted in Fig. 5.1.

Our objective is to estimate the robot's base orientation ($\mathbf{R_t}$), velocity ($\mathbf{v_t}$), position



(a) The 45 Kg Hound robot **(b) The 50 Kg Hound2 robot**

Figure 5.1: The quadruped robot platforms used to test the InEKF and IS. $\mathcal{N}$ is the navigation frame, $\mathcal{B}$ is the body frame, $\mathcal{I}$ is the IMU frame, $\mathcal{L}$ is the LiDAR frame, and $\mathcal{G}$ is the GPS frame.

$(\mathbf{p_t})$, and the position of each foot in contact with the ground $(\mathbf{d_t})$. Additionally, we incorporate biases from the IMU into the state definition, specifically the gyroscope bias $(\mathbf{b_t^\omega})$ and the accelerometer bias $(\mathbf{b_t^a})$. All state variables are defined in frame $\mathcal{N}$; for clarity and to reduce notation complexity, this frame is not explicitly specified in the formulas. Following the approach of Hartley et al. [36], we define the states in a Lie group framework, while retaining the IMU biases in a vector space. This decision stems from their associated equations being non-group-affine, as highlighted by Barrau and Bonnabel [124]. Consequently, this approach results in an "imperfect" implementation of both the InEKF and IS.

Specifically, for N contact points, the state matrices are defined as follows:

- $\mathbf{X} \in \mathbf{SE_{N+2}(3)}$, representing the Lie group state.
- $\mathbf{x} \in \mathbb{R}^6$, representing the vector space biases.

Both matrices are defined as:

$$\mathbf{X} \triangleq \begin{bmatrix} \mathbf{R_t} & \mathbf{v_t} & \mathbf{p_t} & \mathbf{d_{t_1}} & \cdots & \mathbf{d_{t_N}} \\ 0_{1,3} & 1 & 0 & 0 & \cdots & 0 \\ 0_{1,3} & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{1,3} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{x} \triangleq \begin{bmatrix} \mathbf{b}_t^\omega \\ \mathbf{b}_t^a \end{bmatrix} \tag{5.14}$$

To simplify the derivations and enhance readability, a single contact point, $\mathbf{d_t}$, is

assumed for all subsequent equations, as the measurement models for each contact point $\mathbf{d}_{\mathbf{t_i}}$ are identical.

The measurements considered in our analysis are as follows:

$$\tilde{\boldsymbol{\omega}}_{\mathbf{t}}, \tilde{\mathbf{a}}_{\mathbf{t}}, \tilde{\mathbf{q}}_{\mathbf{t}}, \dot{\tilde{\mathbf{q}}}_{\mathbf{t}}, \mathbf{p}_{\mathbf{c}\ell} \tag{5.15}$$

where $\tilde{\boldsymbol{\omega}}_{\mathbf{t}} \in \mathbb{R}^3$, $\tilde{\mathbf{a}}_{\mathbf{t}} \in \mathbb{R}^3$, $\tilde{\mathbf{q}}_{\mathbf{t}} \in \mathbb{R}^{3N}$, $\dot{\tilde{\mathbf{q}}}_{\mathbf{t}} \in \mathbb{R}^{3N}$, and $\mathbf{p}_{\mathbf{c}\ell}$ are the gyroscope, accelerometer, joint positions, joint velocities, and LiDAR point cloud for the N-legged robot, respectively.

5.6.1 Continuous-Time System Dynamics

The system dynamics follow the approach described in (Hartley et al. [36], Yoon et al. [50]), and the equations are reported here for clarity. The IMU measurements consist of the angular velocity ($\tilde{\boldsymbol{\omega}}_{\mathbf{t}}$) and linear acceleration ($\tilde{\mathbf{a}}_{\mathbf{t}}$). These measurements are assumed to be corrupted by slowly varying biases ($\mathbf{b}^{\boldsymbol{\omega}}_{\mathbf{t}}$ and $\mathbf{b}^{\mathbf{a}}_{\mathbf{t}}$) and zero-mean white Gaussian sensor noise ($\mathbf{w}^{\boldsymbol{\omega}}$ and $\mathbf{w}^{\mathbf{a}}$). This relationship is modeled as:

$$\tilde{\boldsymbol{\omega}}_{\mathbf{t}} = \boldsymbol{\omega}_{\mathbf{t}} + \mathbf{b}^{\boldsymbol{\omega}}_{\mathbf{t}} + \mathbf{w}^{\boldsymbol{\omega}} \quad \text{and} \quad \tilde{\mathbf{a}}_{\mathbf{t}} = \mathbf{a}_{\mathbf{t}} + \mathbf{b}^{\mathbf{a}}_{\mathbf{t}} + \mathbf{w}^{\mathbf{a}} \tag{5.16}$$

These biases are represented by a parameter vector that is estimated as part of the right-invariant InEKF state and IS state:

$$\boldsymbol{\theta}_{\mathbf{t}} \triangleq \begin{bmatrix} \mathbf{b}^{\boldsymbol{\omega}}_{\mathbf{t}} \\ \mathbf{b}^{\mathbf{a}}_{\mathbf{t}} \end{bmatrix} \in \mathbb{R}^6 \tag{5.17}$$

The augmented right-invariant error is then defined as:

$$\mathbf{e}^{\mathbf{r}}_{\mathbf{t}} \triangleq (\bar{\mathbf{X}}_{\mathbf{t}} \mathbf{X}^{-1}_{\mathbf{t}}, \bar{\boldsymbol{\theta}}_{\mathbf{t}} - \boldsymbol{\theta}_{\mathbf{t}}) \triangleq (\boldsymbol{\eta}^{\mathbf{r}}_{\mathbf{t}}, \boldsymbol{\zeta}_t) \tag{5.18}$$

Explicitly, the right-invariant error, $\boldsymbol{\eta}^{\mathbf{r}}_{\mathbf{t}}$, is given by:

$$\boldsymbol{\eta}^{\mathbf{r}}_{\mathbf{t}} = \begin{bmatrix} \bar{\mathbf{R}}_{\mathbf{t}} \mathbf{R}^{\top}_{\mathbf{t}} & \bar{\mathbf{v}}_{\mathbf{t}} - \bar{\mathbf{R}}_{\mathbf{t}} \mathbf{R}^{\top}_{\mathbf{t}} \mathbf{v}_{\mathbf{t}} & \bar{\mathbf{p}}_{\mathbf{t}} - \bar{\mathbf{R}}_{\mathbf{t}} \mathbf{R}^{\top}_{\mathbf{t}} \mathbf{p}_{\mathbf{t}} & \bar{\mathbf{d}}_{\mathbf{t}} - \bar{\mathbf{R}}_{\mathbf{t}} \mathbf{R}^{\top}_{\mathbf{t}} \mathbf{d}_{\mathbf{t}} \\ \mathbf{0}_{\mathbf{1,3}} & 1 & 0 & 0 \\ \mathbf{0}_{\mathbf{1,3}} & 0 & 1 & 0 \\ \mathbf{0}_{\mathbf{1,3}} & 0 & 0 & 1 \end{bmatrix} \tag{5.19}$$

The parameter vector error, $\boldsymbol{\zeta_t}$, is defined as:

$$\boldsymbol{\zeta_t} = \begin{bmatrix} \bar{\mathbf{b}}_\mathbf{t}^{\boldsymbol{\omega}} - \mathbf{b}_\mathbf{t}^{\boldsymbol{\omega}} \\ \bar{\mathbf{b}}_\mathbf{t}^{\mathbf{a}} - \mathbf{b}_\mathbf{t}^{\mathbf{a}} \end{bmatrix} \triangleq \begin{bmatrix} \boldsymbol{\zeta}_\mathbf{t}^{\boldsymbol{\omega}} \\ \boldsymbol{\zeta}_\mathbf{t}^{\mathbf{a}} \end{bmatrix} \tag{5.20}$$

This formulation provides the basis for estimating both the IMU biases and the state using the InEKF and IS.

The system dynamics integrate IMU measurements of acceleration and angular velocity, incorporating the assumption of stable foot contact, which implies zero foot velocity during ground contact. The dynamics are expressed as follows:

$$\frac{d}{dt}\mathbf{R_t} = \mathbf{R_t}(\tilde{\boldsymbol{\omega}}_\mathbf{t} - \mathbf{b}_\mathbf{t}^{\boldsymbol{\omega}} - \mathbf{w}^{\boldsymbol{\omega}})^\wedge, \tag{5.21a}$$

$$\frac{d}{dt}\mathbf{v_t} = \mathbf{R_t}(\tilde{\mathbf{a}}_\mathbf{t} - \mathbf{b}_\mathbf{t}^{\mathbf{a}} - \mathbf{w}^{\mathbf{a}}) + \mathbf{g}, \tag{5.21b}$$

$$\frac{d}{dt}\mathbf{p_t} = \mathbf{v_t}, \tag{5.21c}$$

$$\frac{d}{dt}\mathbf{d_t} = \mathbf{R}_t\mathbf{h}_\mathbf{R}(\tilde{\boldsymbol{\alpha}}_\mathbf{t})(-\mathbf{w}_\mathbf{t}^{\mathbf{v}}), \tag{5.21d}$$

$$\frac{d}{dt}\mathbf{b}_\mathbf{t}^{\boldsymbol{\omega}} = \mathbf{w}^{\mathbf{b}^{\boldsymbol{\omega}}}, \tag{5.21e}$$

$$\frac{d}{dt}\mathbf{b}_\mathbf{t}^{\mathbf{a}} = \mathbf{w}^{\mathbf{b}^{\mathbf{a}}} \tag{5.21f}$$

Here, $\mathbf{g} \in \mathbb{R}^3$ is the gravity vector, and $\mathbf{w}^{\boldsymbol{\omega}}$, $\mathbf{w}^{\mathbf{a}}$, $\mathbf{w}^{\mathbf{c}}$, $\mathbf{w}^{\mathbf{b}^{\boldsymbol{\omega}}}$, and $\mathbf{w}^{\mathbf{b}^{\mathbf{a}}}$ represent zero-mean Gaussian noise associated with each process.

The deterministic system dynamics depend on both the inputs $\mathbf{u}_t$ and the parameters $\boldsymbol{\theta_t}$, and can be written as:

$$f_{u_t}(\bar{\mathbf{X}}_\mathbf{t}, \bar{\boldsymbol{\theta}}_\mathbf{t}) = \begin{bmatrix} \bar{\mathbf{R}}_\mathbf{t}(\bar{\boldsymbol{\omega}}_\mathbf{t})_\times & \bar{\mathbf{R}}_\mathbf{t}\bar{\mathbf{a}}_\mathbf{t} + \mathbf{g} & \bar{\mathbf{v}}_\mathbf{t} & \mathbf{0_{3,1}} \\ \mathbf{0_{1,3}} & 0 & 0 & 0 \\ \mathbf{0_{1,3}} & 0 & 0 & 0 \\ \mathbf{0_{1,3}} & 0 & 0 & 0 \end{bmatrix} \tag{5.22}$$

Here $\bar{\boldsymbol{\omega}} \triangleq \tilde{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\mathbf{t}^{\boldsymbol{\omega}}$ $\bar{\mathbf{a}} \triangleq \tilde{\mathbf{a}} - \bar{\mathbf{b}}_\mathbf{t}^{\mathbf{a}}$. These are the bias-corrected angular velocity and acceleration measurements (inputs). To derive the linearized error dynamics, the augmented right-invariant error from Equation 5.20 is differentiated with respect to time:

$$\frac{d}{dt}\mathbf{e}_\mathbf{t}^{\mathbf{r}} = \left(\frac{d}{dt}\boldsymbol{\eta}_\mathbf{t}^{\mathbf{r}}, \begin{bmatrix} \mathbf{w}_\mathbf{t}^{\boldsymbol{\omega}} \\ \mathbf{w}_\mathbf{t}^{\mathbf{a}} \end{bmatrix}\right) \tag{5.23}$$

Applying the chain rule and a first-order approximation, $\boldsymbol{\eta}_\mathbf{t}^{\mathbf{r}} = \mathrm{Exp}(\boldsymbol{\xi}_t) \approx \mathbf{I_d} + \boldsymbol{\xi_t}^\wedge$,

the individual terms of the invariant error dynamics are:

$$\frac{d}{dt}\left(\bar{\mathbf{R}}_\mathbf{t}\mathbf{R}_\mathbf{t}{}^\top\right) \approx \left(\bar{\mathbf{R}}_\mathbf{t}(\mathbf{w}_\mathbf{t}^{\boldsymbol{\omega}} - \boldsymbol{\zeta}_\mathbf{t}^{\boldsymbol{\omega}})\right)_\times \tag{5.24a}$$

$$\frac{d}{dt}\left(\bar{\mathbf{v}}_\mathbf{t} - \bar{\mathbf{R}}_\mathbf{t}\mathbf{R}_\mathbf{t}^\top\mathbf{v}_\mathbf{t}\right) \approx (\mathbf{g})_\times\boldsymbol{\xi}_\mathbf{t}^\mathbf{R} + (\bar{\mathbf{v}}_\mathbf{t})_\times\bar{\mathbf{R}}_\mathbf{t}(\mathbf{w}_\mathbf{t}^{\boldsymbol{\omega}} - \zeta_\mathbf{t}^{\boldsymbol{\omega}}) + \bar{\mathbf{R}}_\mathbf{t}(\mathbf{w}_\mathbf{t}^\mathbf{a} - \boldsymbol{\zeta}_\mathbf{t}^\mathbf{a}) \tag{5.24b}$$

$$\frac{d}{dt}\left(\bar{\mathbf{p}}_\mathbf{t} - \bar{\mathbf{R}}_\mathbf{t}\mathbf{R}_\mathbf{t}^\top\mathbf{p}_\mathbf{t}\right) \approx \boldsymbol{\xi}_\mathbf{t}^\mathbf{v} + (\bar{\mathbf{p}}_\mathbf{t})_\times\bar{\mathbf{R}}_\mathbf{t}(\mathbf{w}_\mathbf{t}^{\boldsymbol{\omega}} - \zeta_\mathbf{t}^{\boldsymbol{\omega}}) \tag{5.24c}$$

$$\frac{d}{dt}\left(\bar{\mathbf{d}}_\mathbf{t} - \bar{\mathbf{R}}_\mathbf{t}\mathbf{R}_\mathbf{t}^\top\mathbf{d}_\mathbf{t}\right) \approx (\bar{\mathbf{d}}_\mathbf{t})_\times\bar{\mathbf{R}}_\mathbf{t}(\mathbf{w}_\mathbf{t}^{\boldsymbol{\omega}} - \zeta_\mathbf{t}^{\boldsymbol{\omega}}) + \bar{\mathbf{R}}_\mathbf{t}\mathbf{h}_\mathbf{R}(\tilde{\boldsymbol{\alpha}}_\mathbf{t})\mathbf{w}_\mathbf{t}^\mathbf{v} \tag{5.24d}$$

The augmented invariant error dynamics depend only on the estimated trajectory through the noise and bias error, $\boldsymbol{\zeta}_t$. When there are no bias errors, the dynamics are independent of the estimated trajectory.

A linear system can now be constructed from Equation 5.24:

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{\xi}_\mathbf{t}\\ \boldsymbol{\zeta}_\mathbf{t}\end{bmatrix} = \mathbf{A}_\mathbf{t}\begin{bmatrix}\boldsymbol{\xi}_\mathbf{t}\\ \boldsymbol{\zeta}_\mathbf{t}\end{bmatrix} + \begin{bmatrix}\mathrm{Ad}_{\bar{\mathbf{X}}_\mathbf{t}} & \mathbf{0}_{\mathbf{12,6}}\\ \mathbf{0}_{\mathbf{6,12}} & \mathbf{I}_\mathbf{6}\end{bmatrix}\mathbf{w}_\mathbf{t} \tag{5.25}$$

where $\mathbf{A}_\mathbf{t}$ and $\mathrm{Ad}_{\bar{\mathbf{X}}_\mathbf{t}}$ are defined as:

$$\mathbf{A}_\mathbf{t} = \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\bar{\mathbf{R}}_\mathbf{t} & \mathbf{0}\\ (\mathbf{g})_\times & \mathbf{0} & \mathbf{0} & \mathbf{0} & -(\bar{\mathbf{v}}_\mathbf{t})_\times\bar{\mathbf{R}}_\mathbf{t} & -\bar{\mathbf{R}}_\mathbf{t}\\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & -(\bar{\mathbf{p}}_\mathbf{t})_\times\bar{\mathbf{R}}_\mathbf{t} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -(\bar{\mathbf{d}}_\mathbf{t})_\times\bar{\mathbf{R}}_\mathbf{t} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix} \quad \mathrm{Ad}_{\bar{\mathbf{X}}_\mathbf{t}} = \begin{bmatrix}\mathbf{R}_\mathbf{t} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ (\mathbf{v}_\mathbf{t})_\times\mathbf{R}_\mathbf{t} & \mathbf{R}_\mathbf{t} & \mathbf{0} & \mathbf{0}\\ (\mathbf{p}_\mathbf{t})_\times\mathbf{R}_\mathbf{t} & \mathbf{0} & \mathbf{R}_\mathbf{t} & \mathbf{0}\\ (\mathbf{p}_{\mathbf{tc}_\mathbf{t}})_\times\mathbf{R}_\mathbf{t} & \mathbf{0} & \mathbf{0} & \mathbf{R}_\mathbf{t}\end{bmatrix} \tag{5.26}$$

and $\mathbf{w}_\mathbf{t}$ is the augmented noise vector:

$$\mathbf{w}_\mathbf{t} = \mathrm{vec}(\mathbf{w}_\mathbf{t}{}^{\boldsymbol{\omega}}, \mathbf{w}_\mathbf{t}^\mathbf{a}, \mathbf{0}_{\mathbf{3,1}}, \mathbf{h}_\mathbf{R}(\tilde{\boldsymbol{\alpha}}_\mathbf{t})\mathbf{w}_\mathbf{t}^\mathbf{v}, \mathbf{w}_\mathbf{t}{}^{\boldsymbol{\omega}}, \mathbf{w}_\mathbf{t}^\mathbf{a}) \tag{5.27}$$

5.7 Invariant Extended Kalman Filter formulation

Building on the robot models, state definitions, and continuous-time system dynamics, we now outline the formulation of the InEKF for state estimation on matrix Lie groups.

5.7.1 Prediction Step

As in (Hartley et al. [36]), the state estimate $\bar{\mathbf{X}}_\mathbf{t}$ is propagated using the deterministic system dynamics, which include the IMU biases. The covariance matrix $\mathbf{P}_\mathbf{t}$ is updated according to the Riccati equation (Maybeck [137]):

$$\frac{d}{dt}(\bar{\mathbf{X}}_\mathbf{t}, \bar{\boldsymbol{\theta}}_\mathbf{t}) = f_{u_t}(\bar{\mathbf{X}}_\mathbf{t}, \bar{\boldsymbol{\theta}}_\mathbf{t}, \mathbf{0}_{\mathbf{6},\mathbf{1}}) \tag{5.28}$$

$$\frac{d}{dt}\mathbf{P}_\mathbf{t} = \mathbf{A}_\mathbf{t}\mathbf{P}_\mathbf{t} + \mathbf{P}_\mathbf{t}\mathbf{A}_\mathbf{t}^\top + \bar{\mathbf{Q}}_\mathbf{t} \tag{5.29}$$

where $\mathbf{A}_\mathbf{t}$ is the Jacobian of the system dynamics, defined in Equation 5.26. $\bar{\mathbf{Q}}_\mathbf{t}$ is the noise covariance matrix. The noise covariance matrix, $\bar{\mathbf{Q}}_\mathbf{t}$, is given by:

$$\bar{\mathbf{Q}}_\mathbf{t} = \begin{bmatrix} \mathrm{Ad}_{\bar{\mathbf{X}}_\mathbf{t}} & \mathbf{0}_{\mathbf{12},\mathbf{6}} \\ \mathbf{0}_{\mathbf{6},\mathbf{12}} & \mathbf{I}_\mathbf{6} \end{bmatrix} \mathrm{Cov}(\mathbf{w}_\mathbf{t}) \begin{bmatrix} \mathrm{Ad}_{\bar{\mathbf{X}}_\mathbf{t}} & \mathbf{0}_{\mathbf{12},\mathbf{6}} \\ \mathbf{0}_{\mathbf{6},\mathbf{12}} & \mathbf{I}_\mathbf{6} \end{bmatrix}^\top \tag{5.30}$$

where $\mathrm{Ad}_{\bar{\mathbf{X}}_\mathbf{t}}$ is the adjoint matrix representation, defined in Equation (5.26).

5.7.2 Right-Invariant Measurement Model

We define the right-invariant measurement model by considering the forward kinematics of each leg, LiDAR measurements, and the GPS position.

5.7.2.1 Forward Kinematics Measurement Model

Let $\tilde{\mathbf{q}}_\mathbf{t} \in \mathbb{R}^{3N}$ represent the joint positions between the robot's body and its contact points. These encoder measurements are assumed to be corrupted by additive white Gaussian noise, $\mathbf{w}_\mathbf{t}^\mathbf{q}$:

$$\tilde{\mathbf{q}}_\mathbf{t} = \mathbf{q}_\mathbf{t} + \mathbf{w}_\mathbf{t}^\mathbf{q} \tag{5.31}$$

Using forward kinematics, the relative position of the contact point with respect to the body is measured. The forward kinematics position measurement $\mathbf{p}_\mathbf{k}(\tilde{\mathbf{q}}_\mathbf{t})$ is expressed as:

$$\mathbf{p}_\mathbf{k}(\tilde{\mathbf{q}}_\mathbf{t}) = \mathbf{R}_\mathbf{t}{}^\top(\mathbf{p}_{\mathbf{t_c}} - \mathbf{p}_\mathbf{t}) + \mathbf{J}_\mathbf{p}(\tilde{\mathbf{q}}_\mathbf{t})\mathbf{w}_\mathbf{t}^\mathbf{q} \tag{5.32}$$

where $\mathbf{J}_\mathbf{p}$ denotes the analytical Jacobian of the forward kinematics function. In matrix form, this measurement has the right-invariant observation form $\mathbf{Y} = \mathbf{X}^{-\mathbf{1}} + \mathbf{b}$ (Equation (5.12)):

$$\underbrace{\begin{bmatrix}\mathbf{p_k}(\tilde{\mathbf{q}}_\mathbf{t})\\0\\1\\-1\end{bmatrix}}_{\mathbf{Y_{kin}}} = \underbrace{\begin{bmatrix}\mathbf{R}_t^\top & -\mathbf{R}_\mathbf{t}^\top\mathbf{v_t} & -\mathbf{R}_\mathbf{t}^\top\mathbf{p_t} & -\mathbf{R}_\mathbf{t}^\top\mathbf{d_t}\\ \mathbf{0}_{1,3} & 1 & 0 & 0\\ \mathbf{0}_{1,3} & 0 & 1 & 0\\ \mathbf{0}_{1,3} & 0 & 0 & 1\end{bmatrix}}_{\mathbf{X}^{-1}}\underbrace{\begin{bmatrix}\mathbf{0}_{3,1}\\0\\1\\-1\end{bmatrix}}_{\mathbf{b_{kin}}} + \underbrace{\begin{bmatrix}\mathbf{J_p}(\tilde{\mathbf{q}}_\mathbf{t})\mathbf{w_t^q}\\0\\0\\0\end{bmatrix}}_{\mathbf{V_{kin}}} \tag{5.33}$$

Here $\mathbf{Y_{kin}}$ is the kinematic observation vector, $\mathbf{b_{kin}}$ is a constant vector, while $\mathbf{V_{kin}}$ is the Gaussian noise vector for the observation model.

5.7.2.2 LiDAR Measurement Model

Based on (Vizzo et al. [9]), we obtain the LiDAR position from point cloud data in a parallel thread, enabling the main estimator thread to maintain fast computation times. Transforming the LiDAR position into the body frame $\mathcal{B}$ ($\mathbf{p_{lid}}$), the measurement in the right-invariant observation form is similar to Equation (5.33):

$$\underbrace{\begin{bmatrix}\mathbf{p_{lid}}\\0\\1\\0\end{bmatrix}}_{\mathbf{Y_{lid}}} = \underbrace{\begin{bmatrix}\mathbf{R_t}^\top & -\mathbf{R}_\mathbf{t}^\top\mathbf{v_t} & -\mathbf{R}_\mathbf{t}^\top\mathbf{p_t} & -\mathbf{R}_\mathbf{t}^\top\mathbf{d_t}\\ \mathbf{0}_{1,3} & 1 & 0 & 0\\ \mathbf{0}_{1,3} & 0 & 1 & 0\\ \mathbf{0}_{1,3} & 0 & 0 & 1\end{bmatrix}}_{\mathbf{X}^{-1}}\underbrace{\begin{bmatrix}\mathbf{0}_{3,1}\\0\\1\\0\end{bmatrix}}_{\mathbf{b_{lid}}} + \underbrace{\begin{bmatrix}\mathbf{J}_{\mathbf{p}_\ell}\mathbf{w}_t^\ell\\0\\0\\0\end{bmatrix}}_{\mathbf{V_{lid}}} \tag{5.34}$$

Here $\mathbf{Y_{lid}}$ is the LiDAR observation vector, $\mathbf{b_{lid}}$ is a constant vector, while $\mathbf{V_{lid}}$ is the Gaussian noise vector for the observation model.

5.7.2.3 GPS Measurement Model

Transforming the GPS position into the body frame $\mathcal{B}$ ($\mathbf{p_{gps}}$), the GPS measurement is expressed in right-invariant observation (Equation (5.12)) form as:

$$\underbrace{\begin{bmatrix}\mathbf{p_{gps}}\\0\\1\\0\end{bmatrix}}_{\mathbf{Y_{gps}}} = \underbrace{\begin{bmatrix}\mathbf{R_t}^\top & -\mathbf{R}_\mathbf{t}^\top\mathbf{v_t} & -\mathbf{R}_\mathbf{t}^\top\mathbf{p_t} & -\mathbf{R}_\mathbf{t}^\top\mathbf{d_t}\\ \mathbf{0}_{1,3} & 1 & 0 & 0\\ \mathbf{0}_{1,3} & 0 & 1 & 0\\ \mathbf{0}_{1,3} & 0 & 0 & 1\end{bmatrix}}_{\mathbf{X}^{-1}}\underbrace{\begin{bmatrix}\mathbf{0}_{3,1}\\0\\1\\0\end{bmatrix}}_{\mathbf{b_{gps}}} + \underbrace{\begin{bmatrix}\mathbf{J}_{\mathbf{p_{gps}}}\mathbf{w_t^{gps}}\\0\\0\\0\end{bmatrix}}_{\mathbf{V_{gps}}} \tag{5.35}$$

Here $\mathbf{Y_{gps}}$ is the GPS observation vector, $\mathbf{b_{gps}}$ is a constant vector, while $\mathbf{V_{gps}}$ is the Gaussian noise vector for the observation model.

5.7.3 Augmented Right-Invariant Observation and Innovation

Starting from Equation (5.33), Equation (5.34), and Equation (5.35), we define an augmented observation vector $\mathbf{Y}_t = [\mathbf{Y_{kin}}^\top \quad \mathbf{Y_{lid}}^\top \quad \mathbf{Y_{gps}}^\top)^\top$, and a corresponding augmented constant vector: $\mathbf{b} = [\mathbf{b_{kin}}^\top \quad \mathbf{b_{lid}}^\top \quad \mathbf{b_{gps}}^\top)^\top$. The right-invariant observation can be expressed as:

$$\underbrace{\begin{bmatrix}\mathbf{Y_{kin}}\\ \mathbf{Y_{lid}}\\ \mathbf{Y_{gps}}\end{bmatrix}}_{\mathbf{Y}_t} = \underbrace{\begin{bmatrix}\mathbf{X}^{-1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{X}^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{X}^{-1}\end{bmatrix}}_{\mathbf{X_{aug}}^{-1}} \underbrace{\begin{bmatrix}\mathbf{b_{kin}}\\ \mathbf{b_{lid}}\\ \mathbf{b_{gps}}\end{bmatrix}}_{\mathbf{b}} + \underbrace{\begin{bmatrix}\mathbf{V_{kin}}\\ \mathbf{V_{lid}}\\ \mathbf{V_{gps}}\end{bmatrix}}_{\mathbf{V}} \tag{5.36}$$

where $\mathbf{X_{aug}}^{-1}$ is the augmented inverse state matrix.

The right-invariant innovation $\mathbf{z}$, defined as $\mathbf{z} = [\mathbf{z_{kin}}^\top \quad \mathbf{z_{lid}}^\top \quad \mathbf{z_{gps}}^\top]^\top$, is then:

$$\underbrace{\begin{bmatrix}\mathbf{z_{kin}}\\ \mathbf{z_{lid}}\\ \mathbf{z_{gps}}\end{bmatrix}}_{\mathbf{z}} = \underbrace{\begin{bmatrix}\mathbf{X} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{X} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{X}\end{bmatrix}}_{\mathbf{X_{aug}}} \underbrace{\begin{bmatrix}\mathbf{Y_{kin}}\\ \mathbf{Y_{lid}}\\ \mathbf{Y_{gps}}\end{bmatrix}}_{\mathbf{Y}} - \underbrace{\begin{bmatrix}\mathbf{b_{kin}}\\ \mathbf{b_{lid}}\\ \mathbf{b_{gps}}\end{bmatrix}}_{\mathbf{b}} \tag{5.37}$$

where $\mathbf{X_{aug}}$ is the augmented state matrix, where the state matrix $\mathbf{X}$ is on the diagonal.

5.7.3.1 Update Equations

The linear update equations follow the approach in (Hartley et al. [36]) and are given by:

$$\left(\bar{\mathbf{X}}_\mathbf{t}^+, \boldsymbol{\theta}_\mathbf{t}^+\right) = \left(\exp\left(\mathbf{K}_\mathbf{t}^{\boldsymbol{\xi}}\boldsymbol{\Pi}\bar{\mathbf{X}}_\mathbf{t}\mathbf{Y}_\mathbf{t}\right)\bar{\mathbf{X}}_\mathbf{t}, \bar{\theta}_\mathbf{t} + \mathbf{K}_\mathbf{t}^{\boldsymbol{\zeta}}\boldsymbol{\Pi}\bar{\mathbf{X}}_\mathbf{t}\mathbf{Y}_\mathbf{t}\right) \tag{5.38a}$$

$$\mathbf{P}_\mathbf{t}^+ = \left(\mathbf{I} - \mathbf{K}_\mathbf{t}\mathbf{H}_\mathbf{t}\right)\mathbf{P}_\mathbf{t}\left(\mathbf{I} - \mathbf{K}_\mathbf{t}\mathbf{H}_\mathbf{t}\right)^\top + \mathbf{K}_\mathbf{t}\bar{\mathbf{N}}_\mathbf{t}{\mathbf{K}_\mathbf{t}}^\top \tag{5.38b}$$

where $\mathbf{K}_\mathbf{t}^{\boldsymbol{\xi}}$ and $\mathbf{K}_\mathbf{t}^{\boldsymbol{\zeta}}$ are the Kalman gain matrices for the state and bias estimates, computed as:

$$\mathbf{S}_\mathbf{t} = \mathbf{H}_\mathbf{t}\mathbf{P}_\mathbf{t}{\mathbf{H}_\mathbf{t}}^\top + \bar{\mathbf{N}}_\mathbf{t} \quad , \quad \mathbf{K}_\mathbf{t} = \begin{bmatrix}\mathbf{K}_\mathbf{t}^{\boldsymbol{\xi}}\\ \mathbf{K}_\mathbf{t}^{\boldsymbol{\zeta}}\end{bmatrix} = \mathbf{P}_\mathbf{t}{\mathbf{H}_\mathbf{t}}^\top\mathbf{S}_\mathbf{t}^{-1} \tag{5.39}$$

with the following output and noise matrices:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{\mathbf{kin}}^\top & \mathbf{H}_{\mathbf{lid}}^\top & \mathbf{H}_{\mathbf{gps}}^\top \end{bmatrix}^\top \quad , \quad \mathbf{N} = \begin{bmatrix} \mathbf{N}_{\mathbf{kin}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{N}_{\mathbf{lid}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{N}_{\mathbf{gps}} \end{bmatrix} \tag{5.40a}$$

The individual terms are defined as:

$$\mathbf{H}_{\mathbf{kin}} = \begin{bmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & -\mathbf{I} & \mathbf{I}_{3,1} \end{bmatrix} \quad , \quad \mathbf{N}_{\mathbf{kin}} = \mathbf{R}_\mathbf{t} \mathbf{J}_\mathbf{p}(\tilde{\mathbf{q}}_\mathbf{t}) \boldsymbol{\Sigma}(\mathbf{w}_\mathbf{t}^\mathbf{q}) \mathbf{J}_\mathbf{p}^\top(\tilde{\mathbf{q}}_\mathbf{t}) {\mathbf{R}_\mathbf{t}}^\top \tag{5.41a}$$

$$\mathbf{H}_{\mathbf{lid}} = \begin{bmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & -\mathbf{I} & \mathbf{0}_{3,1} \end{bmatrix} \quad , \quad \mathbf{N}_{\mathbf{lid}} = \mathbf{R}_\mathbf{t} \mathbf{J}_{\mathbf{p_{lid}}} \boldsymbol{\Sigma}(\mathbf{w}_\mathbf{t}^\mathbf{lid}) \mathbf{J}_{\mathbf{p_{lid}}}^\top {\mathbf{R}_\mathbf{t}}^\top \tag{5.41b}$$

$$\mathbf{H}_{\mathbf{gps}} = \begin{bmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & -\mathbf{I} & \mathbf{0}_{3,1} \end{bmatrix} \quad , \quad \mathbf{N}_{\mathbf{gps}} = \mathbf{R}_\mathbf{t} \mathbf{J}_{\mathbf{p_{gps}}} \boldsymbol{\Sigma}(\mathbf{w}_\mathbf{t}^\mathbf{gps}) \mathbf{J}_{\mathbf{p_{gps}}}^\top {\mathbf{R}_\mathbf{t}}^\top \tag{5.41c}$$

Here $\boldsymbol{\Sigma}(\mathbf{w}_\mathbf{t}^\mathbf{q})$, $\boldsymbol{\Sigma}(\mathbf{w}_\mathbf{t}^\mathbf{lid})$, and $\boldsymbol{\Sigma}(\mathbf{w}_\mathbf{t}^\mathbf{gps})$ are the covariance matrices of the joint position, LiDAR position and GPS position noise, respectively.

5.7.4 Addition and Removal of Contact Points

As described in (Hartley et al. [36]), the addition and removal of contact points are essential for handling the discrete events where contact points are created or broken as the robot navigates its environment. We report the method here for completeness. These operations require dynamically updating the observer's state representation.

5.7.4.1 Removing Contact Points

To remove a contact point, the corresponding state variable is marginalized by eliminating its associated row and column from the matrix Lie group. The corresponding elements in the covariance matrix are also removed. This is achieved through a linear transformation. For instance, transitioning from one contact point to zero contact points results in the reduced covariance:

$$\begin{bmatrix} \boldsymbol{\xi}_\mathbf{t}^\mathbf{R} \\ \boldsymbol{\xi}_\mathbf{t}^\mathbf{v} \\ \boldsymbol{\xi}_\mathbf{t}^\mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_\mathbf{t}^\mathbf{R} \\ \boldsymbol{\xi}_\mathbf{t}^\mathbf{v} \\ \boldsymbol{\xi}_\mathbf{t}^\mathbf{p} \\ \boldsymbol{\xi}_\mathbf{t}^\mathbf{d} \end{bmatrix} \tag{5.42}$$

which implies:

$$\boldsymbol{\xi}_\mathbf{t}^{\text{new}} \triangleq \mathbf{M} \boldsymbol{\xi}_\mathbf{t} \quad \text{and} \quad {\mathbf{P}_\mathbf{t}}^{\text{new}} = \mathbf{M} \mathbf{P}_\mathbf{t} \mathbf{M}^\top \tag{5.43}$$

where $\mathbf{M}$ is the transformation matrix.

5.7.4.2 Adding Contact Points

When the robot establishes a new contact, the state and covariance matrices must be augmented. Special care is required to initialize the mean and covariance for the new contact point. For example, when transitioning from zero to one contact point, the initial mean is computed using the forward kinematics relation:

$$\bar{\mathbf{d}}_\mathbf{t} = \bar{\mathbf{p}}_\mathbf{t} + \bar{\mathbf{R}}_\mathbf{t}\mathbf{h}_\mathbf{p}(\tilde{\boldsymbol{\alpha}}_\mathbf{t}) \tag{5.44}$$

where $\mathbf{h}_\mathbf{p}(\tilde{\boldsymbol{\alpha}}_t)$ represents the forward kinematics transformation.

To compute the new covariance, the right-invariant error is analyzed:

$$\begin{aligned}
\boldsymbol{\eta}^\mathbf{d}_\mathbf{t} &= \bar{\mathbf{d}}_\mathbf{t} - \bar{\mathbf{R}}_\mathbf{t}\mathbf{R}_\mathbf{t}^\top\mathbf{d}_\mathbf{t} \\
&= \bar{\mathbf{p}}_\mathbf{t} + \bar{\mathbf{R}}_\mathbf{t}\mathbf{h}_\mathbf{p}(\tilde{\boldsymbol{\alpha}}_\mathbf{t}) + \bar{\mathbf{R}}_\mathbf{t}{\mathbf{R}_\mathbf{t}}^\top\mathbf{d}_\mathbf{t} \\
&= \bar{\mathbf{p}}_\mathbf{t} + \bar{\mathbf{R}}_\mathbf{t}\mathbf{h}_\mathbf{p}(\tilde{\boldsymbol{\alpha}}_\mathbf{t}) + \bar{\mathbf{R}}_\mathbf{t}{\mathbf{R}_\mathbf{t}}^\top\bar{\mathbf{p}}_\mathbf{t} + \bar{\mathbf{R}}_\mathbf{t}{\mathbf{R}_\mathbf{t}}^\top\mathbf{h}_\mathbf{p}(\tilde{\boldsymbol{\alpha}}_\mathbf{t} - {\mathbf{w}_\mathbf{t}}^{\boldsymbol{\alpha}}) \\
&\approx \boldsymbol{\eta}^\mathbf{p}_\mathbf{t} + \bar{\mathbf{R}}_\mathbf{t}\mathbf{J}_\mathbf{p}(\tilde{\boldsymbol{\alpha}}_\mathbf{t}){\mathbf{w}_\mathbf{t}}^{\boldsymbol{\alpha}}
\end{aligned} \tag{5.45}$$

where $\mathbf{J}_\mathbf{p}(\tilde{\boldsymbol{\alpha}}_\mathbf{t})$ is the Jacobian of the forward kinematics. This leads to an approximate relation for the contact error:

$$\boldsymbol{\xi}^\mathbf{d}_\mathbf{t} \approx \boldsymbol{\xi}^\mathbf{p}_\mathbf{t} + \bar{\mathbf{R}}_\mathbf{t}\mathbf{J}_\mathbf{p}(\tilde{\boldsymbol{\alpha}}_\mathbf{t}){\mathbf{w}_\mathbf{t}}^{\boldsymbol{\alpha}} \tag{5.46}$$

The augmented covariance is computed using a linear map:

$$\begin{bmatrix} \boldsymbol{\xi}^\mathbf{R}_\mathbf{t} \\ \boldsymbol{\xi}^\mathbf{v}_\mathbf{t} \\ \boldsymbol{\xi}^\mathbf{p}_\mathbf{t} \\ \boldsymbol{\xi}^\mathbf{d}_\mathbf{t} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}^\mathbf{R}_\mathbf{t} \\ \boldsymbol{\xi}^\mathbf{v}_\mathbf{t} \\ \boldsymbol{\xi}^\mathbf{p}_\mathbf{t} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \bar{\mathbf{R}}_\mathbf{t}\mathbf{J}_\mathbf{p}(\tilde{\boldsymbol{\alpha}}_\mathbf{t}) \end{bmatrix} {\mathbf{w}_\mathbf{t}}^{\boldsymbol{\alpha}} \tag{5.47}$$

This implies:

$$\boldsymbol{\xi}^{\text{new}}_\mathbf{t} \triangleq \mathbf{F}_\mathbf{t}\boldsymbol{\xi}_\mathbf{t} + \mathbf{G}_\mathbf{t}{\mathbf{w}_\mathbf{t}}^{\boldsymbol{\alpha}} \tag{5.48a}$$

$${\mathbf{P}_\mathbf{t}}^{\text{new}} = \mathbf{F}_\mathbf{t}\mathbf{P}_\mathbf{t}{\mathbf{F}_\mathbf{t}}^\top + \mathbf{G}_\mathbf{t}\text{Cov}({\mathbf{w}_\mathbf{t}}^{\boldsymbol{\alpha}}){\mathbf{G}_\mathbf{t}}^\top \tag{5.48b}$$

where: $\mathbf{F}_\mathbf{t}$ is the augmentation matrix, while $\mathbf{G}_\mathbf{t}$ is the noise matrix, which depends on the error variable choice.

Remark: The above method for handling the addition and removal of contact points is reported here for completeness and follows the approach outlined in (Hartley et al.

[36]). The augmentation matrix $\mathbf{F_t}$ and noise matrix $\mathbf{G_t}$ depend on the system dynamics and must be carefully designed.

5.7.5 Summary of the Invariant Extended Kalman Filter

In this section, we have presented the formulation of a Right Invariant Extended Kalman Filter (R-InEKF) for legged robots. This filter is classified as "imperfect" because it estimates IMU biases from the gyroscope and accelerometer, which cannot satisfy group affine properties (Barrau [132]).

The filter builds upon the work of (Hartley et al. [36]) and is designed to handle intermittent contacts, which are created and broken as the robot navigates through its environment. Additionally, we incorporate global position measurements from LiDAR-odometry and GPS into its measurement model of the filter, enabling robust and accurate state estimation even in challenging conditions.

5.8 Invariant Smoother formulation

Unlike filtering approaches, which estimate the current state of the system, smoothing methods aim to recover the maximum a posteriori (MAP) estimate of the entire trajectory, given a set of measurements $\mathbf{Z}$. The problem is formulated as:

$$\begin{aligned} \mathbf{X}^*_{0:n}, \mathbf{x}^*_{0:n} &= \underset{\mathbf{X}_{0:n}, \mathbf{x}_{0:n}}{\arg\max} \; p(\mathbf{X}_{0:n}, \mathbf{x}_{0:n} \mid \mathbf{Z}_{0:n}) \\ &= \underset{\mathbf{X}_{0:n}, \mathbf{x}_{0:n}}{\arg\max} \; p(\mathbf{X}_{0:n}, \mathbf{x}_{0:n}) p(\mathbf{Z}_{0:n} \mid \mathbf{X}_{0:n}, \mathbf{x}_{0:n}). \end{aligned} \tag{5.49}$$

Based on the state definition and sensor measurements, following (Yoon et al. [50]), the MAP problem is decomposed into four components: *Prior*, *Propagation*, *Observation*, and *Loop closure* distributions. This leads to the following formulation:

$$\begin{aligned} \mathbf{X}^*_{0:n}, \mathbf{x}^*_{0:n} = \underset{\mathbf{X}_{0:n}, \mathbf{x}_{0:n}}{\arg\max} \; & \underbrace{p(\mathbf{X}_0, \mathbf{x}_0)}_{\text{Prior}} \underbrace{\prod_{i=0}^{n-1} p(\mathbf{X}_{i+1}, \mathbf{x}_{i+1} \mid \mathbf{X}_i, \mathbf{x}_i, \mathbf{Z}_i)}_{\text{Propagation}} \\ & \underbrace{\prod_{i=0}^{n} p(\mathbf{Z}_i \mid \mathbf{X}_i, \mathbf{x}_i)}_{\text{Observation}} \underbrace{\prod_{a,b \in \mathcal{L}} p(\mathbf{Z}_{a:b} \mid \mathbf{X}_a, \mathbf{x}_a, \mathbf{X}_b, \mathbf{x}_b)}_{\text{Loop}} \end{aligned} \tag{5.50}$$

where $\mathbf{Z}_i$ is the sensor observation at timestep i, and $\mathcal{L}$ is the set of long-term observations, with a and b representing the start and end timesteps of each loop closure constraint. The *Prior* distribution represents the initial state distribution, the *Prop-*

agation is the state transition distribution, the *Observation* is the sensor observation distribution, and the *Loop* is a long-term observation distribution.

The MAP problem can be rewritten as a nonlinear least squares problem:

$$\begin{aligned}\mathbf{X}^*_{0:n}, \mathbf{x}^*_{0:n} = \arg\min_{\mathbf{X}_{0:n}, \mathbf{x}_{0:n}} & \underbrace{\|\mathbf{r}_{\mathbf{Pri}}\|^2_{\Sigma_{\text{Pri}}}}_{\text{Prior}} + \underbrace{\sum_{i=0}^{n-1} \|\mathbf{r}_{\mathbf{Prop_i}}\|^2_{\Sigma_{\text{Prop}_i}}}_{\text{Propagation}} \\ & + \underbrace{\sum_{i=0}^{n} \|\mathbf{r}_{\mathbf{Obs_i}}\|^2_{\Sigma_{\text{Obs}_i}}}_{\text{Observation}} + \underbrace{\sum_{a,b \in \mathcal{L}} \|\mathbf{r}_{\mathbf{Loop_{a,b}}}\|^2_{\Sigma_{\text{Loop}_{a,b}}}}_{\text{Loop closure}}\end{aligned} \tag{5.51}$$

where $\mathbf{r}$ and $\mathbf{\Sigma}$ denote the residual functions and covariance matrices for each distribution.

To solve the nonlinear problem, a perturbation $\mathbf{e}_{0:n}$ is introduced at the current operating point, leading to the following optimization:

$$\begin{aligned}\mathbf{e}^*_{0:n} = \arg\min_{\mathbf{e}_{0:n}} & \|\bar{\mathbf{r}}_{\mathbf{Pri}} - \mathbf{J}_{\mathbf{Pri}}\mathbf{e}_{\mathbf{0:n}}\|^2_{\Sigma_{\text{Pri}}} + \sum_{i=0}^{n-1} \|\bar{\mathbf{r}}_{\mathbf{Prop_i}} - \mathbf{J}_{\mathbf{Prop_i}}\mathbf{e}_{\mathbf{0:n}}\|^2_{\Sigma_{\text{Prop}_i}} \\ & + \sum_{i=0}^{n} \|\bar{\mathbf{r}}_{\mathbf{Obs_i}} - \mathbf{J}_{\mathbf{Obs_i}}\mathbf{e}_{\mathbf{0:n}}\|^2_{\Sigma_{\text{Obs}_i}} + \sum_{a,b \in \mathcal{L}} \|\bar{\mathbf{r}}_{\mathbf{Loop_{a,b}}} - \mathbf{J}_{\mathbf{Loop_{a,b}}}\mathbf{e}_{\mathbf{0:n}}\|^2_{\Sigma_{\text{Loop}_{a,b}}}\end{aligned} \tag{5.52}$$

where $\bar{\mathbf{r}}$ are the residuals, and $\mathbf{J}$ are the Jacobians of the residual functions. At each iteration, the state is updated as:

$$\mathbf{X}^*_i \Leftarrow \text{Exp}(\xi^*_i)\bar{\mathbf{X}}_i \quad \text{and} \quad \mathbf{x}^*_i \leftarrow \bar{\mathbf{x}}_i + \zeta^*_i \tag{5.53}$$

where ξ^* is the perturbation of the state on the manifold, ζ^* is the perturbation of the bias, and $\bar{\mathbf{X}}_i$ is the current estimate of the manifold variable at iteration i.

5.8.1 Derivation of the Cost Functions

In this section, we derive the cost functions for the Prior, Propagation, Observation, and Loop Closure terms in the smoothing framework, based on the formulation in (Yoon et al. [50]).

5.8.1.1 Prior Cost Function

The prior cost captures the initial belief about the system state before incorporating sensor observations. This belief accounts for Gaussian noise in both the manifold and

vector components of the state. The state and its perturbations are represented by $\mathbf{X_0}$, $\mathbf{w_{Pri,M}}$, and $\mathbf{w_{Pri,v}}$, where:

$$\mathbf{w_{Pri}} = \begin{bmatrix} \mathbf{w_{Pri,M}} \\ \mathbf{w_{Pri,v}} \end{bmatrix} \tag{5.54}$$

The initial state is perturbed by noise and is described as follows:

$$\underbrace{\begin{bmatrix} \mathbf{I_5} & \mathbf{0_{5,1}} \end{bmatrix}}_{\mathbf{M_{Pri}}} \mathbf{X_0} \begin{bmatrix} \mathbf{I_5} & \mathbf{0_{5,1}} \end{bmatrix}^\top = \begin{bmatrix} \mathbf{R_0} & \mathbf{v_0} & \mathbf{p_0} \\ \mathbf{0_{1,3}} & 1 & 0 \\ \mathbf{0_{1,3}} & 0 & 1 \end{bmatrix} = \mathrm{Exp}(\mathbf{w_{Pri,M}})\mathbf{X_{Pri}} \tag{5.55a}$$

$$\mathbf{x_0} = \mathbf{x_{Pri}} + \mathbf{w_{Pri,v}} \tag{5.55b}$$

where $\mathbf{X_{Pri}}$ is the prior state belief, $\mathbf{x_{Pri}}$ is the prior belief of the bias, while $\mathbf{M_{Pri}}$ is an auxiliary block operator matrix to map $\mathbf{X_0} \in \mathbb{R}^{6\times6}$ to $\mathbb{R}^{5\times5}$. The matrix $\mathbf{M_{Pri}}$ excludes $\mathbf{d_0}$ from the prior cost, as it is initialized through the forward kinematics relation in the observation factor at the first timestep.

Using the exponential map, the prior belief about the manifold can be written as:

$$\mathbf{M_{Pri}}(\mathrm{Exp}(\boldsymbol{\xi_0})\bar{\mathbf{X}}_\mathbf{0})\mathbf{M_{Pri}}^\top = \mathrm{Exp}(\mathbf{w_{Pri,M}})\mathbf{X_{Pri}} \tag{5.56}$$

After applying $\mathbf{M_{Pri}}$ twice, we derive the linear relation:

$$\mathrm{Exp}(\mathbf{M_0}\boldsymbol{\xi_0})(\mathbf{M_{Pri}}, \bar{\mathbf{X}}_0, \mathbf{M_{Pri}}^\top) = \mathrm{Exp}(\mathbf{w_{Pri,M}}), \mathbf{X_{Pri}} \tag{5.57}$$

where $\mathbf{M_0} = [\mathbb{I}_{\nrightarrow} \quad \mathbf{0_{9,3}}]$. Rearranging this equation and introducing the distance between the prior belief and the current operating point, $\boldsymbol{\xi}_{\mathrm{Pri}}$, we obtain:

$$\mathrm{Exp}(\mathbf{M_0}\boldsymbol{\xi_0})\mathrm{Exp}(-\boldsymbol{\xi}_{\mathrm{Pri}}) = \mathrm{Exp}(\mathbf{w_{Pri,M}}) \tag{5.58}$$

Finally, applying the logarithmic map and the Baker-Campbell-Hausdorff (BCH) formula as explained in (Yoon et al. [50]) , the final expression for the prior cost on the manifold is:

$$\mathbf{w_{Pri,M}} = \mathbf{M_0}\boldsymbol{\xi_0} - \boldsymbol{\xi}_{\mathrm{Pri}} \tag{5.59}$$

Similarly for the bias, the prior belief is represented as:

$$\mathbf{w_{Pri,v}} = \boldsymbol{\zeta_0} - \boldsymbol{\zeta_{Pri}} \tag{5.60}$$

where $\boldsymbol{\zeta}_{\mathbf{Pri}} = \mathbf{x}_{\mathbf{Pri}} - \bar{\mathbf{x}}_{\mathbf{0}}$.

Finally, the following key components are derived: the residual vector, Jacobian matrix, and covariance matrix for the prior cost, which captures the deviation between the prior and current states, the sensitivity of the residual to state changes, and the associated uncertainty, respectively. Finally, the residual function $\mathbf{r}_{\mathbf{Pri}}$, the Jacobian $\mathbf{J}_{\mathbf{Pri}}$, and the corresponding covariance matrix $\boldsymbol{\Sigma}_{\text{Pri}}$ for the Prior cost are:

$$\mathbf{r}_{\mathbf{Pri}} = -\left[\text{Log}(\mathbf{X}_{\mathbf{Pri}}(\mathbf{M}_{\mathbf{Pri}}\bar{\mathbf{X}}_0{\mathbf{M}_{\mathbf{Pri}}}^\top)^{-1}), (\mathbf{x}_{\mathbf{Pri}} - \bar{\mathbf{x}}_0)^\top\right]^\top \tag{5.61a}$$

$$\mathbf{J}_{\mathbf{Pri}} = \begin{bmatrix} \mathbf{M_0} & \mathbf{0_{9,6}} \\ \mathbf{0_{6,9}} & \mathbf{I_6} \end{bmatrix} \tag{5.61b}$$

$$\boldsymbol{\Sigma}_{\mathbf{Pri}} = \text{Cov}(\mathbf{w}_{\mathbf{Pri}}) \tag{5.61c}$$

5.8.2 Propagation Cost Function

The "Propagation" cost models the evolution of the system state from the previous timestep to the current timestep, as dictated by the system dynamics. This cost is derived from the continuous-time system dynamics equations presented in Equation (5.21).

The noise-free propagation functions for the manifold state variable and vector space state variable are denoted as $f_M(\cdot)$ and $f_v(\cdot)$, respectively. The subscript M refers to the dynamics of the $SE_3(3)$ manifold, while v refers to the dynamics in $\mathbb{R}^6$. The explicit definitions are:

$$f_M(\mathbf{X_t}) = \begin{bmatrix} \mathbf{R_t}(\tilde{\boldsymbol{\omega}}_\mathbf{t} - \mathbf{b_t}^{\boldsymbol{\omega}})^\wedge & \mathbf{R_t}(\tilde{\mathbf{a}}_\mathbf{t} - \mathbf{b_t^a}) + \mathbf{g} & \mathbf{v_t} & \mathbf{0_{3\times1}} \\ \mathbf{0}_{3\times3} & \mathbb{I}_3 & & \end{bmatrix} \tag{5.62a}$$

$$f_v(\mathbf{x}_t) = \begin{bmatrix} \mathbf{0_{3\times1}} \\ \mathbf{0_{3\times1}} \end{bmatrix} \tag{5.62b}$$

The continuous-time log-linear error propagation equation derived by Hartley et al. [36] and presented in Equation (5.25), and Section 5.7 is discretized using a forward Euler method. The discrete propagation functions are:

$$f_M^d(\mathbf{X}_i) = \begin{bmatrix} \mathbf{R}_i\text{Exp}((\tilde{\boldsymbol{\omega}}_i - \mathbf{b}_i^{\boldsymbol{\omega}})\Delta t) & \mathbf{v_i^d} & \mathbf{p_i^d} & \mathbf{d}_i \\ \mathbf{0_{3\times3}} & \mathbf{I_{3\times3}} & & \end{bmatrix} \tag{5.63a}$$

$$f_v^d(\mathbf{x}_i) = \begin{bmatrix} \mathbf{b}_i^{\boldsymbol{\omega}} \\ \mathbf{b}_i^{\boldsymbol{a}} \end{bmatrix} \tag{5.63b}$$

where

$$\mathbf{v}_i^d = \mathbf{v}_i + \mathbf{R}_i(\tilde{\mathbf{a}}_i - \mathbf{b}_i^i)\Delta t + \mathbf{g}\Delta t \quad , \quad \mathbf{p}_i^d = \mathbf{p}_i + \mathbf{v}_i\Delta t + \frac{1}{2}\mathbf{R}_i(\tilde{\mathbf{a}}_i - \mathbf{b}_i^a)(\Delta t)^2 + \frac{1}{2}\mathbf{g}(\Delta t)^2 \tag{5.64}$$

The perturbation terms for the manifold and vector space states are defined as:

$$\boldsymbol{\xi}_{i+1}^f = \mathrm{Log}(\mathbf{X}_{i+1} f_M^d(\bar{\mathbf{X}}_i)) \quad \text{and} \quad \boldsymbol{\zeta}_{i+1}^f = \mathbf{x}_{i+1} - f_v^d(\bar{\mathbf{x}}_i) \tag{5.65}$$

Using forward Euler discretization, the perturbed propagation equation is:

$$\begin{bmatrix} \boldsymbol{\xi}_{i+1}^f \\ \boldsymbol{\zeta}_{i+1}^f \end{bmatrix} = (\mathbf{I_{18}} + \mathbf{A}_i\Delta t) \begin{bmatrix} \boldsymbol{\xi}_i \\ \boldsymbol{\zeta}_i \end{bmatrix} \begin{bmatrix} +\mathrm{Ad}_{\bar{\mathbf{X}}_i} & \mathbf{0_{12,6}} \\ \mathbf{0_{6,12}} & \mathbb{I}_{\not 4} \end{bmatrix} \Delta t \ \mathbf{w}_{\mathrm{Prop}}^{\boldsymbol{d}} \tag{5.66}$$

where $\mathbf{A}_i$ and $\mathrm{Ad}_{\bar{\mathbf{X}}_i}$ are defined in Equation (5.26), Δt is the discretizing time interval, while $\mathbf{w}_{\mathrm{Prop}}^{\boldsymbol{d}} = [(\mathbf{w}^{\boldsymbol{\omega}})^\top, (\mathbf{w}^{\boldsymbol{a}})^\top, (\mathbf{w}^{\boldsymbol{a}}\Delta t)^\top, (\mathbf{w}^{\boldsymbol{c}})^\top, (\mathbf{w}^{\boldsymbol{b^\omega}})^\top, (\mathbf{w}^{\boldsymbol{b^a}})^\top]^\top$.

The left-hand side of Equation (5.66) is further expanded to explicitly include the perturbation variables $[\boldsymbol{\xi}_{i+1}^\top, \boldsymbol{\zeta}_{i+1}^\top]$. Neglecting higher-order terms of $\boldsymbol{\Lambda}_M$ in the Jacobian, the residuals can be approximated as:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\xi}_{i+1}^f \\ \boldsymbol{\zeta}_{i+1}^f \end{bmatrix} &= \begin{bmatrix} \mathrm{Log}(\mathbf{X}_{i+1} f_M^d(\bar{\mathbf{X}}_i)) \\ \mathbf{x}_{i+1} - f_v^d(\bar{\mathbf{x}}_i) \end{bmatrix} = \begin{bmatrix} \mathrm{Log}(\mathbf{X}_{i+1}\bar{\mathbf{X}}_{i+1}^{-1}\bar{\mathbf{X}}_{i+1} f_M^d(\bar{\mathbf{X}}_i)^{-1}) \\ \mathbf{x}_{i+1} - \bar{\mathbf{x}}_{i+1} + \bar{\mathbf{x}}_{i+1} - f_v^d(\bar{\mathbf{x}}_s) \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{Log}(\mathrm{Exp}(\boldsymbol{\xi}_{i+1})\mathrm{Exp}(-\boldsymbol{\Lambda}_M)) \\ \boldsymbol{\zeta}_{i+1} - \boldsymbol{\Lambda}_v \end{bmatrix} \approx \begin{bmatrix} \boldsymbol{\xi}_{i+1} \\ \boldsymbol{\zeta}_{i+1} \end{bmatrix} - \boldsymbol{\Lambda} \end{aligned} \tag{5.67}$$

where $\boldsymbol{\Lambda} = [\boldsymbol{\Lambda}_M^\top, \boldsymbol{\Lambda}_v^\top]^\top$ represents the perturbation terms of the state and bias variables.

The residual function $\mathbf{r}_{\mathrm{Prop_i}}$, the Jacobian matrices $\mathbf{J}_{\mathrm{Prop}_{i+1}}$ and $\mathbf{J}_{\mathrm{Prop_i}}$, and the corresponding covariance matrix $\boldsymbol{\Sigma}_{\mathrm{Prop_i}}$ for the Propagation cost are:

$$\mathbf{r}_{\mathrm{Prop_i}} = -\begin{bmatrix} \mathrm{Log}(f_M^d(\bar{\mathbf{X}}_i)\bar{\mathbf{X}}_{i+1}^{-1}) \\ f_v^d(\bar{\mathbf{x}}_i) - \bar{\mathbf{x}}_{i+1} \end{bmatrix} \quad , \quad \mathbf{J}_{\mathrm{Prop_i}} = -(\mathbf{I_{18}} + \mathbf{A}_i\ \Delta t) \quad , \quad \mathbf{J}_{\mathrm{Prop}_{i+1}} = \mathbb{I}_{\not 4 \not 4} \tag{5.68a}$$

$$\boldsymbol{\Sigma}_{\mathrm{Prop_i}} = \mathbf{A}_{\mathrm{Prop_i}}\mathrm{Cov}(\mathbf{w}_{\mathrm{Prop}}^{\boldsymbol{d}})\mathbf{A}_{\mathrm{Prop_i}}{}^\top \quad \text{where} \quad \mathbf{A}_{\mathrm{Prop_i}} = \begin{bmatrix} \mathrm{Ad}_{\bar{\mathbf{X}}_i} & \mathbf{0_{12,6}} \\ \mathbf{0_{6,12}} & \mathbf{I_6} \end{bmatrix} \Delta t \tag{5.68b}$$

5.8.3 Observation Cost Function

The second main contribution of this chapter, together with the formulation of the augmented right-invariant measurement model for the InEKF (Section 5.7.3), is the formulation of an invariant smoother, where the *observation* model incorporates not only the foot positions derived from leg kinematics but also global position data from LiDAR and GPS. The *observation* leverages these global positions to correct the robot's state by comparing its position with known global reference points.

To derive the observation cost function for kinematics, we first express the right-invariant observation in matrix form, as shown in Equation (5.33). From this, we compute the residual function, which is then used to correct the robot's position. The residual function $\mathbf{r_{kin}}$, along with the corresponding Jacobian $\mathbf{J_{kin}}$ and covariance matrix $\mathbf{\Sigma_{kin}}$ are defined as follows:

$$\mathbf{r_{kin}} = \mathbf{X}\mathbf{Y_{kin}} - \mathbf{b_{kin}} \tag{5.69}$$

$$\mathbf{J_{kin}} = [\mathbf{b_{kin}}^{\odot} \ \ \mathbf{0}_{6,6}] \tag{5.70}$$

$$\mathbf{\Sigma_{kin}} = \mathbf{X}\mathbf{\Sigma}(\mathbf{w_t^q})\mathbf{X}^{\top} \tag{5.71}$$

where the definition of the $\odot$ operator is:

$$\boldsymbol{\xi}^{\odot} = \begin{bmatrix} \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & \cdots & \mathbf{0}_{3,3} \\ \xi_1^{\wedge} & \mathbf{0}_{3,3} & \cdots & \mathbf{0}_{3,3} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_k^{\wedge} & \mathbf{0}_{3,3} & \cdots & \mathbf{0}_{3,3} \end{bmatrix} \tag{5.72}$$

Next, we express the *LiDAR measurement model* in a right-invariant form, similar to Equation (5.34), using $\mathbf{p_{lid}}$, which represents the LiDAR position in the body frame $\mathcal{B}$. From this, we compute the residual function for the LiDAR observation, which is subsequently used to correct the robot's position. The residual $\mathbf{r_{lid}}$, the Jacobian $\mathbf{J_{lid}}$, and the covariance matrix $\mathbf{\Sigma_{lid}}$ are defined as follows:

$$\mathbf{r_{lid}} = \mathbf{X}\mathbf{Y_{lid}} - \mathbf{b_{lid}} \tag{5.73a}$$

$$\mathbf{J_{lid}} = [\mathbf{b_{lid}}^{\odot} \ \ \mathbf{0}_{6,6}] \tag{5.73b}$$

$$\mathbf{\Sigma_{lid}} = \mathbf{X}\mathbf{\Sigma_{lid}}(\mathbf{w_t^{lid}})\mathbf{X}^{\top} \tag{5.73c}$$

In this case, the global orientation provided by the LiDAR odometry is excluded from the observation model because it cannot be expressed in a right-invariant form that respects the smoother structure on $SE_k(3)$. Specifically, it is not possible to relate the rotation

data from LiDAR odometry in the required form $\mathbf{Y} = \mathbf{X}^{-1}\mathbf{b}$ of Equation (5.12), which is essential for maintaining the mathematical consistency of the right-invariant formulation. Incorporating the LiDAR orientation into the observation model would probably necessitate additional propagation factors to account for LiDAR measurements. However, this approach should be carefully investigated because LiDAR orientation data represents external measurements, which do not naturally fit into the smoother's internal structure on $SE_k(3)$. Introducing such propagation factors could disrupt the cleaner separation between the dynamics and observation models, leading to increased computational complexity and potential inaccuracies in the smoother formulation. As a result, only the global position measurements from LiDAR odometry are included in our formulation of the observation model, ensuring that the right-invariant structure of the smoother remains consistent and computationally efficient.

Finally, the GPS position residual is obtained from Equation (5.35), following the same approach as for the kinematic observation and LiDAR observation. The residual $\mathbf{r}_{\mathbf{gps}}$, Jacobian $\mathbf{J}_{\mathbf{gps}}$, and covariance matrix $\mathbf{\Sigma}_{\mathbf{gps}}$ are given by:

$$\mathbf{r}_{\mathbf{gps}} = \mathbf{X}\mathbf{Y}_{\mathbf{gps}} - \mathbf{b}_{\mathbf{gps}} \tag{5.74}$$

$$\mathbf{J}_{\mathbf{gps}} = [\mathbf{b}_{\mathbf{gps}}{}^{\odot} \ \mathbf{0}_{6,6}] \tag{5.75}$$

$$\mathbf{\Sigma}_{\mathbf{gps}} = \mathbf{X}\mathbf{\Sigma}_{\mathbf{gps}}(\mathbf{w}_{\mathbf{t}}^{\mathbf{gps}})\mathbf{X}^{\top} \tag{5.76}$$

5.8.4 Contact Loop Closure Cost Function

One of the advantages of the smoother framework is its ability to access states within the history window. This enables the formulation of measurement models that relate states across distant timesteps, which is not possible in filtering frameworks that only handle adjacent states. Leveraging this capability, the IS uses the Contact Loop (CL) method previously proposed by Yoon et al. [50], which enforces that foot positions remain consistent over multiple timesteps when no slip is detected. This is expressed as:

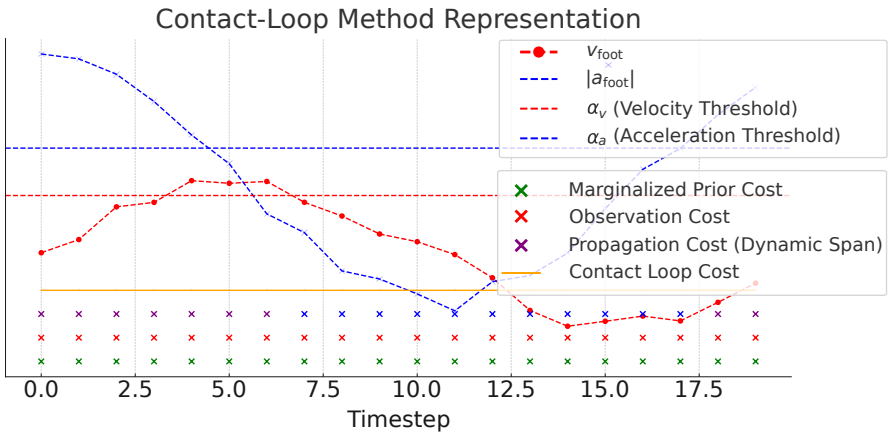


Figure 5.2: Illustration of the contact-loop method: foot velocity v_{foot} and foot acceleration v_{foot} are plotted against time steps. Then thresholds for both velocity α_v and acceleration $\alpha : a$ to identify static and dynamic (i.e. slippage) contact spans. Then factors for different costs are represented: Marginalized prior cost (green markers), Observation cost (red markers), Propagation cost (blue for static spans, purple for dynamic spans), then Contact loop cost (orange connecting line for long-term constraints). Picture adapted from (Yoon et al. [50]).

$$\mathbf{r}_{\mathbf{Loop_{a,b}}} = \underbrace{[\mathbb{I}_3 \quad \mathbf{0}_{3,3}]}_{\mathbf{M_{Loop}}}(\bar{\mathbf{X}}_a - \bar{\mathbf{X}}_b) \underbrace{\begin{bmatrix} \mathbf{0}_{3,1} \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{s} = \bar{\mathbf{d}}_a - \bar{\mathbf{d}}_b \tag{5.77a}$$

$$\mathbf{J}_{\mathbf{Loop_a}} = \mathbf{M}_{\mathbf{Loop}}([\mathbf{s}^{\odot} \quad \mathbf{0}_{6,6}]) \tag{5.77b}$$

$$\mathbf{J}_{\mathbf{Loop_b}} = -\mathbf{M}_{\mathbf{Loop}}([\mathbf{s}^{\odot} \quad \mathbf{0}_{6,6}]) \tag{5.77c}$$

$$\mathbf{\Sigma}_{\mathbf{Loop_{a,b}}} = \mathbf{\Sigma}_{\mathbf{cl}} \Delta t \tag{5.77d}$$

where $\mathbf{\Sigma_{cl}}$ is the covariance matrix, representing the confidence in the CL. Subscripts a and b refer to the start and end timesteps of the long-term observation. The residual function $\mathbf{r}_{\mathbf{Loop_{a,b}}}$, the Jacobian matrices $\mathbf{J}_{\mathbf{Loop_a}}$ and $\mathbf{J}_{\mathbf{Loop_b}}$, and the covariance matrix $\mathbf{\Sigma}_{\mathbf{Loop_{a,b}}}$ for the CL are defined as shown above.

A loop is connected over two distant timesteps if

- None of the absolute values of the estimated foot velocities during the time span exceed a certain threshold α_v.
- None of the absolute values of the foot accelerations exceed the threshold $\bar{\mathbf{a}}_{foot}$.

A graphical representation of the CL model is provided in Fig. 5.2.

5.8.5 Summary of the Invariant Smoother

The IS extends the capabilities of the InEKF by incorporating a smoothing mechanism. This mechanism leverages the history of the robot's state to improve the accuracy of the pose estimation. The IS is designed to handle the complex dynamics of legged robots, which are subject to non-linearities and uncertainties due to the interaction with the environment. The smoother, built upon the work of Yoon et al. [50] is formulated in a right-invariant form, and utilizes a set of cost functions that capture the dynamics of the system, the observations from the sensors, and the constraints imposed by the contact loop. These cost functions are used to optimize the robot's state estimate by minimizing the error between the predicted and observed states. The observation component plays a critical role by incorporating sensor data to correct the robot's pose. Kinematic measurements relate body pose to foot positions, while LiDAR provides global positional data in the body frame, and GPS offers absolute global positions for outdoor scenarios. These observations enable the smoother to correct drift and improve global position accuracy, making it robust to challenging environments and sensor variability.

5.9 Slip Rejection Method

The assumption of static foot contact is often violated in real-world scenarios, particularly when the robot operates on unstable or slippery surfaces. To address this issue, both estimators in this work are designed to reject slippage by employing the Slip Rejection (SR) method from (Kim et al. [47]). The estimated foot velocity is computed as:

$$\mathbf{v_{foot}} = \mathbf{v_i} + \mathbf{R_i}\mathbf{J_p}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{R_i}(\boldsymbol{\omega} - \mathbf{b}^{\omega})^{\wedge}\mathbf{fk}(\mathbf{q}) \tag{5.78}$$

where $\mathbf{q} \in \mathbb{R}^3$, $\dot{\mathbf{q}} \in \mathbb{R}^3$, and $\mathbf{fk}(\mathbf{q})$ are the joint position, joint velocity, and forward kinematics, respectively. The SR mechanism is triggered when the estimated foot velocity exceeds a predefined threshold. When slippage is detected, the uncertainty associated with the static contact foot assumption is increased to account for the potential deviation from the ideal contact model.

The smoother algorithm offers an additional improvement opportunity. Since foot velocity can be recalculated over a history window, the stability of the contact can be reassessed dynamically. This allows for adaptive modification of the covariance matrix, reflecting the updated contact stability estimation. Consequently, the formulation of the cost function can be adjusted, potentially reformulating the entire estimation process for enhanced accuracy.

The graphical representation in Fig. 5.3 provides an overview of the inputs and methods used in the two state estimation frameworks, highlighting the common inputs and key components (or "blocks") of both the InEKF and IS.

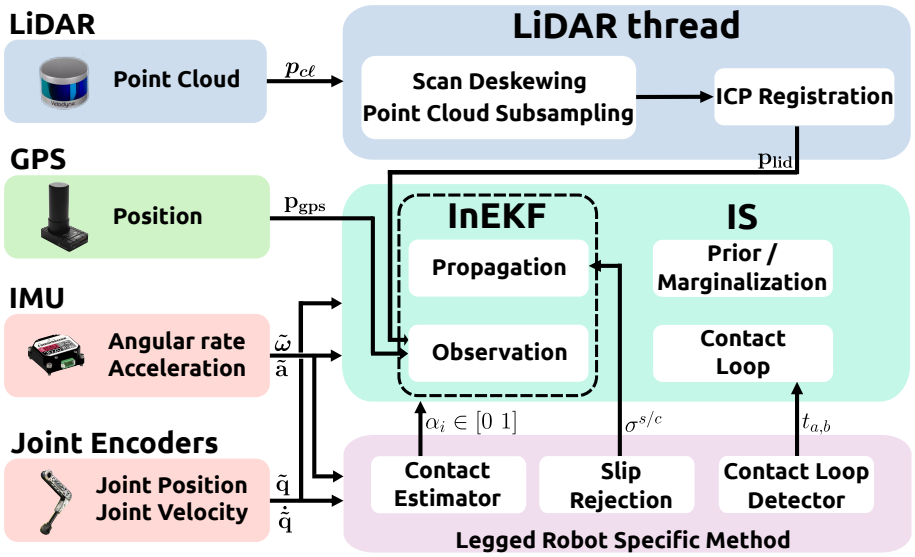


Figure 5.3: Structure of the InEKF and IS: on the left there are the sensor measurements (LiDAR point cloud, GPS position, IMU angular velocity and linear acceleration, and finally joint positions and velocity). LiDAR point cloud is externally processed to obtain a position measurement, that is used, together with the GPS position and leg-kinematics, to obtain the observation cost. The "Legged robot specific method" block includes contact estimation, the slip rejection method described in Section 5.9, and the contact loop detector of Section 5.8.4. The dotted block indicates that Propagation and Observation are common to both estimation frameworks. In fact, the InEKF also takes these two steps. The IS additionally has Prior, Marginalization (Leutenegger et al. [30]), and contact loop blocks.

5.10 Experimental Results

In this section, we present the experimental results of the InEKF and IS algorithms, using sensor data collected from the Hound and Hound2 robotic platforms. These experiments were conducted in two distinct scenarios: a controlled lab indoor environment and a variable outdoor environment. All results are derived from offline processing and analysis of the collected sensor data (Sections 5.10.1 and 5.10.2).

The evaluation focuses on comparing the estimated robot pose against ground truth data. For the indoor experiment, the ground truth pose measurements were provided by a Vicon motion capture system, while for the outdoor experiment, ground truth was obtained from a Holybro RTK GPS system with a helical antenna (Holybro [138]), which delivers centimeter-level precision under open-sky conditions.

The performance of the proposed algorithms was quantified using the two standard metrics already used in Chapter 4: the mean Absolute Trajectory Error (ATE), which assesses global pose estimation accuracy, and the mean Relative Pose Error (RPE), which evaluates local consistency in pose estimation. Additionally, to contextualize our results, we benchmarked InEKF and IS against two state-of-the-art methods. For indoor experiments, we compared them to FAST-LIO (Xu and Zhang [11]), a tightly coupled LiDAR-Inertial Odometry system that integrates LiDAR data and IMU measurements for robust pose estimation. For outdoor experiments, we used KISS-ICP (Vizzo et al. [9]) as the baseline. KISS-ICP is a LiDAR-only odometry system that demonstrated high accuracy (higher than FAST-LIO) in our specific outdoor test environments, making it an appropriate benchmark for this setting.

An analysis of the obtained results in terms of accuracy and time execution is done in Section 5.11.

5.10.1 Indoor Experiment

The 45 kg quadruped robot Hound, equipped with a Livox MID360 LiDAR sensor (Livox [139]), performed walking tests in an indoor environment, designed to simulate real-world uneven terrain, as illustrated in Fig. 5.4. The testing area consisted of a platform built using wooden blocks and steps of varying heights, specifically arranged to challenge the robot's stability and mobility. These variations required the robot to actively adjust its stepping to navigate the uneven block structure effectively.

The varying heights were a critical feature of this experiment, emphasizing the importance of incorporating exteroceptive sensors, such as LiDAR, to address positional

Figure 5.4: Screenshots of the Indoor Experiment with the Hound robot

drift along the z-axis. Positional drift along this axis is inherently unobservable when relying solely on proprioceptive measurements, as previously analyzed by Bloesch et al. [31]. This experiment demonstrated how external sensor data can mitigate such drift, enabling more accurate pose estimation in challenging conditions.

As shown in Fig. 5.5, the integration of LiDAR measurements substantially reduced drift in the z-position. The quantitative improvements are further supported by the mean error values summarized in Tab. 5.1. In these results, the proposed algorithms, leveraging proprioceptive and exteroceptive data, are labeled as InEKF and IS. By contrast, their proprioceptive-only versions, without LiDAR input, are denoted as P-InEKF and P-IS.

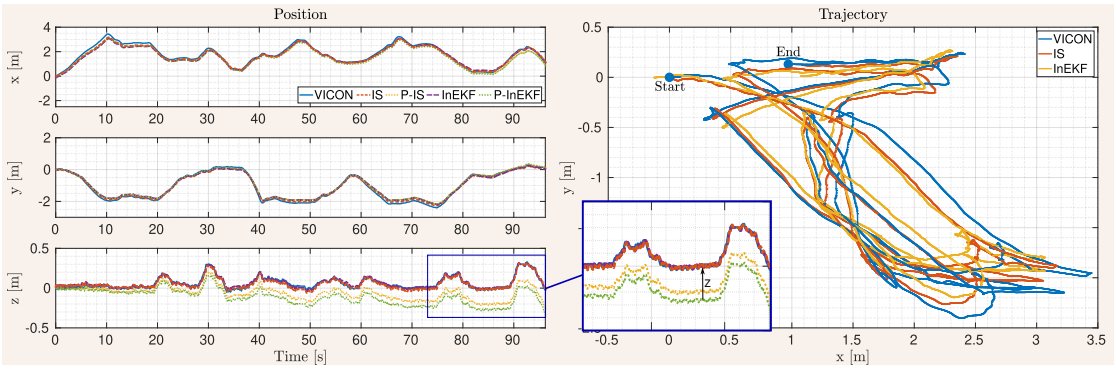


Figure 5.5: Indoor experiment: Comparison between Ground Truth (VICON) and position estimates obtained using proprioceptive-only InEKF (P-InEKF), InEKF, proprioceptive-only IS (P-IS), and IS. The results zoomed on the right demonstrate a clear reduction in **z**-axis drift when LiDAR measurements are incorporated.

Table 5.1: Indoor Experiment with Hound: ATE and RPE over 1 m

Indoor	InEKF	P-InEKF	IS	P-IS	FAST-LIO
ATE [m]	0.18	0.25	**0.12**	0.23	0.50
RPE [m]	0.08	0.09	**0.07**	0.08	0.21

5.10.2 Outdoor Experiment

In this section, we present the results of outdoor experiments conducted to evaluate and compare the performance of the proposed frameworks. The experiments were performed with the 50 kg Hound2 quadruped robot, equipped with a Velodyne VLP16 LiDAR sensor (Ouster [141]) and a Holybro RTK GPS system featuring a helical antenna (Holybro [138]). The robot navigated an outdoor environment along the path depicted in Fig. 5.6b and is shown in operation in Fig. 5.6a.

To assess the frameworks' ability to reduce long-term drift and maintain robustness in outdoor environments, the robot traversed a 300-meter path. The experiment was designed as a closed route, with the robot returning to its starting point. The trajectory data and analysis results are presented in Fig. 5.7 and summarized in Tab. 5.2.

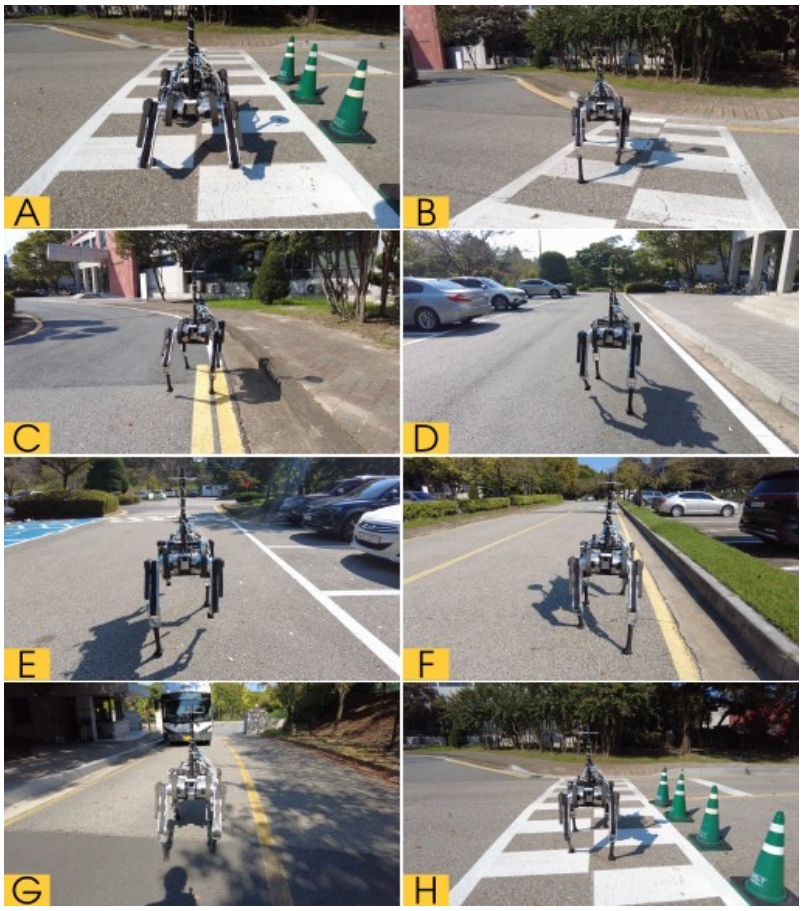
The results demonstrated that the inclusion of exteroceptive measurements, such as LiDAR and GPS, significantly reduced long-term drift compared to proprioceptive-only solutions. This improvement is evident in the z-axis drift reduction and the mean error values reported in Tab. 5.2. The proposed frameworks, combining both proprioceptive and exteroceptive data, are labeled as InEKF and IS, while their proprioceptive-only counterparts are denoted as P-InEKF and P-IS. For a more detailed analysis, we also evaluated the performance of the frameworks using only LiDAR as the exteroceptive measurement, excluding GPS data. These configurations are labeled as L-InEKF and L-IS in Tab. 5.2. To benchmark their performance, we compared the results against KISS-ICP, a state-of-the-art LiDAR-only odometry algorithm. The results indicated that even when GPS was excluded, the proposed frameworks (L-InEKF and L-IS) outperformed KISS-ICP, demonstrating the added value of integrating leg-kinematics data into state estimation for legged robots.

Table 5.2: Outdoor Experiment with Hound2: ATE and RPE over 1 m

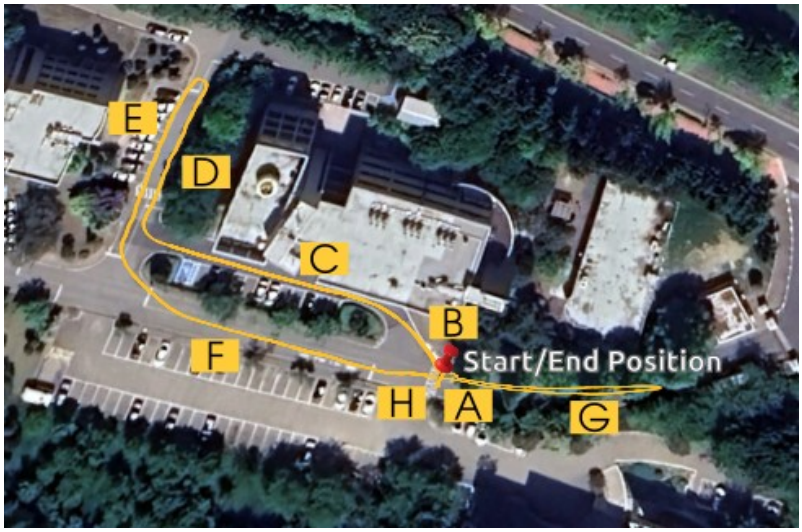
Outdoor	InEKF	P-InEKF	IS	P-IS	L-InEKF	L-IS	KISS-ICP
ATE [m]	0.17	6.57	**0.15**	6.34	1.68	1.37	2.15
RPE [m]	0.07	0.10	**0.06**	0.09	0.09	0.08	0.13

5.11 Discussion

In this chapter, we proposed two novel methods for robot pose estimation that combine the strengths of filter-based and smoother-based approaches with the invariance prop-



(a) Screenshots of the Outdoor Experiment with the Hound2 robot.



(b) Top view of the path walked by the Hound2 robot during the outdoor experiment. The original picture is obtained with Google Earth (Google [140])

Figure 5.6: Outdoor Experiment with the Hound2 robot: each letter indicates the position of the robot on the traversed path.

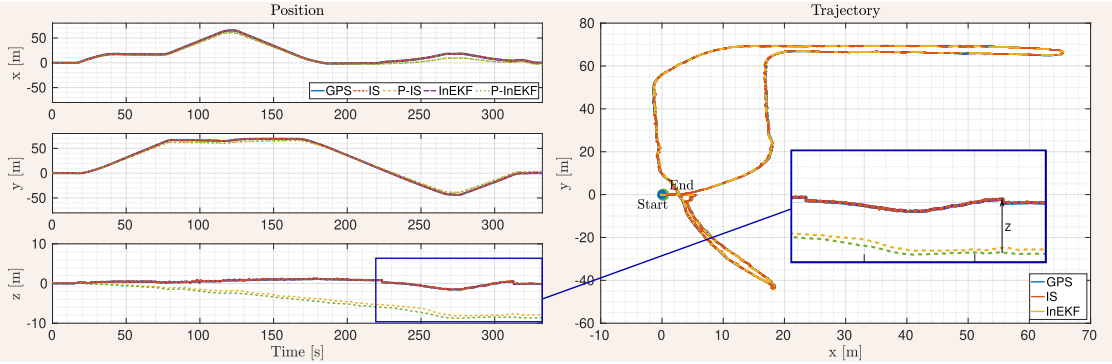


Figure 5.7: Outdoor experiment Ground Truth (GPS) vs. position estimated using proprioceptive-only InEKF (P-InEKF), InEKF, proprioceptive-only IS (P-IS), and IS. The results clearly show the improvement in the **z**-axis drift when using LiDAR and GPS measurements.

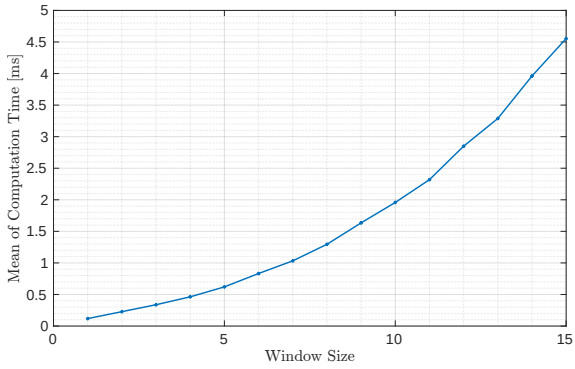
erties of the $SE_k(3)$ manifold. The proposed methods, InEKF and IS, were validated through experiments on two quadruped robots, Hound and Hound2, in both controlled indoor and realistic outdoor environments.

The experimental results demonstrated that the proposed methods consistently outperformed state-of-the-art algorithms such as FAST-LIO and KISS-ICP. Both InEKF and IS effectively corrected positional drift along the z-axis in indoor and outdoor scenarios, highlighting their robustness in managing vertical displacement errors. In the outdoor experiments, the LiDAR-only versions of the proposed methods exhibited reduced accuracy compared to the versions that incorporated GPS data. This discrepancy is expected, as outdoor LiDAR odometry can be affected by challenges such as dynamic obstacles, sparse or unreliable features, and environmental complexity. Integrating GPS data was critical for improving pose estimation in such cases. Despite this limitation, the LiDAR-only configurations of InEKF and IS still outperformed the benchmark methods, demonstrating the efficacy of the proposed frameworks.

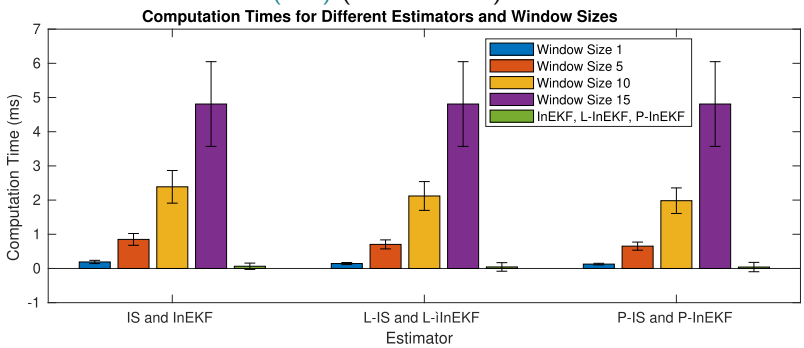
5.11.1 Considerations about time execution

In all tests, the IS algorithm achieved better positional accuracy than InEKF. This advantage is likely due to IS's ability to optimize pose estimation by leveraging both past and current observations within a specified time window. However, the InEKF provided comparable accuracy while requiring significantly less computational power, making it highly suitable for real-time applications.

During the experiments, the IS algorithm employed a time window of 15 frames to ensure all relevant updates from LiDAR and GPS data were captured. While this larger window size improved positional accuracy, it increased computation time.



(a) Mean of Computation Time with increasing Window Size (WS) (from 1 to 15).



(b) Comparison of average computation times for the proposed frameworks.

Figure 5.8: Top: Mean of the computation time of the IS, when increasing the WS. Bottom: Comparison of average computation times for three variants of the invariant smoothing estimator (IS, L-IS, P-IS) and their InEKF-based counterparts (InEKF, L-InEKF, P-InEKF) across different WS (1, 5, 10, and 15). Each bar represents the mean computation time (in milliseconds), with error bars indicating one standard deviation. The results show how increasing the window size affects the computational load of the IS.

- **IS Algorithm** (window size of 15): 4.5 milliseconds per iteration on average on a laptop with an Intel Core vPro Essential i7 processor.
- **InEKF Algorithm**: 0.06 milliseconds per iteration on the same machine

The execution time of the InEKF is consistent with the results reported for MUSE during online experiments, where the state estimator provided real-time feedback to the controller (Section 4.6.8).

To assess computational scalability, the IS algorithm was also tested with reduced window sizes. In particular, a comparison of the computation time of the IS with a window size of 1, 5, 10, 15 and the InEKF is given in Fig. 5.8b, where the plot shows the mean distribution of computation times and their standard deviation. Fig. 5.8a, instead shows how the average of the computation time increases when the WS of the IS increases from 1 to 15. In the configuration with WS 1 - which hence exploits the

same measurements correction as the InEKF - the IS algorithm achieved an execution time of 0.18 milliseconds per iteration but with slightly reduced positional accuracy, giving results that are comparable to those obtained with the InEKF. For instance, in the indoor experiment, The ATE and RPE values for a WS of 1, 5, 10, and 15, and the ATE and RPE values for the InEKF are in Tab. 5.3.

Table 5.3: ATE and RPE for IS with different WS and for the InEKF

	IS WS:1	IS WS:5	IS WS:10	IS WS:15	InEKF
ATE [m]	0.14	0.13	0.13	0.12	0.18
RPE [m]	0.08	0.08	0.07	0.07	0.08

From Tab. 5.3, it is evident that the IS is more accurate than the InEKF, even with a WS of 1. However, in this case, the computation time for the IS with a WS of 1 remains higher than that of the InEKF (0.18 ms vs 0.06 ms, respectively). Nevertheless, the results of both the IS and the InEKF are still comparable overall. These results highlight the trade-off between computational efficiency and accuracy. The comparable performance in terms of accuracy of IS and InEKF demonstrates that InEKF is particularly well-suited for real-time applications requiring low-latency pose estimation. Meanwhile, the full-window IS configuration, with its higher computational demand, is better suited in offline applications where precision is critical, such as detailed mapping or high-fidelity localization tasks.

5.11.2 Limitations

While the incorporation of LiDAR greatly enhanced z-axis performance, our implementation did not include orientation correction due to the challenges related to the incorporation of the rotation in the Euclidean measurement space, while maintaining an invariant structure. This omission represents a limitation of the current approach and suggests an area for future research, which will aim to address this limitation and, additionally, explore methods for integrating orientation correction in invariant frameworks to enhance overall pose estimation accuracy.

Another limitation arises when GPS data is unavailable or unreliable (e.g. indoor or in case of cloudy and overcast skies). In such cases, the proposed method relies solely on LiDAR and leg-kinematics data to correct drift. However:

- Drift cannot be corrected with leg-kinematics alone.

- LiDAR odometry is subject to limitations in certain environments, such as feature-sparse areas or dynamic scenes.

To improve robustness and accuracy in such scenarios, incorporating additional sensor modalities, such as visual odometry, could enhance the proposed method's effectiveness.

5.12 Conclusion

In this chapter, we introduced two novel methods for robot pose estimation that leverage the advantages of the invariance properties of the $SE_k(3)$ manifold. The proposed methods were evaluated on two quadruped robots, Hound and Hound2, in both indoor and outdoor environments. The results demonstrated that the methods outperformed state-of-the-art algorithms such as FAST-LIO and KISS-ICP.

Notably, both the InEKF and IS were able to correct drift along the z-axis, as expected. However, in the outdoor experiment, the LiDAR-only version of the method showed reduced accuracy. This outcome was expected, as LiDAR odometry in outdoor environments can be affected by external factors such as moving people, a lack of reliable features, or environmental complexity. In such cases, incorporating GPS data proved critical for achieving more accurate pose estimations.

Nevertheless, even in the LiDAR-only configuration, the proposed method outperformed the state-of-the-art algorithms in both indoor and outdoor scenarios. Specifically, in the indoor experiment, the proposed methods outperformed FAST-LIO in terms of both ATE and RPE, while in the outdoor experiment, L-IS and L-InEKF outperformed KISS-ICP in both metrics.

The IS demonstrated superior position accuracy across all tests, but the InEKF achieved comparable results. During the experiments, a window size of 15 was selected for the IS to ensure all potential updates from LiDAR and GPS were captured, minimizing the risk of losing important exteroceptive information. While this choice enhanced position accuracy, it also increased computational time.

For future work, we plan to:

- Perform a thorough comparison of the proposed methods with MUSE. The use of Lie Groups in the invariant state estimation framework could offer advantages over MUSE, which relies on a LTV-KF in its Sensor Fusion module. Our results have shown that the InEKF achieves comparable execution times to MUSE; however, we have only evaluated the InEKF on offline data, not in online experiments. In contrast, the IS has demonstrated higher accuracy than the InEKF. It would be

valuable to investigate whether this superior accuracy holds up against MUSE, particularly in terms of rotational error. This is especially relevant because MUSE applies orientation corrections using exteroceptive data, which the IS does not currently incorporate.

- Include orientation correction in the proposed method to further enhance accuracy.
- Integrate additional sensor modalities, such as visual odometry, to improve the robustness and performance of the approach, particularly in challenging environments where LiDAR or GPS data alone may be insufficient.
- Fuse the proprioceptive-based frameworks in Lie Group with exteroceptive measurements in a tightly coupled manner and compare the performance with the current approach.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

Throughout the three years of this Ph.D. research, we studied and explored multiple approaches to address the challenges of state estimation for legged robots. Three central questions guided our work:

1. Can we make the robot aware of its surrounding environment, particularly detecting whether the terrain it is traversing is slippery or not?
2. How can slip detection improve state estimation?
3. Can we provide robust and accurate state estimation, and which techniques are best suited for this purpose?

In this thesis, we proposed methods and frameworks to answer these questions. The slip detection algorithm enabled slippery-terrain awareness, improving the state estimation in challenging conditions. MUSE, a multi-sensor state estimator, demonstrated fast and accurate performance, suitable for real-time applications. Additionally, the invariant frameworks (InEKF and IS) leveraged Lie theory to provide mathematically consistent and robust estimation.

First, in Chapter 3 recognizing the importance of terrain awareness for legged robots, we proposed an algorithm to detect slippage. This algorithm can enable robots to identify and respond to unstable or slippery terrain, improving their ability to adapt to different surfaces and maintain stable locomotion. The slippage detection method is a key step

toward ensuring reliable operation in diverse environments, addressing a fundamental limitation of traditional state estimators. In our case, slip detection has been used to improve state estimation, allowing us to discard potentially unreliable measurements that can negatively affect the estimation process.

Building on this, in Chapter 4 we introduced MUSE, a multi-sensor state estimation framework based on Kalman filtering. MUSE integrates data from multiple sensors, including IMUs, encoders, force/torque sensors, cameras, and LiDARs, to provide accurate and reliable state estimation for quadruped robots. MUSE is designed to handle the challenges of real-world environments, such as slippery or uneven terrain, and is built to be modular and flexible, allowing it to interface with various robot platforms and sensor configurations. MUSE demonstrated its capability to fuse data from multiple sensors efficiently, achieving both speed and accuracy. Its reliability and performance in online experiments make it particularly suited for scenarios where computational efficiency and accuracy are critical, such as in robotic locomotion and navigation tasks.

Finally, in Chapter 5 we developed two advanced state estimation frameworks based on Lie theory: the Invariant Extended Kalman Filter (InEKF) and the Invariant Smoother (IS). These frameworks leverage the mathematical properties of Lie groups to achieve better consistency and robustness in state estimation. The InEKF proved to be highly efficient, making it suitable for real-time applications, while the invariant smoother demonstrated superior accuracy, making it ideal for offline or mapping applications where computational time is less constrained.

In summary, this thesis presents a comprehensive exploration of state estimation techniques for legged robots. From slippage detection to the development of advanced multi-sensor and Lie-theory-based frameworks, the work covered a wide spectrum of approaches and mathematical tools. The proposed methods address key challenges in the field, contributing solutions that are both theoretically sound and practically relevant.

6.2 Future Works

Nevertheless, despite the progress achieved, this journey has raised several questions that warrant further exploration. In particular:

- **Which method is better, MUSE or the invariant frameworks?** Each has its own strengths and limitations, and determining the most suitable approach requires further investigation under diverse scenarios.
- **Which algorithm is more suitable for SLAM and mapping applications?**

This question becomes especially relevant when dealing with dynamic environments, where current methods often fail. Understanding how to best handle dynamic elements in SLAM remains an open challenge.

These unanswered questions point toward future directions for research. A key direction is a comprehensive comparison of MUSE and the proposed invariant methods, InEKF and IS. While this thesis has demonstrated the individual strengths of each framework, a detailed evaluation of their performance under diverse conditions will provide a clearer understanding of their respective advantages and limitations. Specifically:

Accuracy vs. Efficiency: MUSE has demonstrated high-speed performance, making it suitable for real-time applications. The invariant methods, particularly the smoother ones, excel in accuracy but come at a higher computational cost. A direct comparison could help identify scenarios where each approach is most effective.

Robustness in Complex Environments: Comparing how MUSE and the invariant frameworks handle slippage, sensor noise, or loss of data (e.g., from GPS or LiDAR) could provide insights into their robustness.

Applicability to Offline vs. Real-Time Applications: MUSE is designed for real-time operation, while the smoother-based approach may be more suited for offline mapping or high-precision localization tasks. Evaluating their performance in these contexts could help determine the best use cases for each method. Furthermore, despite the computation time for the InEKF suggested it is suitable for real-time applications, our algorithm has never been tested online on a real robot. This is a crucial step to validate the algorithm and to understand its performance in real-world scenarios.

Convergence properties: XKF vs. InEKF The convergence properties of the two filters, the XKF and the InEKF, warrant a detailed comparison, particularly in terms of attitude estimation stability and robustness to initialization errors. The cascade structure of NLO+XKF has been shown to exhibit **global** stability in most scenarios, even under large initialization errors, as demonstrated by Fink and Semini [34], Mahony et al. [107, 112]. This robustness arises from the NLO, which is Lyapunov-stable and capable of handling substantial initial orientation errors while reliably converging to the true attitude. Once the NLO provides a bounded and convergent estimate, the LTV-KF operates within a regime where linearization errors are limited. Furthermore, the XKF, applied in a linear time-varying manner along the NLO trajectory, delivers near-optimal

corrections of the Gaussian noise, thereby enhancing overall accuracy. In contrast, the InEKF offers a fundamentally different approach by preserving invariance on the rotation group $\mathrm{SO}(3)$. This ensures that the estimated attitude remains consistent with the underlying manifold, an important feature for applications requiring strict geometric adherence. However, unlike the NLO, the InEKF has been shown to provide **local** or **semi-global** stability, as highlighted in (Barrau and Bonnabel [124]), and does not guarantee global convergence when initialization errors are large. A systematic comparison of these two filters in terms of their attitude estimation capabilities could yield valuable insights into the stability characteristics of the InEKF, particularly its sensitivity to initialization errors. Moreover, exploring their relative performance under conditions of small initialization errors presents an interesting direction for future research. Such an investigation would not only clarify the operational limits of each filter but also guide their application in practical scenarios where trade-offs between stability, accuracy, and computational efficiency are critical.

This comparison could be conducted through extensive real-world experiments, covering diverse terrains, dynamic environments, and varying sensor configurations. Such a study would not only validate the proposed frameworks but also provide a clearer roadmap for selecting or hybridizing methods for specific robotic applications. A deeper exploration of the trade-offs between different estimation techniques, their applicability to SLAM, and their performance in dynamic and complex environments will help advance the field of legged robotics. As the field progresses, the insights gained from this work form a foundation for further discoveries and innovations in robust state estimation, enabling legged robots to navigate and adapt to continuously changing environments.

Additionally, future research will focus on enhancing key components of the proposed frameworks to improve overall estimation performance. Advanced terrain estimation techniques, such as real-time estimation of friction coefficients and terrain properties such as inclination and softness, will be crucial for enabling fully autonomous operations. Furthermore, more reliable mapping methods will support long-term autonomy by helping robots build a comprehensive understanding of their environments. These advancements will enhance the robot's ability to autonomously navigate complex terrain, maintain stability, and perform sophisticated tasks.

Bibliography

[1] Ylenia Nisticò, Shamel Fahmi, Lucia Pallottino, Claudio Semini, and Geoff Fink. On slip detection for quadruped robots. *Sensors*, 22(8), 2022. ISSN 1424-8220. DOI: 10.3390/s22082967.

[2] David Wisth, Marco Camurri, and Maurice Fallon. VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots. *IEEE Transactions on Robotics*, 2022. DOI: 10.1109/TRO.2022.3193788.

[3] Hyunjun Lim, Byeongho Yu, Yeeun Kim, Joowoong Byun, Soonpyo Kwon, Haewon Park, and Hyun Myung. WALK-VIO: Walking-motion-adaptive leg kinematic constraint visual-inertial odometry for quadruped robots. *arXiv preprint arXiv:2111.15164*, 2021.

[4] Yeeun Kim, Byeongho Yu, Eungchang Mason Lee, Joonha Kim, Haewon Park, and Hyun Myung. STEP: State estimator for legged robots using a preintegrated foot velocity factor. *IEEE Robotics and Automation Letters*, 7(2):4456–4463, 2022. DOI: 10.1109/LRA.2022.3150844.

[5] Unitree. Aliengo by Unitree Robotics https://www.unitree.com/products/aliengo/, 2019. Last accessed in December 2024.

[6] Marco Camurri, Milad Ramezani, Simona Nobili, and Maurice Fallon. Pronto: A multi-sensor state estimator for legged robots in real-world scenarios. *Frontiers in Robotics and AI*, 7, 2020. ISSN 2296-9144. DOI: 10.3389/frobt.2020.00068.

[7] Michael Bloesch, Michael Burri, Hannes Sommer, Roland Siegwart, and Marco Hutter. The two-state implicit filter recursive estimation for mobile robots. *IEEE Robotics and Automation Letters*, 3(1):573–580, 2018. DOI: 10.1109/LRA.2017.2776340.

[8] Kenny Chen, Ryan Nemiroff, and Brett T. Lopez. Direct LiDAR-Inertial Odometry: Lightweight LIO with Continuous-Time Motion Correction. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3983–3989, 2023. DOI: 10.1109/ICRA48891.2023.10160508.

[9] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. KISS-ICP: In defense of point-to-point ICP – simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters*, 8(2):1029–1036, February 2023. DOI: 10.1109/lra.2023.3236571.

[10] Young-Ha Shin, Seungwoo Hong, Sangyoung Woo, JongHun Choe, Harim Son, Gijeong Kim, Joon-Ha Kim, KangKyu Lee, Jemin Hwangbo, and Hae-Won Park. Design of KAIST HOUND, a Quadruped Robot Platform for Fast and Efficient Locomotion with Mixed-Integer Nonlinear Optimization of a Gear Train. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6614–6620, 2022. DOI: 10.1109/ICRA46639.2022.9811755.

[11] Wei Xu and Fu Zhang. FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021. DOI: 10.1109/LRA.2021.3064227.

[12] Peter Fankhauser and Marco Hutter. ANYmal: a unique quadruped robot conquering harsh environments. *Research Features*, 126:54–57, May 2018. doi: 10.3929/ethz-b-000262484.

[13] Boston Dynamics. Spot-the Agile Mobile Robot, 2020. URL https://bostondynamics.com/products/spot. Last accessed in December 2024.

[14] Evan Ackerman. Boston dynamics' spot is helping chernobyl move towards safe decommissioning, 2020. URL https://spectrum.ieee.org/boston-dynamics-spot-chernobyl. Last accessed in December 2024.

[15] Claudio Semini, Victor Barasuol, Michele Focchi, Chundri Boelens, Mohamed Emara, Salvatore Casella, Octavio Villarreal, Romeo Orsolino, Geoff Fink, Shamel Fahmi, Gustavo Medrano-Cerda, Dhinesh Sangiah, Jack Lesniewski, Kyle Fulton, Michel Donadon, Mike Baker, and Darwin G Caldwell. Brief introduction to the quadruped robot HyQReal. In *Italian Conference on Robotics and Intelligent Machines (I-RIM)*, pages 1–2, Rome, October 2019.

[16] Claudio Semini and Matteo Gatti. Vinum project, 2024. URL https://vinum-robot.eu/. Last accessed in December 2024.

[17] Paolo Guadagna, M Fernandes, F Chen, Alessandro Santamaria, Tao Teng, Tommaso Frioni, DG Caldwell, Stefano Poni, C Semini, and Matteo Gatti. Using deep learning for pruning region detection and plant organ segmentation in dormant spur-pruned grapevines. *Precision Agriculture*, 24(4):1547–1569, 2023. DOI: 10.1007/s11119-023-10006-y.

[18] DFKI Robotics Innovation Center. Crex, crater explorer, 2020. URL https://robotik.dfki-bremen.de/en/research/robot-systems/crex. Last accessed in December 2024.

[19] Alexander Dettmann, Steffen Planthaber, Vinzenz Bargsten, Raul Dominguez, Gianluca Cerilli, Marco Marchitto, Geoff Fink, Michele Focchi, Victor Barasuol, Claudio Semini, et al. Towards a generic navigation and locomotion control system for legged space exploration. In *16th Symposium on Advanced Space Technologies in Robotics and Automation*, 2022.

[20] Lorenzo Amatucci, Giulio Turrisi, Angelo Bratta, Victor Barasuol, and Claudio Semini. VERO: A vacuum-cleaner-equipped quadruped robot for efficient litter removal. *Journal of Field Robotics*, 2024. DOI: 10.1002/rob.22350.

[21] Evan Ackerman. Robot dog cleans up beaches with foot-mounted vacuums. thanks to VERO, Genoa has fewer cigarette butts littering the ground, 2024. URL https://spectrum.ieee.org/robot-dog-vacuum. Last accessed in December 2024.

[22] Boston Dynamics Inc. Atlas | Partners in Parkour, 2021. URL https://www.youtube.com/watch?v=tF4DML7FIWk&ab_channel=BostonDynamics. Last accessed in December 2024.

[23] Unitree Robotics. Unitree G1, Humanoid agent AI avatar, 2024. URL https://www.unitree.com/g1. Last accessed in December 2024.

[24] Figure. Figure is the first-of-its-kind AI robotics company bringing a general purpose humanoid to life., 2024. URL https://www.figure.ai/. Last accessed in December 2024.

[25] Timothy D. Barfoot. *State Estimation for Robotics: Second Edition*. Cambridge University Press, 2 edition, 2024. DOI: 10.1017/9781009299909.

[26] KVH Industries. KVH P-1775 IMU, 2024. URL https://canalgeomatics.com/product/kvh-p-1775-imu/. Last accessed in December 2024.

[27] Avago Technologies. AEDA-3300 Series, Ultra Miniature, High Resolution Incremental Kit Encoders, 2024. URL https://media.digikey.com/pdf/Data%20Sheets/Avago%20PDFs/AEDA-3300%20Series.pdf. Last accessed in December 2024.

[28] Inc Technical Laboratory Systems. HEX 6-Axis Force/Torque sensor, 2024. URL https://tech-labs.com/products/hex-6-axis-forcetorque-sensor. Last accessed in December 2024.

[29] Michael Bloesch and Marco Hutter. *Technical Implementations of the Sense of Balance*, pages 1–29. Springer Netherlands, Dordrecht, 2016. ISBN 978-94-007-7194-9. DOI: 10.1007/978-94-007-7194-9_69-2.

[30] 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. DOI: 10.1177/0278364914554813.

[31] 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. *Robotics*, 17:17–24, July 2013. DOI: 10.15607/RSS.2012.VIII.003.

[32] Marco Hutter, Christian Gehring, Michael Bloesch, Mark A Hoepflinger, C David Remy, and Roland Siegwart. StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion. In *Adaptive Mobile Robotics*, pages 483–490. World Scientific, September 2012. doi: 10.1142/9789814415958_0062.

[33] Nicholas Rotella, Michael Bloesch, Ludovic Righetti, and Stefan Schaal. State estimation for a humanoid robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 952–958, 2014. DOI: 10.1109/IROS.2014.6942674.

[34] Geoff Fink and Claudio Semini. Proprioceptive sensor fusion for quadruped robot state estimation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10914–10920, 2020. DOI: 10.1109/IROS45743.2020.9341521.

[35] Claudio Semini, Nikolaos Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. Design of HyQ – a hydraulically and electrically actuated quadruped robot. *IMechE Part I: Journal of Systems and Control Engineering*, 225(6):831–849, feb 2011. DOI: 10.1177/0959651811402275.

[36] Ross Hartley, Maani Ghaffari, Ryan M Eustice, and Jessy W Grizzle. Contact-aided invariant extended Kalman filtering for robot state estimation. *The International Journal of Robotics Research*, 39(4):402–430, 2020. DOI: 10.1177/0278364919894385.

[37] Agility Robotics. Cassie bipedal robot, 2024. https://agilityrobotics.com/, Last accessed in October 2024.

[38] Prashanth Ramadoss, Giulio Romualdi, Stefano Dafarra, Francisco Javier Andrade Chavez, Silvio Traversaro, and Daniele Pucci. DILIGENT-KIO: A proprioceptive base estimator for humanoid robots using extended kalman filtering on matrix lie groups. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2904–2910, 2021. DOI: 10.1109/ICRA48506.2021.9561248.

[39] Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes Von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, et al. The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural networks*, 23(8-9):1125–1134, 2010. DOI: 10.1016/j.neunet.2010.08.010.

[40] Marco Camurri, Maurice Fallon, Stéphane Bazeille, Andreea Radulescu, Victor Barasuol, Darwin G. Caldwell, and Claudio Semini. Probabilistic contact estimation and impact detection for state estimation of quadruped robots. *IEEE Robotics and Automation Letters*, 2(2):1023–1030, 2017. DOI: 10.1109/LRA.2017.2652491.

[41] 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 *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6058–6064. IEEE, 2013. DOI: 10.1109/IROS.2013.6697236.

[42] Fabian Jenelten, Jemin Hwangbo, Fabian Tresoldi, C Dario Bellicoso, and Marco Hutter. Dynamic locomotion on slippery ground. *IEEE Robotics and Automation Letters*, 4(4):4170–4176, October 2019. DOI: 10.1109/LRA.2019.2931284.

[43] Marco Hutter, Christian Gehring, Andreas Lauber, Fabian Gunther, Carmine Dario Bellicoso, Vassilios Tsounis, Péter Fankhauser, Remo Diethelm, Samuel Bachmann, Michael Blösch, et al. ANYmal - toward legged robots for harsh environments. *Advanced Robotics*, 31(17):918–931, 2017. DOI: 10.1080/01691864.2017.1378591.

[44] David Wisth, Marco Camurri, and Maurice Fallon. Preintegrated velocity bias estimation to overcome contact nonlinearities in legged robot odometry. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 392–398. IEEE, 2020. DOI: 10.1109/ICRA40945.2020.9197214.

[45] Médéric Fourmy, Thomas Flayols, Pierre-Alexandre Léziart, Nicolas Mansard, and Joan Solà. Contact forces preintegration for estimation in legged robotics using factor graphs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1372–1378. IEEE, 2021. DOI: 10.1109/ICRA48506.2021.9561037.

[46] Shamel Fahmi, Geoff Fink, and Claudio Semini. On State Estimation for Legged Locomotion Over Soft Terrain. *IEEE Sensors Letters*, 5(1):1–4, 2021. DOI: 10.1109/LSENS.2021.3049954.

[47] Joon-Ha Kim, Seungwoo Hong, Gwanghyeon Ji, Seunghun Jeon, Jemin Hwangbo, Jun-Ho Oh, and Hae-Won Park. Legged robot state estimation with dynamic contact event information. *IEEE Robotics and Automation Letters*, 6(4):6733–6740, 2021.

[48] Shangru Yang, Qingjun Yang, Rui Zhu, Zhenyang Zhang, Congfei Li, and Hu Liu. State estimation of hydraulic quadruped robots using invariant-EKF and kinematics with neural networks. *Neural Computing and Applications*, pages 1–14, 2023. DOI: 10.1007/s00521-023-08755-y.

[49] Hilton Marques Souza Santana, João Carlos Virgolino Soares, Ylenia Nisticò, Marco Antonio Meggiolaro, and Claudio Semini. Proprioceptive state estimation for quadruped robots using invariant kalman filtering and scale-variant robust cost functions. *arXiv preprint arXiv:2410.05256*, 2024.

[50] Ziwon Yoon, Joon-Ha Kim, and Hae-Won Park. Invariant smoother for legged robot state estimation with dynamic contact event information. *IEEE Transactions on Robotics*, 40:193–212, 2024. DOI: 10.1109/TRO.2023.3328202.

[51] Intel RealSense. Depth Camera D435, 2024. URL https://www.intelrealsense.com/depth-camera-d435/. Last accessed in December 2024.

[52] Ouster. Vlp-16 Mid-range LiDAR sensor, 2024. URL https://ouster.com/products/hardware/vlp-16. Last accessed in December 2024.

[53] Dongjae Lee, Minwoo Jung, Wooseong Yang, and Ayoung Kim. Lidar odometry survey: recent advancements and remaining challenges. *Intelligent Service Robotics*, 17(2):95–118, 2024. DOI: 10.1007/s11370-024-00515-8.

[54] Basheer Al-Tawil, Thorsten Hempel, Ahmed Abdelrahman, and Ayoub Al-Hamadi. A review of visual SLAM for robotics: evolution, properties, and future applications. *Frontiers in Robotics and AI*, 11:1347985, 2024. DOI: 10.3389/frobt.2024.1347985.

[55] Ali Tourani, Hriday Bavle, Jose Luis Sanchez-Lopez, and Holger Voos. Visual SLAM: What are the current trends and what to expect? *Sensors*, 22(23), 2022. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/22/23/9297. DOI: 10.3390/s22239297.

[56] ROS. tf, 2024. URL https://wiki.ros.org/tf. Last accessed in January 2025.

[57] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004. DOI: 10.1109/CVPR.2004.1315094.

[58] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics & Automation Magazine*, 18(4):80–92, 2011. DOI: 10.1109/MRA.2011.943233.

[59] Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90, 2012. DOI: 10.1109/MRA.2012.2182810.

[60] 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. DOI: 10.1109/TRO.2016.2624754.

[61] Joan Sola, Andre Monin, and Michel Devy. BiCamSLAM: Two times mono is more than stereo. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4795–4800, 2007. DOI: 10.1109/ROBOT.2007.364218.

[62] Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. *Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera*, pages 235–252. Springer International Publishing, Cham, 2017. ISBN 978-3-319-29363-9. DOI: 10.1007/978-3-319-29363-9_14.

[63] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5): 1147–1163, 2015. DOI: 10.1109/TRO.2015.2463671.

[64] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33 (5):1255–1262, 2017. DOI: 10.1109/TRO.2017.2705103.

[65] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. DOI: 10.1109/tro.2021.3075644.

[66] 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, 2016. DOI: 10.1109/TRO.2016.2623335.

[67] Gabriel Fischer Abati, João Carlos Virgolino Soares, Vivian Suzano Medeiros, Marco Antonio Meggiolaro, and Claudio Semini. Panoptic-SLAM: Visual SLAM in dynamic environments using panoptic segmentation. In *2024 21st International Conference on Ubiquitous Robots (UR)*, pages 01–08, 2024. DOI: 10.1109/UR61395.2024.10597506.

[68] Weicai Ye, Xinyue Lan, Shuo Chen, Yuhang Ming, Xingyuan Yu, Hujun Bao, Zhaopeng Cui, and Guofeng Zhang. Pvo: Panoptic visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9579–9589, June 2023.

[69] Zhengyou Zhang. *Iterative Closest Point (ICP)*, pages 718–720. Springer International Publishing, Cham, 2021. ISBN 978-3-030-63416-2. DOI: 10.1109/TRO.2017.2705103.

[70] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009. DOI: http://dx.doi.org/10.15607/RSS.2009.V.021.

[71] Kenny Chen, Brett T. Lopez, Ali-akbar Agha-mohammadi, and Ankur Mehta. Direct lidar odometry: Fast localization with dense point clouds. *IEEE Robotics and Automation Letters*, 7(2):2000–2007, 2022. DOI: 10.1109/LRA.2022.3142739.

[72] Cyrill Stachniss. Running KISS-ICP on KITTI, 2022. URL https://www.youtube.com/watch?v=kMMH8rA1ggI&ab_channel=CyrillStachniss. Last accessed in December 2024.

[73] Tixiao Shan and Brendan Englot. LeGO-LOAM: Lightweight and ground-optimized LiDAr odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018. doi: 10.1109/IROS.2018.8594299. DOI: 10.1109/IROS.2018.8594299.

[74] Simone Ferrari, Luca Di Giammarino, Leonardo Brizi, and Giorgio Grisetti. MAD-ICP: It is all about matching data–robust and informed lidar odometry. *arXiv preprint arXiv:2405.05828*, 2024.

[75] Ji Zhang, Sanjiv Singh, et al. LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2(9), pages 1–9. Berkeley, CA, 2014. DOI: 10.15607/RSS.2014.X.007.

[76] Tiziano Guadagnino, Xieyuanli Chen, Matteo Sodano, Jens Behley, Giorgio Grisetti, and Cyrill Stachniss. Fast sparse lidar odometry using self-supervised feature selection on intensity images. *IEEE Robotics and Automation Letters*, 7 (3):7597–7604, 2022. DOI: 10.1109/LRA.2022.3184454.

[77] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10): 1053–1072, 2017. DOI: 10.1177/0278364917728574.

[78] 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. DOI: 10.1109/TRO.2018.2853729.

[79] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5135–5142. IEEE, 2020. DOI: 10.1109/IROS45743.2020.9341176.

[80] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. FAST-LIO2: Fast Direct LiDAR-Inertial Odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022. DOI: 10.1109/TRO.2022.3141876.

[81] Jonas Beuchert, Marco Camurri, and Maurice Fallon. Factor graph fusion of raw GNSS sensing with IMU and LiDAR for precise robot localization without a base station. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8415–8421. IEEE, 2023. DOI: http://dx.doi.org/10.1109/ICRA48891.2023.10161522.

[82] Hilti. Hilti SLAM Challenge, 2021. URL https://hilti-challenge.com/index.html. Last accessed in January 2025.

[83] Milad Ramezani, Kasra Khosoussi, Gavin Catt, Peyman Moghadam, Jason Williams, Paulo Borges, Fred Pauling, and Navinda Kottege. Wildcat: Online continuous-time 3d lidar-inertial slam. *arXiv preprint arXiv:2205.12595*, 2022.

[84] Hyungtae Lim, Daebeom Kim, Beomsoo Kim, and Hyun Myung. Adalio: Robust adaptive lidar-inertial odometry in degenerate indoor environments. In *2023 20th International Conference on Ubiquitous Robots (UR)*, pages 48–53, 2023. doi: 10.1109/UR57808.2023.10202252.

[85] Hyungtae Lim, Suyong Yeon, Soohyun Ryu, Yonghan Lee, Youngji Kim, Jaeseong Yun, Euigon Jung, Donghwan Lee, and Hyun Myung. A single correspondence is enough: Robust global registration to avoid degeneracy in urban environments. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8010–8017, 2022. doi: 10.1109/ICRA46639.2022.9812018.

[86] Boston Dynamics Inc. Atlas and beyond: the world's most dynamic robots, 2024. URL https://bostondynamics.com/atlas/. Last accessed in December 2024.

[87] NASA. VALKYRIE, NASA's first bipedal humanoid robot, 2023. https://www.nasa.gov/wp-content/uploads/2023/06/r5-fact-sheet.pdf, Last accessed in October 2024.

[88] Defense Advanced Research Projects Agency. DARPA Robotics Challenge (DRC), 2015. URL https://www.darpa.mil/program/darpa-robotics-challenge. Last accessed in December 2024.

[89] Defense Advanced Research Projects Agency. DARPA Subterranean (SubT) Challenge, 2017. URL https://www.darpa.mil/program/darpa-robotics-challenge. Last accessed in December 2024.

[90] Sangli Teng, Mark Wilfried Mueller, and Koushil Sreenath. Legged robot state estimation in slippery environments using invariant extended Kalman filter with velocity update. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3104–3110, 2021. DOI: 10.1109/ICRA48506.2021.9561313.

[91] Hans Kumar, J. Joe Payne, Matthew Travers, Aaron M. Johnson, and Howie Choset. Periodic SLAM: Using cyclic constraints to improve the performance of visual-inertial SLAM on legged robots. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9477–9483, 2022. DOI: 10.1109/ICRA46639.2022.9811634.

[92] Shuo Yang, Zixin Zhang, Zhengyu Fu, and Zachary Manchester. Cerberus: Low-drift visual-inertial-leg odometry for agile locomotion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4193–4199, 2023. DOI: 10.1109/ICRA48891.2023.10160486.

[93] Guangjun Ou, Dong Li, and Hanmin Li. Leg-KILO: Robust kinematic-inertial-LiDAR odometry for dynamic legged robots. *IEEE Robotics and Automation Letters*, 9(10):8194–8201, 2024. DOI: 10.1109/LRA.2024.3440730.

[94] Kamak Ebadi, Lukas Bernreiter, Harel Biggie, Gavin Catt, Yun Chang, Arghya Chatterjee, Christopher E. Denniston, Simon-Pierre Deschênes, Kyle Harlow, Shehryar Khattak, Lucas Nogueira, Matteo Palieri, Pavel Petráček, Matěj Petrlík, Andrzej Reinke, Vít Krátký, Shibo Zhao, Ali-akbar Agha-mohammadi, Kostas Alexis, Christoffer Heckman, Kasra Khosoussi, Navinda Kottege, Benjamin Morrell, Marco Hutter, Fred Pauling, François Pomerleau, Martin Saska, Sebastian Scherer, Roland Siegwart, Jason L. Williams, and Luca Carlone. Present and

future of slam in extreme environments: The darpa subt challenge. *IEEE Transactions on Robotics*, 40:936–959, 2024. doi: 10.1109/TRO.2023.3323938.

[95] Jaejun Park, Do Hun Kong, and Hae-Won Park. Design of anti-skid foot with passive slip detection mechanism for conditional utilization of heterogeneous foot pads. *IEEE Robotics and Automation Letters*, 4(2):1170–1177, 2019. DOI: 10.1109/LRA.2019.2895888.

[96] Taiyu Okatani and Isao Shimoyama. Evaluation of ground slipperiness during collision using MEMS local slip sensor. In *2019 IEEE 32nd International Conference on Micro Electro Mechanical Systems (MEMS)*, pages 823–825, Seoul, Korea (South), January 2019. DOI: 10.1109/MEMSYS.2019.8870701.

[97] Yerkebulan Massalim, Zhanat Kappassov, Atakan Varol, and Vincent Hayward. Robust detection of absence of slip in robot hands and feet. *IEEE Sensors Journal*, 21(24):27897–27904, November 2021. DOI: 10.1109/JSEN.2021.3127501.

[98] Hiroshi Takemura, Masato Deguchi, Jun Ueda, Yoshio Matsumoto, and Tsukasa Ogasawara. Slip-adaptive walk of quadruped robot. *Robotics and Autonomous Systems*, 53(2):124–141, November 2005. DOI: 10.1016/j.robot.2005.07.002.

[99] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot hrp-2. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 2, pages 1083–1090 Vol.2, 2004. DOI: 10.1109/ROBOT.2004.1307969.

[100] K. Kaneko, F. Kanehiro, S. Kajita, M. Morisawa, K. Fujiwara, K. Harada, and H. Hirukawa. Slip observer for walking on a low friction floor. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 634–640, 2005. DOI: 10.1109/IROS.2005.1545184.

[101] Michele Focchi, Victor Barasuol, Marco Frigerio, Darwin G Caldwell, and Claudio Semini. Slip detection and recovery for quadruped robots. In *Robotics Research*, pages 185–199. Springer, January 2018. DOI: 10.1007/978-3-319-60916-4_11.

[102] KVH Industries. 1750 IMU, Fiber Optic Gyro Inertial Measurement Unit, 2019. URL https://canalgeomatics.com/wp-content/uploads/2019/11/kvh-1750-imu-datasheet.pdf. Last accessed in December 2024.

[103] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model-predictive control. *IEEE Transactions on Robotics*, 39(5):3402–3421, 2023. DOI: 10.1109/TRO.2023.3275384.

[104] Intel RealSense. evo: Python package for the evaluation of odometry and SLAM, 2019. URL https://www.intelrealsense.com/visual-inertial-tracking-case-study/. Last accessed in December 2024.

[105] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. DOI: 10.1115/1.3662552.

[106] D Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, 2006. DOI: 10.1002/0470045345.

[107] Robert Mahony, T. Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *Automatic Control, IEEE Transactions on*, 53:1203 – 1218, 07 2008. DOI: 10.1109/TAC.2008.923738.

[108] Tor A. Johansen and Thor I. Fossen. The eXogenous Kalman filter (XKF). *International Journal of Control*, 90(2):161–167, 2017. DOI: 10.1080/00207179.2016.1172390.

[109] Leonard A McGee and Stanley F Schmidt. Discovery of the Kalman filter as a practical tool for aerospace and industry. Technical report, 1985.

[110] Eric A Wan and Rudolph Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[111] David Salmond and Neil J. Gordon. An introduction to particle filters. 2006. URL https://api.semanticscholar.org/CorpusID:777437.

[112] Robert Mahony, Jochen Trumpf, and Tarek Hamel. Observers for kinematic systems with symmetry. *IFAC Proceedings Volumes*, 46(23):617–633, 2013. DOI: 10.3182/20130904-3-FR-2041.00212.

[113] David Wisth, Marco Camurri, Sandipan Das, and Maurice Fallon. Unified multi-modal landmark tracking for tightly coupled lidar-visual-inertial odometry. *IEEE Robotics and Automation Letters*, 6(2):1004–1011, 2021. DOI: 10.1109/LRA.2021.3056380.

[114] F Landis Markley and John L Crassidis. *Fundamentals of spacecraft attitude determination and control*, volume 1286. Springer, 2014. DOI: 10.1007/978-1-4939-0802-8.

[115] Håvard Fjær Grip, Thor I. Fossen, Tor A. Johansen, and Ali Saberi. Globally exponentially stable attitude and gyro bias estimation with application to GNSS/INS integration. *Automatica*, 51:158–166, 2015. ISSN 0005-1098. DOI: 10.1016/j.automatica.2014.10.076.

[116] Michael Grupp. evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo, 2017.

[117] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251, 2018. doi: 10.1109/IROS.2018.8593941. DOI: 10.1109/IROS.2018.8593941.

[118] Jan Bayer and Jan Faigl. On autonomous spatial exploration with small hexapod walking robot using tracking camera intel realsense T265. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2019. doi: 10.1109/ECMR.2019.8870968. DOI: 10.1109/ECMR.2019.8870968.

[119] Lorenzo Amatucci, Giulio Turrisi, Angelo Bratta, Victor Barasuol, and Claudio Semini. Accelerating model predictive control for legged robots through distributed optimization. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[120] Douglas Wildrube Bertol, Geoff Fink, Ylenia Nisticò, Gianluca Cerilli, Marco Marchitto, and Claudio Semini. A practical real-time distributed software framework for mobile robots. *Second Workshop on Quality and Reliability Assessment of Robotic Software Architectures and Components, ICRA 2023 Workshop*, 2023.

[121] Shuo Yang, Zixin Zhang, Zhengyu Fu, and Zachary Manchester. Cerberus: Low-drift visual-inertial-leg odometry for agile locomotion. In *2023 IEEE International*

Conference on Robotics and Automation (ICRA), pages 4193–4199, 2023. DOI: 10.1109/ICRA48891.2023.10160486.

[122] Axel Barrau and Silvere Bonnabel. Invariant kalman filtering. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):237–257, 2018. DOI: 10.1146/annurev-control-060117-105010.

[123] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[124] Axel Barrau and Silvère Bonnabel. The invariant extended kalman filter as a stable observer. *IEEE Transactions on Automatic Control*, 62(4):1797–1812, 2017. DOI: 10.1109/TAC.2016.2594085.

[125] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

[126] Tzu-Yuan Lin, Ray Zhang, Justin Yu, and Maani Ghaffari. Legged robot state estimation using invariant kalman filtering and learned contact events. *arXiv preprint arXiv:2106.15713*, 2021.

[127] Sangli Teng, Mark Wilfried Mueller, and Koushil Sreenath. Legged robot state estimation in slippery environments using invariant extended kalman filter with velocity update. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3104–3110, 2021. DOI: 10.1109/ICRA48506.2021.9561313.

[128] Yuan Gao, Chengzhi Yuan, and Yan Gu. Invariant filtering for legged humanoid locomotion on a dynamic rigid surface. *IEEE/ASME Transactions on Mechatronics*, 27(4):1900–1909, 2022. DOI: 10.1109/TMECH.2022.3176015.

[129] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008. ISBN 978-0-691-13298-3.

[130] Paul Chauchat, Axel Barrau, and Silvere Bonnabel. Invariant smoothing on lie groups. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1703–1710, 2018. DOI: 10.1109/IROS.2018.8594068.

[131] Alex Walsh, Jonathan Arsenault, and James Richard Forbes. Invariant sliding window filtering for attitude and bias estimation. In *2019 American Control Conference (ACC)*, pages 3161–3166, 2019. DOI: 10.23919/ACC.2019.8814702.

[132] Axel Barrau. *Non-linear state error based extended Kalman filters with applications to navigation*. Theses, Ecole Nationale Supérieure des Mines de Paris, September 2015. URL https://pastel.hal.science/tel-01344622.

[133] Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[134] Gregory S Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 1: Classical Results and Geometric Methods*. Springer Science & Business Media, 2009.

[135] Gregory S Chirikjian. *Stochastic models, information theory, and Lie groups, volume 2: Analytic methods and modern applications*, volume 2. Springer Science & Business Media, 2011.

[136] Ethan Eade. Lie groups for 2d and 3d transformations. *URL http://ethaneade.com/lie. pdf, revised Dec*, 117:118, 2013.

[137] PS Maybeck. Stochastic models, estimation, and control, 1982.

[138] Holybro. Holybro, 2024. URL https://holybro.com/. Last accessed in December 2024.

[139] Livox. Livox Mid-360 LiDAR, 2024. URL https://www.livoxtech.com/mid-360. Last accessed in December 2024.

[140] Google. Google Earth, 2024. URL https://www.google.it/earth/. Last accessed in December 2024.

[141] Ouster. Velodyne VLP-16 LiDAR, 2024. URL https://ouster.com/products/hardware/vlp-16. Last accessed in December 2024.

[142] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3.2, page 5. Kobe, Japan, 2009.

[143] Joan Sola, Joan Vallvé, Joaquim Casals, Jérémie Deray, Médéric Fourmy, Dinesh Atchuthan, Andreu Corominas-Murtra, and Juan Andrade-Cetto. WOLF: A modular estimation framework for robotics based on factor graphs. *IEEE Robotics and Automation Letters*, 7(2):4710–4717, 2022. DOI: 10.1109/LRA.2022.3151404.

[144] Object Management Group. Data Distribution Service (DDS), 2024. URL https://www.dds-foundation.org/what-is-dds-3/. Last accessed in December 2024.

[145] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022. DOI: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

A

Appendix

Modern robotics systems require robust and efficient software frameworks to handle the complexities of real-time operations and distributed architectures. While widely adopted solutions like the Robot Operating System (ROS) (Quigley et al. [142]) provide a generic foundation, their limitations in scenarios demanding high precision, resource efficiency, and specific communication protocols highlight the need for specialized frameworks.

This appendix introduces DLS2, a novel real-time distributed software framework, designed specifically for mobile robots, tested on quadrupedal platforms, and implemented in C++. DLS2 has been employed to manage real-time communication between the robot's sensors and the MUSE state estimator presented in Chapter 4. Developed to overcome the challenges of existing solutions such as the one proposed in (Sola et al. [143]), which lacked the ability to execute each processor in its own thread, the DLS2 framework emphasizes modularity, containerization, real-time scheduling, and distributed operations. Its architecture employs a layered design, ensuring that components with distinct requirements — such as control modules and user interfaces — are logically grouped and independently managed.

Furthermore, the framework's use of Data Distribution Service (DDS)-based communication (Management Group [144]) ensures seamless integration with other robotic systems, such as ROS2 (Macenski et al. [145]), while maintaining flexibility and performance. This integration facilitates the deployment of state-of-the-art algorithms in mobile robotics, paving the way for robust, real-time, and distributed robotic solutions.

The DLS2 framework has been introduced in the following publication:

> Douglas Wildgrube Bertol, Geoff Fink, **Ylenia Nisticò**, Gianluca Cerilli, Marco Marchitto, and Claudio Semini, "*A Practical Real-Time Distributed Software Framework for Mobile Robots*", Second Workshop on Quality and Reliability Assessment of Robotic Software Architectures and Components, *ICRA 2023 Workshop*.

This work was conceptualized by Douglas Wildgrube Bertol and Geoff Fink. Software development was carried out by Douglas Wildgrube Bertol, Geoff Fink, Gianluca Cerilli, and Marco Marchitto. I contributed by developing and testing MUSE integrated into the software, as well as conducting experiments to test the developed tools. The initial draft was prepared by Douglas Wildgrube Bertol and Geoff Fink, with all authors contributing to the review and editing process.

In the sections that follow, the software's architecture, key features, and operational principles are detailed, highlighting its contributions to the field of robotic software engineering.

A.1 Software Architecture

The DLS2 framework is built on a modular, layered architecture designed to cater to the specific requirements of mobile robotic systems. Each layer in the architecture is dedicated to managing a group of modules with shared characteristics, ensuring that system components are logically organized and efficiently supervised. The layered architecture supports distributed deployment, enabling components to operate across multiple computational nodes. This flexibility ensures resource optimization and system robustness, as processes can be dynamically relocated to balance the load or maintain redundancy. A schematic of the DLS2 software architecture is presented in Fig. A.1. Key layers include:
Control Layer: This layer handles modules requiring real-time scheduling and low-level interface access. The Control Layer governs resource-intensive processes such as controllers, ensuring they operate with minimal latency and fail-safe constraints.
Estimation Layer: The Estimation layer manages signal gathering, state estimation, and processing modules. It ensures robust and accurate data integration from multiple sources, crucial for reliable decision-making. This is the layer where the MUSE state estimator is integrated.
Service Layer: The Service Layer provides middleware services like communication man-

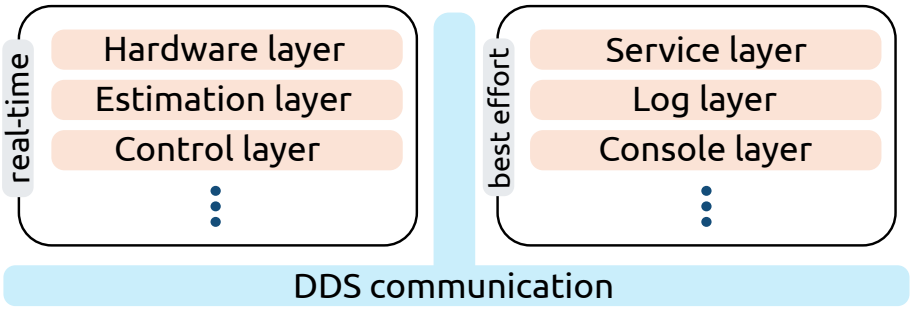


Figure A.1: Schematic of the DLS2 Software Architecture

agement and resource allocation. It ensures seamless interactions between distributed components.

Logging and Console Layers: These two layers are responsible for non-critical operations like system logging and user interface management. These layers focus on usability and diagnostics without impacting performance-critical processes.

A.2 Key Features and Operational Principles

The operational framework of DLS2 is guided by its commitment to modularity, real-time performance, and distributed resource management. Below is a summary of how these principles translate into its functioning: **Real-time scheduling** – The framework provides real-time scheduling capabilities, ensuring deterministic behavior for control and estimation processes. This feature is essential for mobile robotic systems, where precise timing is critical for stability and performance. The DLS2 framework provides direct access to the low-level kernel Application Programming Interface (API) to implement precise timing mechanisms. Furthermore, DLS2 employs DDS to facilitate low-latency, reliable communication between distributed processes. This communication offers high-performance, real-time data sharing with Quality of Service (QoS) guarantees. DLS2 remains compatible with ROS2, enabling interoperability with existing robotic software ecosystems. **Containerization** – The framework employs containerization to isolate modules, avoiding compatibility issues and enabling fast deployment. This design simplifies system updates and ensures reproducibility in diverse hardware environments. **Distributed Architecture** – DLS2 supports true distribution, where processes can operate on separate nodes while maintaining synchronized communication. Modules can be duplicated or migrated dynamically across computational nodes to enhance fault tolerance. Additionally, by isolating modules in containers, researchers can experiment with cutting-edge techniques without disrupting the overall system. **Modular and Resource-Efficient Design** – System components are designed as self-contained

modules that can be reused, updated, or replaced independently. This ensures that only relevant components are deployed for specific applications. The library-oriented framework allows users to deploy only the required modules, reducing computational overhead and resource consumption. The modularity also simplifies the integration of custom algorithms and tools.

A.3 Integration of MUSE into DLS2 Framework

The MUSE state estimator described in Chapter 4 is seamlessly integrated into the DLS2 framework as a collection of modular plugins. This integration leverages DLS2's containerized and distributed architecture, allowing real-time, efficient communication between MUSE modules and other robotic systems.

A.3.1 Plugin structure in DLS2

Each module of MUSE is implemented as a plugin within DLS2, adhering to a standardized structure for consistency and ease of integration. The plugin structure includes the following components:

1. `reader_inputs()`: Retrieves the required data inputs, such as sensor readings or intermediate outputs from other modules.
2. `run()`: Executes the core computational logic of the module, where algorithms specific to the module are implemented.
3. `publish_outputs()`: Outputs the results of the computation, making them available to other modules or systems.

A.3.2 Modules in MUSE

The MUSE estimator comprises both low-level and high-level estimation modules, each targeting specific aspects of state estimation:

A.3.2.1 Low-level Estimation Modules

- **Contact Detection**: This module takes joint state readings from the joint encoders and outputs the contact status for each leg. The purpose is to identify which legs are in contact with the ground, crucial for state estimation, stability, and locomotion control.

- **Attitude Estimation**: This module receives inputs from the IMU accelerometer and gyroscope, and outputs the robot orientation and angular velocity. This module is necessary to determine the robot's attitude providing fundamental data for locomotion and navigation.
- **Leg Odometry**: This module estimates the robot's movement based on leg motion and sensor readings. It receives inputs from the contact detection and attitude estimation module, from the joint encoders and from the IMU.
- **Slip Detection**: This module detects slip events by comparing the expected leg motion with the actual motion. It receives inputs from the contact detection module and the joint states.

A.3.2.2 High-level Estimation Modules

- **Camera Odometry**: This module uses visual data to estimate the robot's position in space, taking camera images and giving the pose of the robot as output.
- **LiDAR Odometry**: This module utilizes 3D spatial data to refine positional estimates, taking LiDAR scans as input and providing the robot's pose as output.
- **Sensor Fusion**: This is the last module, where data from multiple sources are combined to produce a robust and accurate estimate of the robot's state. In particular, it reads inputs from contact detection, attitude estimation, and leg odometry, (optionally also from slip detection, camera odometry, and LiDAR odometry, depending on the user's configuration) and outputs the robot's pose and velocity.

Each plugin operates independently and can be activated or deactivated dynamically during experiments. This flexibility allows to customize the system based on the task or available sensors. For instance, camera odometry can be turned off in low-light environments, relying instead on LiDAR and low-level estimation modules for sensor fusion. The modularity allows easy addition or removal of plugins, facilitating experimentation with new algorithms or sensors without impacting the rest of the system. Online activation or deactivation of modules provides the ability to adapt to changing experimental conditions or resource availability. An example of this is shown in Fig. A.2.

Additionally, the modular structure ensures efficient execution, with the average runtime of each plugin's `run()` function measured at 0.05 ms on average (this is also due to the choice of using filtering methods, as already explained in Section 4.6.8). This speed enables the system to provide real-time feedback to the robot's locomotion controller,

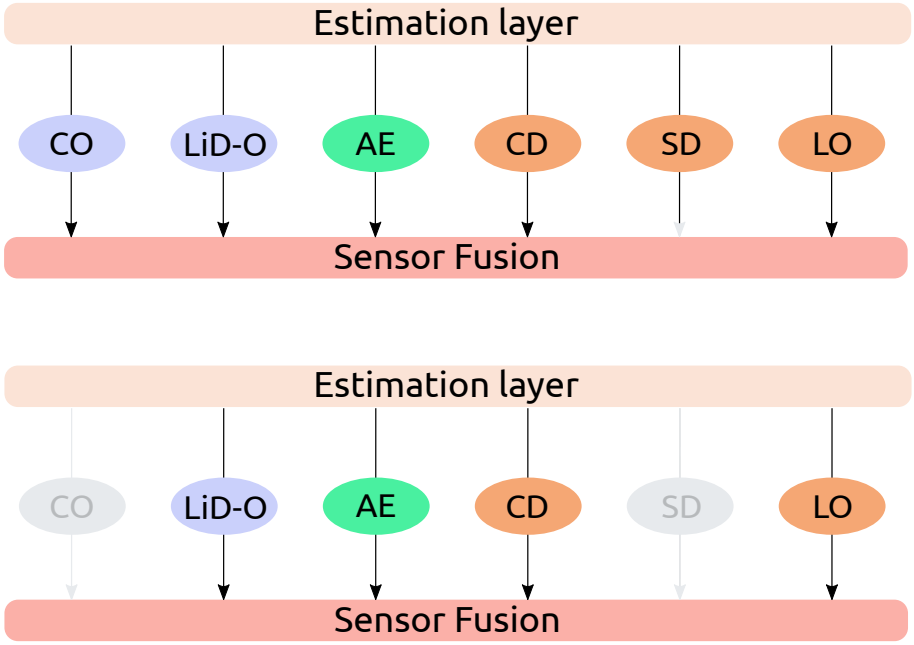


Figure A.2: Dynamic Activation and Deactivation of Modules in DLS2. In the upper figure, all modules are active, while in the lower figure, the camera odometry module and the slip detection module are deactivated. This procedure can be executed online, allowing the system to adapt to changing conditions.

ensuring smooth and responsive operation. By isolating computational tasks into plug-ins, the system ensures that resource-intensive processes do not interfere with real-time control requirements.

A.4 Conclusion

The DLS2 framework represents an advancement in robotic software engineering, offering a real-time, distributed solution tailored to the needs of mobile robots. By emphasizing modularity, real-time scheduling, and distributed operations, DLS2 provides a robust software foundation for state-of-the-art algorithms and tools. The integration of the MUSE state estimator into the framework demonstrates the flexibility and efficiency of the system, enabling seamless communication between modules and other robotic systems. The framework's layered architecture, containerization, and distributed operations ensure that mobile robots can operate with precision, reliability, and performance, paving the way for future advancements in robotic software engineering.

Furthermore, the integration of the MUSE state estimator into the DLS2 framework exemplifies the power of modular, real-time software architecture in mobile robotics. By treating each estimation module as an independent plugin with standardized operations, reading inputs, running computations, and publishing outputs, the framework ensures

flexibility, efficiency, and robust performance.

This modular structure not only enables seamless communication and collaboration between low-level and high-level estimation tasks but also provides the ability to dynamically adjust the active components based on experimental needs or environmental constraints. The integration of advanced modules such as camera and LiDAR odometry, alongside foundational estimators such as contact, attitude, and slip detection, highlights the adaptability of the framework to accommodate diverse sensing technologies.

The measured efficiency of plugin execution, averaging 0.05 ms per module, confirms the suitability of DLS2 for real-time feedback applications. This performance ensures reliable communication with the locomotion controller, maintaining the stability and responsiveness essential for dynamic mobile robots.

In summary, the combination of DLS2's distributed architecture and MUSE's modular design demonstrates a scalable and high-performing solution for real-time robotic state estimation. This framework lays a strong foundation for future innovations in state estimation and robot control, empowering researchers and engineers to deploy cutting-edge algorithms with ease and confidence.

B

Publications

B.1 List of Publications

Ylenia Nisticò, Shamel Fahmi, Lucia Pallottino, Claudio Semini, and Geoff Fink, "*On slip detection for quadruped robots*". In MDPI Sensors, 22(8), 2967, 2022. DOI: 10.3390/s22082967

Douglas Wildgrube Bertol, Geoff Fink, **Ylenia Nisticò**, Gianluca Cerilli, Marco Marchitto, and Claudio Semini, "*A Practical real-Time Distributed Software Framework for Mobile Robots*". In Second Workshop on Quality and Reliability Assessment of Robotic Software Architectures and Components, IEEE/RSJ International Conference on Robotics and Automation (ICRA) Workshop, 29 May–2 June 2023.

Hilton Marques Souza Santana, João Carlos Virgolino Soares, **Ylenia Nisticò**, Marco Antonio Meggiolaro, Claudio Semini, "*Proprioceptive State Estimation for Quadruped Robots using Invariant Kalman Filtering and Scale-Variant Robust Cost Functions*". In IEEE-RAS International Conference on Humanoid Robots 2024 (Humanoids 2024), November 22–24, 2024. DOI: 10.48550/arXiv.2410.05256

Ylenia Nisticò, João Carlos Virgolino Soares, Geoff Fink, and Claudio Semini, "*Multi-Sensor Fusion for Quadruped Robot State Estimation on Challenging Terrain*". In 2024

I-RIM 3D Conference, 6th Edition (I-RIM 2024), October 25–27, 2024.

Ylenia Nisticò, João Carlos Virgolino Soares, Lorenzo Amatucci, Geoff Fink, and Claudio Semini, "*MUSE: A Real-Time Multi-Sensor State Estimator for Quadruped Robots*". Under review at IEEE Robotics and Automation Letters (RA-L).

Ylenia Nisticò*, Hajun Kim*, João Carlos Virgolino Soares, Geoff Fink, Hae-Won Park, and Claudio Semini, "*Multi-Sensor Fusion for Quadruped Robot State Estimation using Invariant Filtering and Smoothing*". Under review at IEEE Robotics and Automation Letters (RA-L). * Equal contribution.