UNIVERSITÄT BONN

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Agrar-, Ernährungs- und Ingenieurwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn
Institut für Geodäsie und Geoinformation

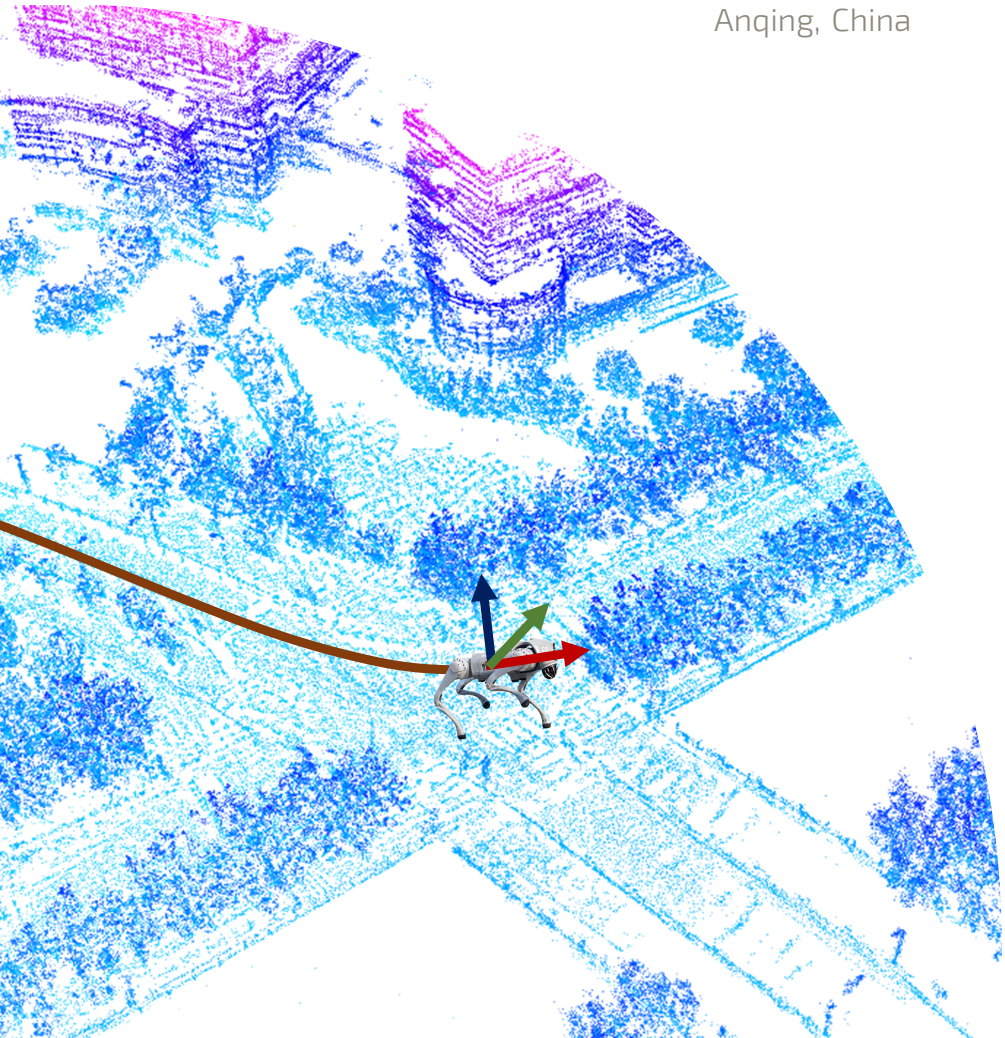
Mobile Robot State Estimation Based on Aided Inertial Navigation

von

Yibin Wu

aus

Anqing, China



Bonn 2026

Referent:
Prof. Dr. Heiner Kuhlmann, University of Bonn, Germany

1. Korreferent:
Prof. Dr. Cyrill Stachniss, University of Bonn, Germany

2. Korreferent:
Prof. Dr. Maurice Fallon, University of Oxford, United Kingdom

Tag der mündlichen Prüfung: 23.01.2026

Angefertigt mit Genehmigung der Agrar-, Ernährungs- und Ingenieurwissenschaftlichen Fakultät der Universität Bonn

Zusammenfassung

Robuste und genaue Zustandsschätzung ist ein fundamentales Element, das es mobilen Robotern ermöglicht, autonom zu agieren. Um dieses Ziel zu erreichen, sind moderne mobile Roboter typischerweise mit einer Vielzahl von Sensoren ausgestattet, wie Kameras, Light Detection and Ranging Sensoren (LiDARs), GNSS-Empfängern (Global Navigation Satellite System) und Inertialmesseinheiten (IMUs), da diese Sensoren komplementäre Informationen über die Umgebung und die Bewegung des Roboters liefern. Unter all diesen Sensoren spielen IMUs eine zentrale Rolle, da sie unabhängig von der Umgebung arbeiten können. Zusätzlich liefern IMUs hochfrequente Messungen der Bewegung in sechs Freiheitsgraden (6-DoF) und sind kostengünstig. Allerdings neigen IMUs aufgrund von inhärentem Rauschen und Bias-Instabilitäten zu einem erheblichen Fehlerdrift über die Zeit.

Um den Drift von IMUs zu mindern, wurden verschiedene Unterstützungssensoren mit inertialen Navigationssystemen (INS) integriert. Beispielsweise liefern GNSS absolute Positionsdaten, Odometer (z.B. Radsensoren) bieten Geschwindigkeitsmessungen, und Kameras sowie LiDARs liefern räumliche Geometrieinformationen. Allerdings sind diese Unterstützungssensoren nicht immer zuverlässig: GNSS verschlechtert sich in Straßenschluchten und Wäldern, Odometer reagieren empfindlich auf Radschlupf und Terrainänderungen, und Kameras sowie LiDARs sind von Lichtverhältnissen und Wetterbedingungen abhängig. Neben diesen externen Sensoren können auch die Bewegungsprofile verschiedener mobiler Roboter genutzt werden, um IMUs auf eine selbstständige Weise zu unterstützen und den Fehlerdrift zu reduzieren, wenn andere Sensoren ausfallen. Solche Bewegungsrestriktionen umfassen unter anderem die nicht-holonomen Zwangsbedingungen (NHC) von Radrobotern und die Vorwärtskinematikmodelle von Laufrobotern.

Zusammenfassend ist es entscheidend, alle verfügbaren Informationen aus verschiedenen Quellen zu fusionieren, um eine robuste und genaue Zustandsschätzung zu erreichen, die auf spezifische Anwendungen zugeschnitten ist. Einerseits müssen fortschrittliche Sensordatenfusionsalgorithmen entwickelt werden, um die komplementären Vorteile der verschiedenen Sensoren vollständig zu nutzen und

die Gesamtleistung zu verbessern. Andererseits ist die Verbesserung der IMU-basierten propriozeptiven Zustandsschätzung entscheidend, um eine kontinuierliche und zuverlässige Pose-Schätzung sicherzustellen, wenn andere Sensoren aufgrund von Umwelteinflüssen ausfallen.

Der Hauptbeitrag dieser Arbeit ist die Entwicklung neuartiger Ansätze, die verschiedene Unterstützungsquellen ausnutzen, um die Genauigkeit, Robustheit und Effizienz der kostengünstigen IMU-basierten Zustandsschätzung zu verbessern. Dies umfasst sowohl Verbesserungen der IMU-basierten propriozeptiven Odometrie als auch die Fusion von IMUs mit anderen exterozeptiven Sensoren. Die Arbeit gliedert sich in drei Hauptteile:

Der erste Teil konzentriert sich auf die Zustandsschätzung für Radroboter. Insbesondere präsentieren wir neuartige Algorithmen unter Verwendung einer am Rad montierten IMU (Wheel-IMU), um die kontinuierliche Rotation des Rades auszunutzen. Wir beginnen mit einem propriozeptiven Odometriesystem, das ausschließlich eine Wheel-IMU verwendet und automatisch den konstanten Biasfehler der IMU durch die Radrotation mindert. Anschließend erweitern wir die vorgeschlagene Wheel-IMU-Odometrie zu einem SLAM-System, indem wir die vom Wheel-IMU geschätzten Terrainmerkmale (Querneigung der Straße) extrahieren, um eine Schleifenschlusserkennung zu ermöglichen. In diesem speziellen Entwurf fungiert die IMU nicht mehr nur als Eigenbewegungssensor, sondern auch als exterozeptiver Sensor, der Umweltmerkmale erfassen kann. Um die langfristige Genauigkeit der Zustandsschätzung in großräumigen Umgebungen weiter zu verbessern, integrieren wir GNSS mit der Wheel-IMU, wobei die komplementären Vorteile der kurzzeitigen zuverlässigen Odometrie der Wheel-IMU und der absoluten Positionsinformation von GNSS genutzt werden. Nach unserem besten Wissen sind wir die Ersten, die ein SLAM-System ausschließlich mit einem Wheel-IMU vorschlagen, sowie die Ersten, die GNSS mit einem Wheel-IMU zur mobilen Roboterlokalisierung integrieren.

Der zweite Teil dieser Arbeit konzentriert sich auf die propriozeptive Zustandsschätzung von Laufrobotern. Inspiriert von den Erkenntnissen aus unseren vorherigen Arbeiten zum Wheel-IMU schlagen wir vor, mehrere IMUs an den Beinen (Leg-IMUs) des Laufroboters zu platzieren, um die 6-DOF-Bewegungsschätzfähigkeit der IMUs an verschiedenen Positionen des Roboters zu nutzen. Darüber hinaus verwenden wir das Vorwärtskinematikmodell des Laufroboters, um die Fehlerdrift der mehreren IMUs zu begrenzen.

Der dritte Teil dieser Arbeit behandelt ein generisches inertiales Navigationssystem mit Unterstützung, das nicht auf Bewegungsprofile von Rad- oder Laufrobotern beschränkt ist. In diesem Kontext schlagen wir ein LiDAR-inertiales Odo-

metriesystem vor, das präzise 3D-Geometrieinformationen von LiDAR-Sensoren nutzt, um die IMU zu unterstützen. Insbesondere entwickeln wir ein eng gekoppeltes LiDAR-inertiales Odometriesystem basierend auf einer Punkt-zu-Punkt-Matching und dem klassischen extended Kalman filter (EKF) für hohe Effizienz. Darüber hinaus schlagen wir ein adaptives Datenassoziationsmodell vor, um die Anzahl der einzustellenden Parameter je nach Umgebungstyp zu reduzieren.

Alle in dieser Arbeit vorgestellten Ansätze wurden in begutachteten Konferenzbeiträgen und Fachzeitschriftenartikeln veröffentlicht. Unser vorgeschlagenes LiDAR-inertiales Odometriesystem (LIO-EKF) belegte den zweiten Platz in der LiDAR-Inertial-Track-Challenge der International Conference on Computer Vision 2023. Zusätzlich haben wir Implementierungen unserer in dieser Arbeit vorgestellten Methoden als Open Source bereitgestellt, um die Forschungsgemeinschaft zu unterstützen und weitere Forschung zu erleichtern.

Abstract

Robust and accurate state estimation is a fundamental building block that enables mobile robots to achieve autonomy. To achieve this goal, modern mobile robots are typically equipped with a variety of sensors, such as cameras, light detection and ranging sensors (LiDARs), Global Navigation Satellite System (GNSS) receivers, and inertial measurement units (IMUs), because these sensors provide complementary information about the environment and the robot's egomotion. Among all these sensors, IMUs play a central role as they can work independently without being affected by the environments. Additionally, IMUs can provide high-frequency 6 degree-of-freedom (DoF) motion measurements and are low-cost. However, IMUs are prone to inherent noise and bias instability, resulting in significant error drift over time.

To mitigate IMU drift, various aiding sources have been integrated with inertial navigation systems (INS). For example, GNSS provides absolute position data; odometers (i.e., wheel encoders) offer velocity measurements; and cameras and LiDARs provide spatial geometry constraints. However, these aiding sensors are not always reliable: GNSS deteriorates in urban canyons and forests; odometers are sensitive to wheel slip and terrain variations; and cameras and LiDARs are affected by lighting and weather conditions. In addition to these external sensors, the motion profiles of different mobile robots can also be exploited to aid IMUs in a self-contained manner and reduce error drift when other sensors fail. Such motion constraints include the non-holonomic constraints (NHC) of wheeled robots and the forward kinematic models of legged robots.

In conclusion, it is crucial to fuse all available information from different sources to achieve robust and accurate state estimation. On one hand, advanced sensor fusion algorithms must be developed to fully take the complementary advantages of IMU and other sensors. On the other hand, enhancing IMU-based proprioceptive state estimation is essential to ensure continuous and reliable pose estimation when other sensors fail due to environmental disturbances.

The main contribution of this thesis is the development of novel approaches that exploits different aids to improve the accuracy, robustness, and efficiency of low-cost IMU-based state estimation. These contributions include not only improvements to IMU-based proprioceptive odometry but also the fusion of IMUs

with other exteroceptive sensors. Emphasis is placed on the use of low-cost inertial sensors, with the goal of reducing overall hardware costs and thereby enabling large-scale deployment. The thesis is structured into three main parts.

The first part focuses on the state estimation for wheeled robots. Specifically, we present novel algorithms using a wheel-mounted IMU (Wheel-IMU) to take advantage of the continuous rotation of the wheel. We begin with a proprioceptive odometry estimation system (Wheel-INS) using only a Wheel-IMU, which automatically mitigates the constant bias error of the IMU with the wheel's rotation. We then extend the proposed Wheel-INS to a SLAM system by extracting the terrain features (road bank angles) estimated by the Wheel-IMU to enable loop closure detection. In this special design, the IMU is no longer merely an egomotion sensing device; it also functions as an exteroceptive sensor, capable of perceiving environmental features. To further improve long-term state estimation accuracy of the sensor fusion system in large-scale environments, we integrate GNSS with the Wheel-IMU, leveraging the complementary advantage of the short-term reliable odometry form the Wheel-IMU and the absolute position information from GNSS. To the best of our knowledge, we are the first to propose a SLAM system using only a Wheel-IMU, as well as the first to integrate GNSS with a Wheel-IMU for mobile robot localization.

The second part focuses on the proprioceptive state estimation of legged robots. Inspired by insights from our previous works on Wheel-IMU, we propose to place multiple IMUs on the legs (Leg-IMUs) of the legged robot to exploit the 6 DOF motion estimation capability of the IMUs at different locations of the robot. We further employ the forward kinematic model of the legged robot to constrain the error drift of the multiple IMUs.

The third part focuses on a generic aided inertial navigation system that is not constrained by the motion profiles of wheeled or legged robots. In this context, we propose a LiDAR-inertial odometry system that leverages accurate 3D geometric information from LiDARs to aid the IMU. Specifically, we propose a tightly-coupled LiDAR-inertial odometry system based on point-to-point matching and the classical extended Kalman filter (EKF) scheme for efficiency. In addition, we propose an adaptive data association model to reduce the parameters to tune for a given type of environment.

All our proposed approaches presented in this thesis have been published in peer-reviewed conference papers and journal articles. Our proposed LiDAR-inertial odometry system (LIO-EKF) received the second place award in the LiDAR-Inertial Track of the 2023 International Conference on Computer Vision SLAM Challenge. Additionally, we have made implementations of our methods presented in this thesis open-source to benefit the community.

Acknowledgements

It has been a wonderful and unique journey in my life to pursue my Ph.D. at the Institute of Geodesy and Geoinformation in the University of Bonn. First and foremost, I would like to express my deepest appreciation to my supervisor, Prof. Heiner Kuhlmann. I am sincerely grateful to him for accepting me as one of his Ph.D. students and for providing an excellent research environment and continuous support throughout my doctoral studies. His guidance has fostered an atmosphere that encouraged me to develop both personally and academically, to challenge my limits, and to pursue excellence in all aspects of my work. I also want to thank him for his kind support in making my research exchange program in the UK possible.

I also wish to extend my heartfelt thanks to Dr. Lasse Klingbeil. I truly enjoyed our insightful discussions and greatly appreciate his continuous help and support in both my research and life.

Special thanks go to Prof. Cyrill Stachniss for his generous support and guidance on my research. During our collaboration, I have learned a lot of research skills from him and his group, not only academic writing and presentation skills, but also critical thinking, problem formulation, and how to conduct rigorous and impactful research.

Furthermore, I would like to thank Prof. Maren Bennewitz, Dr. Jens Behley, and Dr. Tiziano Guadagnino for their valuable guidance and collaboration in my projects and publications. I also extend my gratitude to Prof. Maurice Fallon, Prof. Dimitrios Kanoulas for their efforts in making my visiting program possible, as well as their supervision and support during my research exchange at the University of Oxford and University College London. I was lucky to also have Prof. Frank Dellaert involved in my research project in the UK. Many thanks to him for his insightful feedback and support. I would like to extend my appreciation to all the lab members in their groups for their help, encouragement, and the enjoyable after-work activities. I have learned a great deal and truly cherished my time in the UK.

Many thanks to my friends and colleagues — Meltem Cantürk, Markus Wagner, Eike Koller, Lina Stausberg, Christian Gaus, Felix Esser, Manuel Mittelstedt, Gereon Tombrink, André Cornelißen, Annika Tobies, Ansgar Derier, Laura

Zabawa, Berit Jost, Yue Pan, Xingguang Zhong, Liren Jin, and Haofei Kuang, Xieyuanli Chen, Si Yang, and many others, for the inspiring discussions, wonderful events, and unforgettable parties that made my time here both fulfilling and enjoyable. These will remain cherished memories throughout my life. I would like to especially thank Meltem Cantürk for sharing a lot of great time with me in the office and after work, and Gereon Tombrink for his kind help with the German abstract of this thesis.

Many thanks to the administrative staff of the institute, especially Karin Bremer and Bernd Binnenbruck, for their continuous support and assistance in various administrative matters. In addition, I would like to thank all the technical staff, Holger Milz, Dirk Nießitt, Michael Acker, Ralf Becker, Martin Blome, and Michael Plech for their kind help with the hardware and laboratory facilities.

I also wish to express my sincere gratitude to my former supervisors and mentors, Prof. Xiaoji Niu and Prof. Jian Kuang, for shaping my research taste and preparing me well for my doctoral studies. Their continuous guidance and support have been invaluable to me.

Last but not least, I would like to thank my parents, Xuemei Zhao and Lixun Wu, for their unconditional love, endless support, and constant encouragement in all aspects of my life.

The work presented in this thesis is partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob.

Contents

Preface

This dissertation presents the exploration of different aids to enhance the accuracy, robustness, and efficiency of the IMU-based state estimation for mobile robots. It is based on the following publications that are all subject to a peer-review process:

- **Publication A** [76] (Peer-reviewed, Journal):
  X. Niu, Y. Wu, and J. Kuang. Wheel-INS: A wheel-mounted MEMS IMU-based dead reckoning system. *IEEE Transactions on Vehicular Technology*, 70(10):9814–9825, 2021 doi:10.1109/TVT.2021.3108008

- **Publication B** [110] (Peer-reviewed, Journal):
  Y. Wu, J. Kuang, X. Niu, J. Behley, L. Klingbeil, and H. Kuhlmann. Wheel-SLAM: Simultaneous localization and terrain mapping using one wheel-mounted IMU. *IEEE Robotics and Automation Letters*, 8(1):280–287, 2023 doi:10.1109/LRA.2022.3226071

- **Publication C** [111] (Peer-reviewed, Journal):
  Y. Wu, J. Kuang, X. Niu, C. Stachniss, L. Klingbeil, and H. Kuhlmann. Wheel-GINS: A GNSS/INS integrated navigation system with a wheel-mounted imu. *IEEE Transactions on Intelligent Transportation Systems*, 26(5):6891–6903, 2025 doi:10.1109/TITS.2025.3527815

- **Publication D** [108] (Peer-reviewed, Conference):
  Y. Wu, J. Kuang, S. Khorshidi, X. Niu, L. Klingbeil, M. Bennewitz, and H. Kuhlmann. DogLegs: Robust proprioceptive state estimation for legged robots using multiple leg-mounted IMUs. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025 doi:10.1109/IROS60139.2025.11246027

- **Publication E** [107] (Peer-reviewed, Conference):
  Y. Wu, T. Guadagnino, L. Wiesmann, L. Klingbeil, C. Stachniss, and H. Kuhlmann. LIO-EKF: High frequency LiDAR-inertial odometry using extended Kalman filters. In *Proc. of the IEEE International Confer-*

ence on Robotics and Automation, pages 13741–13747, 2024 doi:10.1109/ICRA57147.2024.10610667

The content of these publications is summarized in Chapter 4, Chapter 5 and Chapter 6, respectively. The author of this dissertation has made the main contribution to each of these publications and, in particular, has provided the respective methodology.

Our proposed algorithms presented in this thesis are open-source to benefit the community for facilitating further research. The links to the repositories of our implementations are provided below and correspond to the publications mentioned above.

- Section 4.1 presented our proposed wheel-mounted IMU-based localization system, called Wheel-INS. This implementation is available online at: https://github.com/i2Nav-WHU/Wheel-INS.
- Section 4.2 presented our proposed Wheel-SLAM, a simultaneous localization and terrain mapping system using one wheel-mounted IMU. This implementation is available online at: https://github.com/i2Nav-WHU/Wheel-SLAM.
- Section 4.3 presented our proposed GNSS/Wheel-IMU integrated localization system, called Wheel-GINS. The implementation is available at: https://github.com/i2Nav-WHU/Wheel-GINS.
- Chapter 5 presented our multiple leg-mounted IMUs-based state estimation system for legged robots, called DogLegs. The implementation of the baseline leg odometry is available online at: https://github.com/YibinWu/leg-odometry.
- Chapter 6 presented our tightly-coupled high frequency LiDAR-inertial odometry system, LIO-EKF. The implementation is available at: https://github.com/YibinWu/LIO-EKF.

The following are publications that are not included in this dissertation but are related to the work presented in this dissertation. The author of this dissertation has made the main contribution to each of these publications and, in particular, has provided the respective methodology.

- **Publication F** [112] (Peer-reviewed, Journal):
  Y. Wu, X. Niu, and J. Kuang. A comparison of three measurement models for the wheel-mounted MEMS IMU-based dead reckoning system. *IEEE Transactions on Vehicular Technology*, 70(11):11193–11203, 2021 doi:10.1109/TVT.2021.3102409

- **Publication G** [109] (Peer-reviewed, Journal):
  Y. Wu, J. Kuang, and X. Niu. Wheel-INS2: Multiple MEMS IMU-based dead reckoning system with different configurations for wheeled robots. *IEEE Transactions on Intelligent Transportation Systems*, 24(3):3064–3077, 2023 doi:10.1109/TITS.2022.3220508

Chapter 1

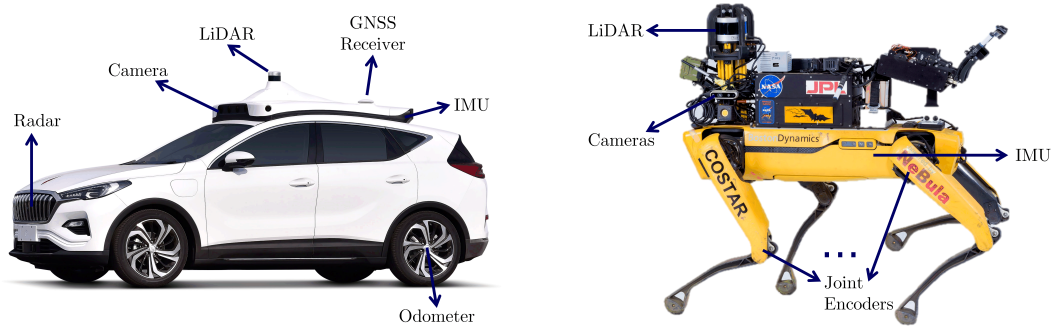
Introduction

1.1 Motivation

Mobile robots with full autonomy hold great promise for improving human life across a wide range of domains. In disaster scenarios, rescue robots can navigate hazardous environments to search for survivors. In agriculture, autonomous robots monitor crop growth and optimize farming practices. Service robots assist in offices, restaurants, and hospitals, enhancing efficiency and user experience. Meanwhile, self-driving cars reduce traffic congestion and improve road safety, transforming the future of transportation. To enable these capabilities, accurate, robust, and efficient state estimation is essential for mobile robots. It is a core component of autonomous navigation, as it answers the critical question: "Where am I?" Generally, effective state estimation systems must meet three key requirements: accuracy, robustness, and efficiency. **Accuracy** refers to the ability to estimate the robot's pose closely aligned with its true state [54, 129]. **Robustness** indicates the ability to maintain reliable pose estimation under diverse conditions and disturbances [7]. **Efficiency** denotes the capability to provide pose estimates with minimal computational overhead, which is crucial for real-time control, especially on resource-constrained robotic platforms [32]. Additionally, the hardware cost of the state estimation system should be minimized for widespread practical application [87, 94].

To meet these requirements, modern mobile robots are typically equipped with a variety of sensors to fuse information from multiple sources [36], including inertial measurement units (IMUs), cameras, light detection and ranging sensors (LiDARs), wheel encoders, and Global Navigation Satellite Systems (GNSS), as shown in Figure 1.1. Among these sensors, inertial sensor stands out as the only sensor capable of independently estimating the full state of the robot without being affected by environmental conditions. In contrast, other sensors such as cameras and LiDARs are susceptible to various environmental factors, including

illumination changes, motion blur, and adverse weather conditions. Moreover, inertial sensors typically operate at a high frequency, typically between 100 Hz and 1000 Hz, while remaining cost-effective, compact, and requiring minimal power and computational resources. Consequently, IMUs play a central role in the sensor fusion framework for state estimation in mobile robots [60].



(a) Apollo autonomous car by Baidu.

(b) NeBula-SPOT by Boston Dynamics.

Figure 1.1: Different types of mobile robots equipped with multiple sensors.

However, due to the inherent noise and bias instability, IMUs are prone to rapid drift in state estimation, especially for low-cost sensors. To reduce this drift, aiding information is required for the inertial navigation system (INS). In typical sensor fusion framework for mobile robot state estimation, other sensors such as GNSS, LiDAR, and camera are commonly employed to aid the IMU. While these sensors can provide valuable complementary information, they are not always reliable across all environments. For instance, GNSS becomes unavailable in tunnels or indoor settings, cameras are vulnerable to illumination changes and lack of visual features in textureless scenes, while LiDARs are susceptible to performance degradation under adverse weather conditions and in highly unstructured environments [36]. As a result, there remains a critical need for self-contained aiding mechanisms to enhance the accuracy and robustness of IMU-based proprioceptive state estimation systems [98].

For wheeled robots, odometers, i.e., wheel encoders, are typically used to measure the vehicle velocity or traveled distance. In addition, the motion of the wheeled robots is generally governed by two non-holonomic constraints (NHC), which refers to the fact that the velocity of the wheeled robot in the plane perpendicular to the forward direction is almost zero when the vehicle does not slide on or jump off the ground [86, 87, 90]. The robot forward velocity measured by the odometer can be integrated with the NHC to formulate a 3D velocity measurement to aid the IMU, which has been proven as an effective way to improve the accuracy and robustness of the IMU-based state estimation system [86, 106]. However, the reliability of the odometer data is highly dependent on road conditions and vehicle maneuvers, which degenerates if relative slippage occurs between

the tires and contacting surface. In addition, fusing information from different systems is challenging because of different standards, hardware modification, data transfer synchronization, and difficulties in obtaining reliability information along with the data.

For legged robots, joint encoders measurements can be used to compute the leg kinematics and provide a 3D velocity measurement (similar to the role of wheel encoders in wheeled robots) to integrate with the body-mounted IMU (Body-IMU) upon detecting foot contact using foot force sensors. However, force sensors-based contact detection is susceptible to errors due to slippage and sensor degradation over time. In addition, the foot positions during swing phase cannot be estimated and need to be relocated when the foot regains contact with the ground, introducing further inconsistency and error into the robot state estimation [5, 117].

As previously discussed, multi-sensor fusion is essential for mobile robots to estimate their states accurately in different challenging environments, as no single sensor modality can reliably handle all real-world conditions. In conclusion, there are two major directions to develop the IMU-based state estimation systems to meet the key requirements of accuracy, robustness, and efficiency: (i) improving the IMU-based proprioceptive state estimation in a self-contained manner to continuously provide reliable state estimates when exteroceptive sensors fail, and (ii) investigating effective algorithms to fuse IMU with other complementary exteroceptive sensors to improve the overall performance.

This dissertation addresses the fundamental challenges of achieving accurate, robust, and efficient state estimation for mobile robots by introducing novel methods that enhance IMU-based estimation systems through the integration of complementary aids. The proposed approaches not only attain competitive accuracy relative to the state-of-the-art but are also designed with computational efficiency in mind, ensuring their applicability in real-world robotic deployments. Our proposed methods additionally demonstrate improved robustness under challenging conditions. Specifically, this thesis presents techniques that advance proprioceptive state estimation by fully exploiting the IMU's capability to capture motion and locomotion dynamics, as well as methods that fuse IMU data with exteroceptive measurements, including GNSS and LiDAR, to further improve state estimation performance and resilience.

1.2 Main Contributions

The main contributions of this thesis are novel approaches that exploit various aids to enhance the accuracy, robustness, and efficiency of the IMU-based state estimation systems. Our approaches use low-cost, compact commercial-grade

IMUs. These systems are specifically designed for: (i) wheeled robots, (ii) legged robots, and (iii) general-purpose mobile robots operating in both terrestrial and aerial environments.

First, to take advantage of the rotation modulation with the inherent rotation platform of the wheeled robots, this thesis proposes to mount the IMU on the wheel instead of the body of the robot. There are two major advantages of the wheel-mounted IMU (Wheel-IMU). First, it can directly calculate wheel speed by multiplying the gyroscope measurement and wheel radius without an odometer. Second, the continuous rotation of the wheel helps mitigate the constant IMU bias errors. Based on this, we propose Wheel-INS, a dead reckoning system using only a Wheel-IMU. Wheel-INS performs the strapdown INS algorithm to predict the robot state. At the same time, the wheel velocity calculated by the Wheel-IMU gyroscope outputs and wheel radius is treated as an external observation to update the state via an extended Kalman filter (EKF). Experimental results on multiple wheeled platforms demonstrate that the proposed Wheel-INS achieves relative position accuracy with approximately 1% drift over the traveled distance, representing a 23% reduction compared to the conventional odometer-aided INS (ODO/INS). Moreover, Wheel-INS exhibits significant robustness on uneven terrain compared to ODO/INS.

Although Wheel-INS exhibits excellent dead reckoning performance, it is only a relative positioning solution lacking the ability to limit long-term error drift. That is to say, the positioning error of Wheel-INS still accumulates over time. To correct the accumulated error without exteroceptive sensors, we propose Wheel-SLAM, extracting the terrain features sensed by the Wheel-IMU to enable loop closure detection. To the best of our knowledge, this is the first SLAM system using only one low-cost Wheel-IMU in the literature. Field experiments demonstrate the feasibility of performing SLAM using terrain features measured by the Wheel-IMU. Wheel-SLAM achieves an average root mean square error (RMSE) of 8.76 m in position and 2.67 ° in orientation over trajectories spanning several kilometers, representing improvements of approximately 52.6% and 53.2% over Wheel-INS in position and orientation accuracy, respectively.

However, Wheel-SLAM is only applicable for those applications where the robot moves in constrained environments while failing in scenarios where the robot does not have the opportunity to exactly revisit the places it has been before, for example, a self-driving car driving from one city to another. Therefore, external correction signals are necessary to limit the long-term error drift of Wheel-INS in large scale environments. To this end, we propose an integrtaed navigation system (called Wheel-GINS) that fuses Wheel-INS with the absolute positioning information from GNSS via an EKF framework. Experimental results demonstrate that Wheel-GINS achieves centimeter-level position accuracy

and orientation accuracy within 1 ° when GNSS signals are available. More importantly, during 30 s, 60 s, and 120 s GNSS outages, Wheel-GINS reduces position drift by an average of 32% compared to the traditional GNSS/Odometer/INS integrated navigation system (ODO-GINS). Additionally, Wheel-GINS can effectively estimate the Wheel-IMU installation parameters online and, consequently, improve the localization accuracy and practicality of the system. This is, to the best of our knowledge, the first reported GNSS/INS integrated navigation system that employs a Wheel-IMU. Our proposed state estimation systems specially using a Wheel-IMU for wheeled robots is presented in Chapter 4.

Second, for legged robots, we use the leg kinematics computed by the joint encoders measurements to aid the IMU. Specifically, we mount multiple IMUs on the lower legs of the robot and close to the feet (Leg-IMU). Each IMU performs their own state propagation which is then constrained with the robot's kinematic chain through an EKF. Rather than relying on force sensors, we use Leg-IMUs to detect foot contact. Field experiments show that the proposed system reduces absolute translation error by 8% and absolute rotation error by 13% compared with conventional leg odometry (using only a Body-IMU and joint encoders) across diverse terrains. In addition, the average preprocess time per IMU measurement is less than 1 ms in our system due to the efficient algorithm design. Our proposed proprioceptive state estimation system tailored for legged robots with multiple Leg-IMUs is presented in Chapter 5.

Lastly, for general-purpose robots that are not limited to the wheel and leg locomotion, we leverage 3D spacial geometry information from LiDAR to aid the IMU. In this study, the IMU is mounted on the main body of robot. We propose a tightly coupled LiDAR–inertial odometry system (named LIO-EKF) that integrates point-to-point matching and IMU propagation with an error-state EKF framework. We first use IMU to predict the robot state. Correspondences between current LiDAR points and the local map are established to construct the observation model, which is then used to correct the IMU prediction. To further enhance adaptability and ease of deployment, we introduce an adaptive data association model that reduces the number of parameters requiring manual tuning for different environments. Experimental results illustrate that the proposed system performs on par with the state-of-the-art LiDAR-inertial odometry pipelines while improving computation efficiency by more than two times. Our proposed efficient tightly-coupled LiDAR-inertial fusion state estimation system for general-purpose robots is detailed in Chapter 6.

Overall, this thesis presents novel approaches that exploit various aiding sources to enhance the accuracy, robustness, and efficiency of the IMU-based state estimation system for different types of mobile robots. To facilitate further research and promote reproducibility, we have made the implementations of all

proposed methods publicly available.

The remainder of this thesis is structured as follows. In Chapter 2, we review the studies most relevant to the approaches proposed in this thesis and briefly explains our contribution beyond existing works. In Chapter 3, we introduce the fundamental techniques that form the foundation of the algorithms developed in this thesis. The Wheel-IMU-based state estimation system for wheeled robots is presented in Chapter 4, followed by the multiple Leg-IMUs-based state estimation system for legged robots in Chapter 5. The LiDAR–inertial fusion-based state estimation system is introduced in Chapter 6. Finally, Chapter 7 concludes the thesis and outlines potential directions for future research.

Chapter 2

Related Work

A large body of research has been conducted in the field of aided inertial navigation for mobile robot state estimation, making it a vibrant topic at the intersection of the navigation and robotics communities. Numerous textbooks [36, 83, 89, 95] and survey papers [8, 15, 46, 55, 60, 65, 130] focus on the techniques of INS and its integration with other sensors. Cadena et al. [8] surveys the history and current state of a broad set of topics in SLAM, such as robustness and scalability in long-term mapping, metric and semantic representations, theoretical performance guarantees, and other new frontiers, while also outlining open challenges and emerging research issues. Lee et al. [55] provides an overview of recent advances in LiDAR-inertial odometry, while Huang [46] offers a concise review of visual-inertial navigation. Barros et al. [65] presents a comprehensive survey on visual-inertial odometry and SLAM. Chen et al. [15] reviews deep learning-based inertial positioning and its applications in tracking pedestrians, drones, vehicles, and robots. A thorough investigation into multi-sensor integrated navigation systems is presented by Zhuang et al. [130], who categorize fusion algorithms into two main types: (i) analytic-based fusion and (ii) learning-based fusion, with detailed discussions from multiple perspectives. Li et al. [60] reviews the use of machine learning techniques to enhance inertial sensing in various aspects, including sensor design and selection, calibration and error modeling, navigation and motion-sensing algorithms, multi-sensor information fusion, system evaluation, and practical applications.

In this chapter, we briefly review the studies in the literature that are closely related to this thesis and discuss the advantages of our proposed approaches compared to the existing methods. We divided this chapter into five sections in which we deal with individual aspects of the approaches of this thesis. In Section 2.1, we present related works that use Wheel-IMUs for state estimation. Since this is not a well-explored topic, we also review the methods that use Wheel-IMUs to replace wheel encoders for velocity estimation. This is related to all our proposed

Wheel-IMU-based state estimation systems, namely, Whee-INS, Wheel-SLAM and Wheel-GINS. In Section 2.2, we review the state estimation methods using foot-mounted IMUs (Foot-IMUs) for pedestrian navigation. These methods are related to our proposed DogLegs system, which uses Leg-IMUs for legged robot state estimation. In addition, the methods using Foot-IMUs for SLAM are related to our proposed Wheel-SLAM system. Section 2.3 reviews the GNSS/INS integration methods, which are related to our proposed Wheel-GINS system. Section 2.4 reviews the proprioceptive state estimation methods using IMUs and joint encoders for legged robots, which are related to our proposed DogLegs system. Finally, in Section 2.5, we review the LiDAR-inertial odometry methods for mobile robot state estimation, which are related to our proposed LIO-EKF system.

2.1 State Estimation using Wheel-IMUs

To provide a low-cost alternative for the wheel odometer, researchers have proposed various approaches to calculate the wheel speed or traveled distance of the vehicle using the wheel-mounted inertial sensors. Youssef et al. [122] proposed detecting peaks and valleys in the accelerometer signal to count complete wheel cycles and compute the traveled distance of the robot. Coulter et al. [22] used the Wheel-IMU accelerometer to measure the wheel's rotation angle and movement duration. Gersdorf et al. [34] employed a gyroscope on the wheel's rotation axis and two accelerometers on the wheel plane to measure vehicle acceleration and estimate wheel velocity via EKF. Nonetheless, these studies only focused on providing the velocity information instead of the full state estimation of the wheeled vehicles.

The first state estimation system based solely on a Wheel-IMU was proposed by Collin et al. [20, 21]. This system utilized two accelerometers positioned in the plane of the wheel to estimate the wheel's rotation angle and, consequently, the traveled distance. Simultaneously, gyroscope measurements were used to compute the vehicle's heading. Based on this information, the robot's position was then estimated using a 2D dead reckoning approach. However, the method relied on the assumption of a constant vehicle speed and did not account for misalignment errors. As a result, it was unable to cope with variations in speed or misalignment between the Wheel-IMU and the wheel center.

To address the limitations of the previous work, our proposed Wheel-INS adopted an error-state EKF to estimate the vehicle's position, velocity, and attitude using only the Wheel-IMU. We employed the strapdown INS to predict the robot's state and use the wheel speed as a measurement to correct the predicted state. The wheel speed was obtained by multiplying the angular velocity of the Wheel-IMU gyroscope with the known wheel radius. Unlike earlier methods,

Wheel-INS was not constrained by the assumption of constant speed. In addition to estimating 2D position and heading as in prior work, Wheel-INS also estimated the robot's roll angle, which provided valuable terrain information for loop closure detection in our subsequent Wheel-SLAM system. Moreover, our approach accounted for both position and attitude misalignment between the Wheel-IMU and the wheel, significantly improving overall system accuracy.

2.2 Pedestrian Navigation using Foot-mounted IMUs

State estimation using only IMU data has been a long-standing research topic in the field of pedestrian navigation. There are two main approaches in this area, distinguished by their sensor configurations. The first is pedestrian dead reckoning (PDR) systems based on commercial devices such as smartphones [14, 43]. The second involves tracking systems designed for professional applications, such as first responders or tunnel inspectors, which use foot-mounted IMUs (Foot-IMUs). In this thesis, we focus on the latter, as it aligns more closely with the insights underlying our proposed Wheel-INS, Wheel-SLAM, and DogLegs systems.

By detecting periodic foot-ground contact using a Foot-IMU, foot-mounted inertial navigation systems (Foot-INS) exploit the zero-velocity update (ZUPT) to mitigate IMU error drift. Foxlin [30] pioneered the concept of Foot-INS, who demonstrated that ZUPT can significantly reduce the accumulated errors in INS, thereby improving the accuracy of position and orientation estimates. Building upon this, Niu [72] proposed a post-processing method to align pedestrian trajectories across multiple floors by leveraging floor-wise building orientations inferred from the trajectories themselves. More recently, Kuang et al. [53] introduced a shin-mounted inertial navigation system, where the IMU is placed on the pedestrian's shin rather than the foot. This design offers improved usability by eliminating the need for special footwear while maintaining reliable state estimation.

Despite its widespread application in pedestrian indoor navigation and its effectiveness in reducing IMU drift, Foot-INS has seen limited application in legged robots. Unlike humans, legged robots, especially quadrupeds exhibit distinct gaits with less stable foot-ground contact. Moreover, they often operate in challenging terrains, introducing estimation errors when applying conventional pedestrian Foot-INS methods. Therefore, adapting Foot-INS to legged robots remains an open research challenge. Our proposed DogLegs system aims to address this challenge.

To further improve the accuracy and reliablility of Foot-INS, Angermann et al. [2] proposed a pedestrian SLAM system using only a Foot-IMU (FootSLAM) by taking advantage of human perception and cognition. A dynamic Bayesian network was employed to represent the fact that when a pedestrian walks in a constrained environment, e.g., an office building, he or she relies mainly on visual cues to avoid obstacles and determine accessible areas. Specifically, the algorithm was implemented based on a PF where the weights of the particles were updated by the probability of the pedestrian crossing transitions in a regular 2D grid of adjacent hexagons. After that, a probabilistic transition map implicitly encoding the environmental features that influence the pedestrian's visual impression and intention was constructed. The authors further extended FootSLAM to Feet-SLAM [82] by integrating multiple odometry result from multiple pedestrains to improve both the convergence speed and accuracy.

Our Wheel-SLAM system draws inspiration from FootSLAM. Unlike Foot-SLAM, which implicitly leverages building layout through human cognition to perform SLAM, Wheel-SLAM explicitly extracts terrain features from Wheel-IMU for loop closure detection. Furthermore, while FootSLAM adopts a hexagonal grid map to accommodate the complexity of human locomotion, Wheel-SLAM simplifies the grid structure to a square format, reflecting the more constrained and predictable motion patterns of wheeled robots.

2.3 GNSS/INS Fusion

Although GNSS/Wheel-IMU fusion has not been investigated, the GNSS/INS integrated navigation system, using a normal IMU mounted on the body of the robot, has been extensively studied in the past decades [3, 42, 99, 102, 126]. Nowadays, it has become a standard component of many vehicle navigation systems. GNSS/INS integration framework can be divided into loosely coupled and tightly coupled, depending on which measurement from GNSS is used. In the loosely coupled system [45, 86, 123], the position (and velocity) estimated by the GNSS receiver is fused with INS, while the tightly coupled system [99, 101, 114] directly integrates raw GNSS measurements, such as pseudorange and carrier phase measurements. Although various methods, such as factor graphs [64, 91, 125], have been proposed to fuse GNSS with INS, the most common one is still the EKF [86, 102]. Because our proposed Wheel-GINS focuses on investigating the idea and feasibility of fusing GNSS with Wheel-IMU instead of algorithm research for GNSS/INS fusion, we adopt the classical loosely coupled framework and use the EKF in the proposed Wheel-GINS.

To improve the accuracy of GNSS/INS integrated navigation system during GNSS outages, various sensors, such as cameras [19, 59, 75], LiDARs [12, 58, 63],

and odometers [73, 86, 128] have been introduced into the GNSS/INS integrated system. Unlike cameras and LiDARs, which depend on environmental conditions, odometers are preferable due to their independence from the environment [78, 113]. The robot velocity provided by the odometer is always combined with the NHC as a 3D velocity measurement to fuse with INS. Because the proposed Wheel-GINS achieves a similar information fusion as the traditional GNSS/Odometer/INS integrated navigation system, we use it as a benchmark to illustrate the performance of Wheel-GINS.

2.4 Proprioceptive Legged Robot State Estimation

State estimation for the main body of legged robots has received significant attention in recent years. While numerous sensor fusion-based systems leveraging various exteroceptive sensors, e.g., cameras [51, 104, 118] and LiDARs [10, 77, 105], have been proposed, this thesis focuses specifically on proprioceptive sensors, including IMUs and joint encoders.

An early approach to proprioceptive state estimation for legged robots was proposed by Bloesch et al. [5], who employed an EKF to fuse outputs from a Body-IMU, joint encoders, and foot force sensors. This method augmented the foot contact position into the state and updated it with the main body state using the leg kinematics when the foot contacts ground. This approach has become the basis for many subsequent state estimation frameworks. Later, Bloesch et al. [4] extended this work by incorporating the zero velocity measurement and replacing the EKF with an unscented Kalman filter to improve accuracy and robustness.

Taking advantages of the symmetry of state estimation problem, researchers have introduced the invariant extended Kalman filter (InEKF) into the legged robot state estimation, using Lie Groups and Lie algebra for state propagation and error tracking [39, 40, 50, 62, 93, 121]. Hartley et al. [39, 40] derived a continuous time right-invariant EKF for an IMU/contact process model with a forward kinematic measurement model. This method used the same sensor input as Bloesch et al. [4, 5] but with a different state estimator. Lin et al. [62] trained a network to estimate the foot contact and used it in a similar InEKF framework. Kim et al. [50] and Yoon et al. [121] extended InEKF into a fixed-lag invariant smoother, coupled with a foot slippage detection method, which improved state estimation accuracy by optimizing robot states within a sliding window.

To improve the sensing ability of legged robots, additional IMUs have been incorporated into their lower legs. Kolvenbach et al. [52] embedded IMUs in robot feet to inspect concrete deterioration through a specialized scratching motion;

however, these IMUs were not utilized for main body state estimation. Maravgakis et al. [66] proposed estimating the probability of stable foot contact with Leg-IMUs by employing a kernel density estimator to approximate the probability density function. Yang et al. [117] placed multiple IMUs on the robot feet to provide additional foot velocity measurements within the similar EKF structure as Bloesch et al. [5]. However, the potential of IMUs for full attitude estimation was overlooked, and the sensor bias of the multiple IMUs was also not addressed in this study.

In summary, existing legged robots equipped with Leg-IMUs have yet to fully leverage the full-state pose estimation capabilities of the Leg-IMUs. This leaves significant potential for integrating multiple IMUs to reduce random errors of the IMUs, thereby enhancing the accuracy and robustness of main body state estimation.

2.5 LiDAR-Inertial Odometry

A typical LIO system can be divided into two main components: front end and back end [7]. In the front end, the system performs data association to find corresponding points between the current LiDAR scan and the previous scan (or a map) [23, 25, 38]. In contrast, the back end utilizes various state estimation methods to fuse information from both LiDAR and IMU sensors to estimate the robot's pose.

In the front end of LIO systems, feature-based registration is the predominant paradigm. Feature-based registration, exemplified by LOAM [124] and its variants [61, 84, 100], involves extracting edges and planar patches from LiDAR scans and then finding corresponding features in the map to estimate the robot pose. LIO-SAM [85] builds on the front end of LOAM and optimizes the robot state estimates from LiDAR and IMU using a sliding window based on a factor graph. LIO-Mapping [119] uses a similar front end but includes a rotation-constrained refinement method to further enhance pose estimation and point-cloud maps. LINS [79] also relies on edges and planar features for data association. LiLi-OM [57] and FAST-LIO [116] tailor a similar feature extraction module for solid-state LiDARs. Although FAST-LIO2 [115] proposes to register the raw points without feature extraction, it relies on the point-to-plane metric, which also needs local plane approximation and normal vector computation. All of the above approaches require parameter tuning for feature extraction, depending on the specific LiDAR sensor in use and the structure of the environment. Conversely, our proposed LIO-EKF removes such a limitation and uses the classical point-to-point metric in the registration [97].

In terms of the back end, most LiDAR-inertial odometry systems rely on either

the iterated extended Kalman filter (IEKF) [115, 116] or factor graph optimization [48, 64, 68]. Although earlier methods fuse LiDAR and IMU measurements using factor graphs [57, 85, 119], recently the IEKF approaches [79, 115, 116] are gaining prominence. In an IEKF, the correction step of the filter is performed multiple times to reduce the linearization errors at the cost of increased computation. In our proposed LIO-EKF system, we suggest that, in most scenarios, the classical error-state EKF can be employed to fuse LiDAR scans and IMU readings effectively without additional iterations. This is because that we use a proper strapdown INS model [86], which can provide an accurate initial pose estimate between two successive scans.

Overall, our proposed LIO-EKF employs a point-to-point registration in the front end and uses classical EKF without iterations. Although composed of very straightforward components, our approach achieves comparable performance to the state-of-the-art methods in pose estimation accuracy while being significantly faster.

Chapter 3

Basic Techniques

We have motivated in Chapter 1 why it is crucial to exploit aiding information to improve the accuracy, robustness and efficiency of the IMU-based state estimation systems for mobile robots. We have also introduced typical aiding sources commonly integrated with IMUs. Specifically, this thesis presents several novel aided-INS approaches for mobile robot state estimation. All of these methods share a common foundation: the inertial navigation system. In addition, most of our proposed methods are built upon the EKF framework. Our choice of EKF is motivated by two main reasons. First, it is one of the most widely used method for nonlinear state estimation which is straightforward to implement our ideas. Second, it exhibits excellent computational efficiency, which is crucial for real-time applications [94]. It is worth noting that the proposed methods can be readily extended to other nonlinear state estimation methods, such as unscented Kalman filters (UKF) [47] and factor graphs [24].

In this chapter, we briefly present the fundamental techniques that underpin the main contributions of this thesis. We begin by defining the coordinate systems used throughout the thesis, which are essential for understanding both inertial navigation and mobile robot state estimation. Next, we present the basic concepts of inertial navigation, including inertial sensors, the strapdown INS algorithm, and the inertial navigation error propagation model. Finally, we present the EKF pipeline, which forms the core of our proposed aided-INS methods.

3.1 Notations

Throughout this thesis, we use the following notations.

- Vectors are represented by lowercase bold italic letters (e.g., $\boldsymbol{x}$).
- Matrices are denoted by uppercase bold letters (e.g., $\mathbf{A}$), and the transpose

of a matrix is represented by a superscript $^\top$ (e.g., $\mathbf{A}^\top$).

- $\mathbf{R}_a^b$ represents the rotation matrix that transforms coordinates from frame a to frame b.
- $\boldsymbol{x}^a$ represents the vector $\boldsymbol{x}$ expressed in frame a.
- $\hat{\boldsymbol{x}}$ denotes the estimated value or computed value of a variable $\boldsymbol{x}$, which generally deviates from the true value.
- $\tilde{\boldsymbol{x}}$ denotes the measured or observed value of a variable $\boldsymbol{x}$, which also deviates from the true value.
- $\delta\boldsymbol{x}$ denotes the error of a variable $\boldsymbol{x}$.

3.2 Coordinate Systems

The reference coordinates should be first defined to describe the state of the mobile robot. Throughout this thesis, we adopt a three-dimensional Cartesian right-handed coordinate system, in which the three coordinate axes are mutually orthogonal and follow the right-hand rule. That is, the cross product of the unit vectors in the positive directions of the x-axis and y-axis points in the positive direction of the z-axis. The coordinate systems used in this thesis are summarized in Table 3.1. Figure 3.1 provides an illustration of the coordinate frames commonly used in mobile robot state estimation.

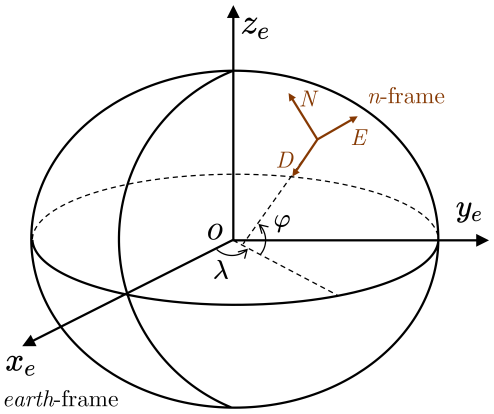
In typical applications, robots usually operate within relatively small-scale environments, such as providing services inside buildings, monitoring crops in fields, and exploring caves and tunnels. In these scenarios, the robot primarily needs to estimate its relative state within the local environment, rather than its absolute state with respect to the Earth. Therefore, we define the *world*-frame at the initial position of the robot as the reference frame to express the robot's state.

In contrast, for large-scale applications, such as autonomous vehicles traveling across cities, where GNSS is integrated with INS, it becomes necessary to consider both the *earth*-frame and the navigation frame (n-frame) to describe the robot's absolute position and heading in a global context.

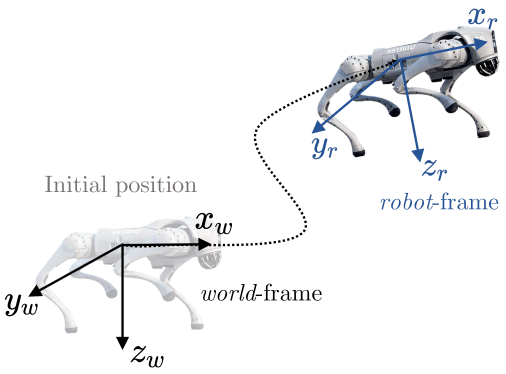
This thesis mainly focuses on pose estimation within local environments. Accordingly, we consider several local coordinate frames, including the *robot*-frame, *world*-frame, IMU *body*-frame, and, for wheeled robots, the *wheel*-frame. An exception is presented in Section 4.3, where the *earth*-frame and n-frame are also involved to describe the GNSS/Wheel-IMU integrated navigation system.

Table 3.1: Definitions of the Coordinates Systems.

Symbol	Description	Definition
earth-frame	The Earth-Centered Earth-Fixed (ECEF) coordinates system.	*origin*: the center of mass of the Earth. *x-axis*: toward the mean meridian of Greenwich. *y-axis*: completing a right-handed orthogonal frame. *z-axis*: parallel to the mean spin axis of the Earth.
n-frame	The local navigation frame.	*origin*: the same as *robot*-frame. *x-axis*: north. *y-axis*: east. *z-axis*: downward vertically.
robot-frame	The coordinates system of the robot body.	*origin*: the mass center of the robot. *x-axis*: forward. *y-axis*: right. *z-axis*: down.
world-frame	The local reference frame.	*origin*: the same as *robot*-frame at the initial position of the robot. *x-* and *y-axes* are on the local horizontal plane with *x-axis* pointing to the robot forward direction. *z-axis*: aligned with the local gravity direction.
wheel-frame	The coordinates system of the wheel.	*origin*: wheel center. *x-axis*: right, perpendicular to the wheel plane. *y-* and *z-axes* are parallel to the wheel plane to complete a right-handed orthogonal frame.
body-frame	The coordinates system of the IMU.	*origin*: IMU measurement center. *Wheel-IMU*: x-axis pointed to right, parallel to the rotation axis of the wheel, *y-* and *z-axes* are perpendicular to the x-axis. *Body-IMU*: the same as *robot*-frame. *Leg-IMU*: the same as *robot*-frame.



(a) Illustration of the *earth*-frame and *n*-frame. N, E, and D represent North, East, and Down, respectively. x_e-y_e-z_e represent the *earth*-frame. λ and φ are the longitude and latitude, respectively.



(b) Illustration of the *world*-frame and *robot*-frame. x_w-y_w-z_w represent the *world*-frame, while x_r-y_r-z_r represent the *robot*-frame. The *body*-frame of the Body-IMU is aligned with the *robot*-frame.

Figure 3.1: Illustration of the coordinate frames commonly used in mobile robot state estimation.

It is important to clarify that the *world*-frame used throughout this thesis does not represent a global reference coordinate system. Instead, it is a local reference frame defined at the robot's initial position, aligned with the gravity vector. Additionally, the x-axis of the *world*-frame is aligned with the robot's forward direction, so that the initial heading of the robot is zero.

3.3 Inertial Navigation

Inertial navigation is a technique for estimating the position, velocity, and attitude of a moving object using inertial sensors, such as accelerometers and gyroscopes. The inertial sensors measure the specific force and angular velocity of the object, which can be integrated over time to obtain the object's motion state. This process can also be seen as a form of dead reckoning. It is important to note that the term "dead reckoning" in this thesis refers to the general method of computing the robot's relative position based on pose increments, rather than being limited to the traditional 2D dead reckoning approach based on traveled distance and heading.

3.3.1 Inertial Sensors

Inertial sensors usually consist of three orthogonal accelerometers and three orthogonal gyroscopes. The accelerometers measure the non-gravitational force per

unit mass, i.e., specific force, which is the acceleration of the body minus the gravitational acceleration, i.e., $\boldsymbol{f} = \boldsymbol{a} - \boldsymbol{g}$, while the gyroscopes measure the angular velocity of the body. However, IMUs are subject to various errors, including bias, scale factor, misalignment, non-orthogonality, and noise [86].

In this thesis, we primarily consider bias, scale factor, and noise errors, as these are the most impactful error sources for low-cost IMUs [86]. Bias refers to a constant offset in sensor output, scale factor errors involve proportional discrepancies between actual and measured values, and noise encompasses random fluctuations in measurements [36]. These errors can accumulate over time, leading to significant drift in position and orientation estimates. Therefore, accurate modeling and compensation of these errors are crucial for enhancing the performance of IMU-based state estimation systems. Accordingly, the raw IMU measurement can be expressed as:

$$\begin{aligned} \tilde{\boldsymbol{f}} &= \boldsymbol{f} + \boldsymbol{b}_a + \operatorname{diag}(\boldsymbol{f})\boldsymbol{s}_a + \boldsymbol{n}_a, \\ \tilde{\boldsymbol{\omega}} &= \boldsymbol{\omega} + \boldsymbol{b}_g + \operatorname{diag}(\boldsymbol{\omega})\boldsymbol{s}_g + \boldsymbol{n}_g, \end{aligned} \tag{3.1}$$

where $\tilde{\boldsymbol{f}}$ and $\tilde{\boldsymbol{\omega}}$ are the specific force and angular velocity measured of the IMU expressed in the IMU *body*-frame, respectively; $\boldsymbol{f}$ and $\boldsymbol{\omega}$ are the true specific force and angular velocity of the body; $\boldsymbol{b}_a$ and $\boldsymbol{b}_g$ are the bias errors of the accelerometers and gyroscopes, respectively; $\boldsymbol{s}_a$ and $\boldsymbol{s}_g$ are the scale factor errors of the accelerometers and gyroscopes, respectively; $\boldsymbol{n}_a$ and $\boldsymbol{n}_g$ are the noise of the accelerometers and gyroscopes, respectively; $\operatorname{diag}(\cdot)$ is the diagonal matrix form of a vector.

3.3.2 Strapdown INS

The inertial navigation system autonomously estimates a vehicle's position, velocity, and attitude over time relative to an initial state, using only the specific force and angular velocity measurements from an IMU. INS can be broadly categorized into two types [83]: strapdown INS, where the IMU is mounted to the frame of the platform, and stabilized INS (or gimballed INS), where the IMU is mounted on an inner multi-axis gimbal system designed to maintain its orientation constant with respect to an inertial frame. In robotics, strapdown INS is predominantly used, as it is more compact, cost-effective, and suitable for mobile platforms where the IMU is directly affixed to the robot's body. In robotics, the term inertial odometry is commonly used as a synonym of INS [11], to emphasize the odometric nature of the estimates. In this thesis, we adopt the term INS and implement the strapdown INS algorithm.

The traditional strapdown INS algorithm is designed for high-grade IMUs, which are expensive and highly accurate. As a result, these algorithms typically

incorporate sophisticated mathematical models that account for various physical effects, such as Earth's rotation and the rotation of n-frame with respect to *earth*-frame. However, in this thesis, we focus on low-cost IMUs, which exhibit significantly larger noise and bias instability compared to their high-grade counterparts. Consequently, we adopt a simplified strapdown INS algorithm that neglects terms whose influence is negligible relative to the dominant sources of error, such as bias and noise. Assuming the IMU is aligned with the robot, the simplified strapdown INS equations (witout considering the state error and IMU measurement error) used in this thesis can be summarized as [83, 95]:

$$\begin{cases} \boldsymbol{p}_k^w = \boldsymbol{p}_{k-1}^w + \frac{1}{2}(\boldsymbol{v}_{k-1}^w + \boldsymbol{v}_k^w)\,\Delta t_k \\ \boldsymbol{v}_k^w = \boldsymbol{v}_{k-1}^w + \boldsymbol{g}\,\Delta t_k + \mathbf{R}_{b\,k-1}^w \Big[\boldsymbol{f}_k\,\Delta t_k + \frac{1}{2}(\boldsymbol{\omega}_k \times \boldsymbol{f}_k)\,\Delta {t_k}^2 \\ \qquad + \frac{1}{12}(\boldsymbol{\omega}_{k-1} \times \boldsymbol{f}_k + \boldsymbol{\omega}_k \times \boldsymbol{a}_{k-1})\,\Delta {t_k}^2 \Big] \\ \mathbf{R}_{b\,k}^w = \mathbf{R}_{b\,k-1}^w \mathrm{Exp}(\boldsymbol{\omega}_k\,\Delta t_k + \frac{1}{12}(\boldsymbol{\omega}_{k-1} \times \boldsymbol{\omega}_k)\,\Delta {t_k}^2) \end{cases}, \tag{3.2}$$

where $\boldsymbol{p}^w$, $\boldsymbol{v}^w$, and $\mathbf{R}_b^w$ represent the true position, velocity, and rotation matrix of the robot with respect to the *world*-frame, respectively; k and $k-1$ represent the current and previous time steps, respectively; Δt_k is the time interval between the two time steps; $\boldsymbol{f}_k$ and $\boldsymbol{\omega}_k$ are the true value of the specific force and angular velocity at time step k in the IMU *body*-frame; $\boldsymbol{g}$ is the gravity vector in the *world*-frame; $\mathrm{Exp}(\cdot)$ is the matrix exponential function transforming the rotation vector to rotation matrix. Compared to the simplified IMU kinematics model widely used in the robotics community [85, 89], we take into account the second-order terms in the velocity and attitude update equations to reduce the truncation error during the discrete integration process, especially when the robot undergoes high dynamics, e.g., rotating with the wheel.

In sum, given the initial state and the IMU measurements, the strapdown INS algorithm can recursively compute the robot's position, velocity, and attitude with respect to that initial state over time using Equation (3.2).

3.3.3 INS Error Propagation

Due to the inherent errors in the IMU measurements, the strapdown INS inevitably has estimation error at every time epoch which accumulates over time. The estimation errors are defined as the difference between the true value and the estimated value [86], i.e.,

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \delta\boldsymbol{x}, \tag{3.3}$$

where $\hat{\boldsymbol{x}}$ is the estimated value, $\boldsymbol{x}$ is the true value, and $\delta\boldsymbol{x}$ is the error. Consequently, the robot state errors of the strapdown INS can be expressed as:

$$\begin{cases} \hat{\boldsymbol{p}} = \boldsymbol{p} + \delta\boldsymbol{p} \\ \hat{\boldsymbol{v}} = \boldsymbol{v} + \delta\boldsymbol{v} \\ \hat{\mathbf{R}}_b^w = (\mathbf{I} - \boldsymbol{\phi}\times)\mathbf{R}_b^w \end{cases}, \tag{3.4}$$

while the IMU measurement errors can be expressed as:

$$\begin{cases} \tilde{\boldsymbol{f}} = \boldsymbol{f} + \delta\boldsymbol{f} \\ \tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \delta\boldsymbol{\omega} \end{cases}, \tag{3.5}$$

where $\mathbf{R}_b^w$ is the rotation matrix from the IMU *body*-frame to the *world*-frame; $\boldsymbol{\phi}$ represents the rotation error; $\mathbf{I}$ is the 3×3 identity matrix. Combining Equation (3.1) and Equation (3.5), we have $\delta\boldsymbol{f} = \boldsymbol{b}_a + \text{diag}(\boldsymbol{f})\boldsymbol{s}_a + \boldsymbol{n}_a$ and $\delta\boldsymbol{\omega} = \boldsymbol{b}_g + \text{diag}(\boldsymbol{\omega})\boldsymbol{s}_g + \boldsymbol{n}_g$.

The variation of the errors of INS over time can be described by a set of differential equations. As discussed in previous section, we adopt a simplified version of strapdown INS algorithm by neglecting the trivial terms. Similarly, we apply the same simplification to the error propagation model. In addition to the robot pose error, the inertial sensor error, namely, bias and scale factor error also evolve over time. These are usually also estimated in the INS. In this thesis, we model both IMU bias and scale factor errors using a first-order Gauss-Markov process [86]. The continuous-time error propagation model of the strapdown INS can be expressed as follows:

$$\begin{cases} \delta\dot{\boldsymbol{p}} = \delta\boldsymbol{v} \\ \delta\dot{\boldsymbol{v}} = (\mathbf{R}_b^w\boldsymbol{f}) \times \boldsymbol{\phi} + \mathbf{R}_b^w\delta\boldsymbol{f} \\ \dot{\boldsymbol{\phi}} = -\mathbf{R}_b^w\delta\boldsymbol{\omega} \\ \dot{\boldsymbol{b}}_g = -(1/\tau_{b_g})\boldsymbol{b}_g + \boldsymbol{w}_{b_g} \\ \dot{\boldsymbol{b}}_a = -(1/\tau_{b_a})\boldsymbol{b}_a + \boldsymbol{w}_{b_a} \\ \dot{\boldsymbol{s}}_g = -(1/\tau_{s_g})\boldsymbol{s}_g + \boldsymbol{w}_{s_g} \\ \dot{\boldsymbol{s}}_a = -(1/\tau_{s_a})\boldsymbol{s}_a + \boldsymbol{w}_{s_a} \end{cases}, \tag{3.6}$$

where τ_{b_g}, τ_{b_a}, τ_{s_g}, and τ_{s_a} are the correlation time in the first-order Gauss-Markov model of the gyroscope bias, accelerometer bias, gyroscope scale factor, and accelerometer scale factor, respectively; $\boldsymbol{w}_{b_g}$ and $\boldsymbol{w}_{b_a}$ denote the driving white noise of the residual bias errors of the gyroscope and accelerometer, respectively; $\boldsymbol{w}_{s_g}$ and $\boldsymbol{w}_{s_a}$ denote the driving white noise of the scale factor errors of the gyroscope and accelerometer, respectively; $\times$ represents the cross product of two vectors. The detailed derivation process and the discrete-time model can be found in [83, 95].

3.4 Extended Kalman Filter

EKF is a widely used method for nonlinear state estimation. It is an extension of the classical Kalman filter, which is designed for linear systems. The EKF approximates the nonlinear system by linearizing it around the current estimate using Taylor series expansion. To simplify the expression, here we do not consider the estimation error and measurement error to derive the EKF equations. The ideal system model and observation model (without error) in an EKF can be expressed as follows:

$$\begin{aligned} \boldsymbol{x}_k &= f(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k), \\ \boldsymbol{z}_k &= h(\boldsymbol{x}_k, \boldsymbol{v}_k), \end{aligned} \tag{3.7}$$

where $\boldsymbol{x}_k$ is the state vector at time step k; $\boldsymbol{z}_k$ is the measurement vector at time step k; $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the process noise and measurement noise, respectively. The process noise and measurement noise are assumed to be zero-mean Gaussian white noise with covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively, namely, $\boldsymbol{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$ and $\boldsymbol{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$. $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are independent of each other and also independent of the previous state $\boldsymbol{x}_{k-1}$.

The EKF consists of two main steps: prediction and update. In the prediction step, the EKF uses the system model to predict the robot state and its covariance. The state transition model is usually represented as a nonlinear function of the current state and control input. The predicted state and covariance can be expressed as:

$$\begin{aligned} \boldsymbol{x}_{k|k-1} &= f(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k), \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^\top + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^\top, \end{aligned} \tag{3.8}$$

where $\boldsymbol{x}_{k|k-1}$ is the predicted state at time step k given the previous state estimate $\boldsymbol{x}_{k-1}$; $\mathbf{P}_{k|k-1}$ is the predicted covariance matrix; $\mathbf{F}_{k-1}$ is the Jacobian matrix of the state transition function with respect to the state, namely, $\mathbf{F}_{k-1} = \frac{\partial f}{\partial \boldsymbol{x}_{k-1}}$; $\mathbf{G}_{k-1}$ is the Jacobian matrix of the state transition function with respect to the process noise, namely, $\mathbf{G}_k = \frac{\partial f}{\partial \boldsymbol{w}_k}$; and $\mathbf{Q}_k$ is the process noise covariance matrix. In the update step, the EKF uses the measurement model to update the predicted state and covariance based on the new measurement. The updated state and covariance can be expressed as:

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{x}_{k|k-1} + \mathbf{K}_k(\boldsymbol{z}_k - h(\boldsymbol{x}_{k|k-1})), \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^\top + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^\top, \end{aligned} \tag{3.9}$$

where $\boldsymbol{z}_k$ is the measurement vector at time step k; $h(\cdot)$ is the measurement function; $\boldsymbol{x}_k$ is the updated state estimate; $\mathbf{K}_k$ is the Kalman gain; and $\mathbf{H}_k$ is the Jacobian matrix of the measurement function with respect to the state, namely, $\mathbf{H}_k = \frac{\partial h}{\partial \boldsymbol{x}_k}$. While $\mathbf{P}_k$ can be expressed in several equivalent forms, we employ

the Joseph form [86] in this work, as it guarantees the symmetry and positive definiteness of the $\mathbf{P}_k$ matrix. The Kalman gain can be computed as:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^\top(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\top + \mathbf{R}_k)^{-1}, \tag{3.10}$$

where $\mathbf{R}_k$ is the measurement noise covariance matrix.

The EKF algorithm starts with an initial state estimate and covariance, and then iteratively updates the state estimate and covariance based on the new measurements. However, EKF requires linearization of the nonlinear system model and measurement model, which can introduce linearization errors. The linearization errors are caused by the approximation of the nonlinear functions using Taylor series expansion, which only considers the first-order terms and neglects the higher-order terms. This can lead to inaccurate state estimates, especially in highly nonlinear systems. The linearization error is more pronounced when the state estimate is far from the true state, which can occur in cases of large initial errors or rapid changes in the system dynamics. In such cases, the EKF may diverge or produce unreliable estimates.

To mitigate the nonlinearity error, we employ the error state formulation as the system state in this thesis. Since the error state typically has a smaller magnitude than the full state, the linearization error introduced by neglecting higher-order terms is significantly reduced. The continuous-time error state model is given by Equation (3.6). The error state vector can be summarized as:

$$\delta\boldsymbol{x} = \left[\delta\boldsymbol{p}^\top \ , \ \delta\boldsymbol{v}^\top \ , \ \boldsymbol{\phi}^\top \ , \ \boldsymbol{b}_g^\top \ , \ \boldsymbol{b}_a^\top, \ \boldsymbol{s}_g^\top \ , \ \boldsymbol{s}_a^\top\right]^\top \in \mathbb{R}^{21}. \tag{3.11}$$

It is important to note that the scale factor errors of the gyroscope and accelerometer are not always included in the state vector, as their observability requires significant motion of the robot. In this thesis, we incorporate these errors only in the Wheel-IMU-based state estimation system, where the Wheel-IMU undergoes continuous rotation with the wheel, providing sufficient excitation to estimate the scale factor errors reliably. In other cases, we do not estimate the IMU scale factor errors and instead set them to zero in order to reduce the dimensionality of the state vector.

Chapter 4

State Estimation for Wheeled Robots Using a Wheel-Mounted IMU

Wheeled robots are among the most widely used robotic platforms in both daily life and industrial environments due to their efficient mobility. Examples include autonomous driving cars, logistics robots, and cleaning robots. To provide accurate and robust proprioceptive state estimation when exteroceptive sensors (e.g., GNSS, cameras, and LiDARs) are unreliable, wheel encoders and motion constraints are often used to integrate with IMU for wheeled robot dead reckoning.

Typically, the motion of wheeled robots is subject to two constraints: the velocity component perpendicular to the vehicle's forward direction remains near zero when the vehicle does not slide or lift off the ground [86]. This is known as non-holonomic constraint (NHC). Combining NHC with the forward velocity measured by the odometer, a 3D velocity observation can be fused with INS. This method has been proven an effective approach to limit the error drift of INS [106, 113]. However, the reliability of odometer data is highly sensitive to road conditions and robot maneuvers, and its performance degrades in the presence of wheel slippage. Furthermore, fusing data from heterogeneous systems presents additional challenges, including differences in hardware standards, the need for system modifications, synchronization issues, and the difficulty of acquiring reliability metrics for each data source.

To enhance the accuracy and robustness of the proprioceptive state estimation systems and its fusion with exteroceptive sensors while minimizing hardware costs and system complexity, this chapter presents novel approaches based on a Wheel-IMU. First, we introduce Wheel-INS, a complete dead reckoning solution that relies solely on a Wheel-IMU while achieving similar information fusion as

ODO/INS. The installation scheme of the Wheel-IMU and the system's characteristic are also analyzed. Then, we extend Wheel-INS to Wheel-SLAM, a simultaneous localization and terrain mapping framework leveraging Wheel-IMU data. We demonstrate how terrain features can be extracted from Wheel-IMU measurements to enable loop closure detection within Wheel-INS. Finally, we present Wheel-GINS, a GNSS/INS integration system built on top of Wheel-INS. In addition, we provide detailed modeling and online estimation of Wheel-IMU installation parameters, including the Wheel-IMU lever arm, mounting angle, and wheel radius error.

4.1 Wheel-INS: Dead Reckoning with Only One Wheel-IMU

In this section, we present our study using only one Wheel-IMU for robot state estimation. In Wheel-INS, the strapdown INS is performed to predict the state of the robot. Since the Wheel-IMU rotates continuously with the wheel, the constant gyroscope bias is inherently mitigated during the INS integration process. This effect is known as rotation modulation [28], which is explained in Section 4.1.2. At the same time, the wheel velocity calculated by the gyroscope outputs and wheel radius is treated as an external observation with NHC to update the IMU state. For the sake of simplicity and efficiency, we use the error-state EKF to implement the information fusion and state estimation. The state corrections estimated by the filter are fed back to update the robot pose and compensate the IMU outputs. Figure 4.2 shows the system structure.

In sum, we make two key claims: (i) Wheel-INS achieves overall better localization accuracy compared to ODO/INS. (ii) Wheel-INS exhibits significant tolerance to constant gyroscope bias. We show experimental results to support these claims in Section 4.1.5. Additionally, we present a comprehensive discussion and experimental results on the characteristics of the Wheel-IMU, including the rotation modulation effect, misalignment errors, and design of the error state.

4.1.1 Wheel-IMU Installation

In our proposed Wheel-INS framework, only one Wheel-IMU is used, no other sensors. Unlike the conventional ODO/INS in which the IMU is placed on the vehicle body or in the trunk, in the proposed Wheel-INS, the IMU is mounted on the wheel of the robot to take advantage of the inherent rotation platform of the vehicle for rotation modulation and to obtain the wheel velocity without an odometer.

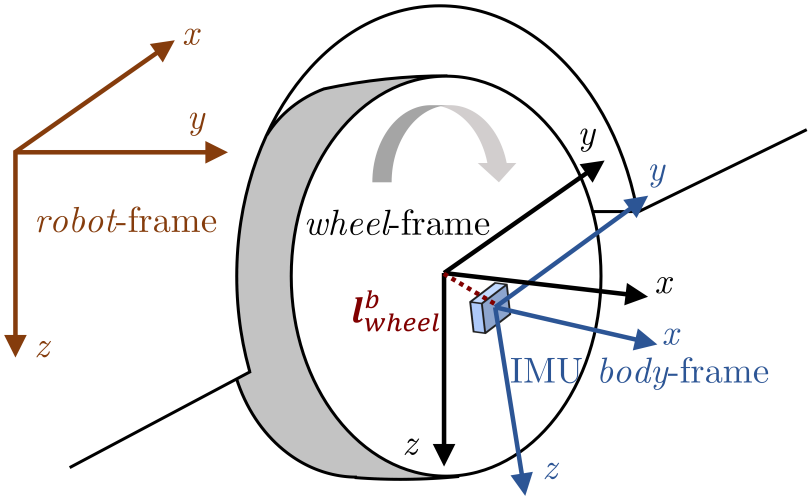


Figure 4.1: Illustration of the *robot*-frame, *wheel*-frame, and IMU *body*-frame. Details of the definition of these coordinate systems can be found in Table 3.1. $\boldsymbol{l}^b_{wheel}$ indicates the lever arm between the Wheel-IMU and the wheel center in the IMU *body*-frame; the non-parallelism of the *wheel*-frame and IMU *body*-frame indicates the mounting angles between the Wheel-IMU and the wheel.

Figure 4.1 illustrates the installation of Wheel-IMU and the definition of the involved coordinate systems. Table. 3.1 lists the details of the relevant coordinate systems. To ensure that the dead reckoning system intuitively reflects the robot's state without being influenced by steering maneuvers, it is preferable to mount the IMU on a non-steering wheel. However, it can also be mounted on a steering wheel if the steering angle is known. Consequently, the proposed Wheel-INS is not limited to differential or Ackermann steering platforms, but can also be applied to other steering mechanisms, provided that the relationship between the wheel and the robot body is known.

According to the definitions of the *robot*-frame and *wheel*-frame, given a stable vehicle structure, the difference between the wheel heading and robot heading should remain constant, namely,

$$\psi_{wheel} = \psi_{robot} + \pi/2, \tag{4.1}$$

where ψ_{wheel} and ψ_{robot} denote the wheel heading and robot heading, respectively. If the Wheel-IMU were perfectly aligned with the wheel, its heading would equal the wheel heading, and consequently, we can directly determine the robot's heading from the Wheel-IMU state. Additionally, we can compute the wheel forward velocity using the Wheel-IMU data without an odometer, i.e.,

$$v_{wheel} = \omega_x r, \tag{4.2}$$

where ω_x is the gyroscope output of the Wheel-IMU in the x-axis; r is the wheel radius. However, misalignment is inevitable in practical systems that integrate

information from multiple sensors. As shown in Fig. 4.1, the misalignment errors of Wheel-IMU include position misalignment and attitude misalignment. We explain the Wheel-IMU misalignment error in detail in Section 4.1.3. Before that, we first introduce the rotation modulation effect of the Wheel-IMU.

4.1.2 Rotation Modulation

With the movement of the vehicle, the Wheel-IMU continuously rotates around the rotation axis of the wheel, i.e., the x-axis of the IMU *body*-frame in Figure 4.1. Without loss of generality, we assume that the x-axis of the Wheel-IMU coincides with that of the w-frame. According to Equation (3.6), the attitude error differential equation can be written as

$$\begin{aligned}\mathbf{R}_b^w \delta\boldsymbol{\varepsilon}^b &= \begin{bmatrix}1 & 0 & 0\\ 0 & \cos\omega t & -\sin\omega t\\ 0 & \sin\omega t & \cos\omega t\end{bmatrix}\begin{bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\end{bmatrix}\\ &= \begin{bmatrix}\varepsilon_x\\ \varepsilon_y\cos\omega t - \varepsilon_z\sin\omega t\\ \varepsilon_y\sin\omega t + \varepsilon_z\cos\omega t\end{bmatrix},\end{aligned} \tag{4.3}$$

where ω is the IMU rotation angular rate; $\mathbf{R}_b^w$ is the rotation matrix from the IMU *body*-frame to the *world*-frame; $\delta\boldsymbol{\varepsilon}^b = \begin{bmatrix}\varepsilon_x & \varepsilon_y & \varepsilon_z\end{bmatrix}^\top$ is the constant part of the gyroscope error of the Wheel-IMU. Then the attitude error caused by the constant gyroscope errors of the Wheel-IMU in one wheel rotation period can be computed as

$$\int_{\mathrm{T}} \mathbf{R}_b^w \delta\boldsymbol{\varepsilon}^b dt = \begin{bmatrix}\int_{\mathrm{T}}\varepsilon_x dt\\ \int_{\mathrm{T}}(\varepsilon_y\cos\omega t - \varepsilon_z\sin\omega t)dt\\ \int_{\mathrm{T}}(\varepsilon_y\sin\omega t + \varepsilon_z\cos\omega t)dt\end{bmatrix}, \tag{4.4}$$

where T denotes the rotation period of the wheel. These equations show that because of the wheel rotation, the constant measurement error in the non-rotating axes of the gyroscope are modulated into sine wave in the *world*-frame. After an integral period, the accumulated pitch and heading error are canceled. The IMU velocity errors caused by the accelerometer errors can be analyzed in a similar way. More detailed explanation of the IMU rotation modulation can be found in [1, 28].

In conclusion, by rotating the IMU around a fixed axis, the measurement errors in the other two axes of the Wheel-IMU can be modulated into sinusoidal signals. Thus, the velocity errors in the right and down directions, along with the pitch and heading error caused by IMU measurement errors, can be suppressed inherently. However, if the sensing axis of the Wheel-IMU is not perfectly aligned

with the rotation axis, the effectiveness of the rotational modulation is compromised. Therefore, it is essential to compensate for misalignment errors.

4.1.3 Wheel-IMU Misalignment Errors

In Wheel-INS, the observation information is the 3D robot velocity in the *robot*-frame, including the forward velocity and NHC. Fusing this information with INS requires knowledge of the installation relationship between the robot and Wheel-IMU. However, misalignment is inevitable in any real-world system that integrates multiple sensors. If not properly addressed, it can severely degrade system performance.

As illustrated in Figure 4.1, the misalignment errors include both position misalignment (referred to as the lever arm), pointing from the Wheel-IMU center to the wheel center expressed in the IMU *body*-frame, and attitude misalignment (referred to as the mounting angles), which describe the rotation from the IMU *body*-frame to the *wheel*-frame.

The lever arm not only introduces errors in the projection of the IMU-indicated robot velocity from the IMU *body*-frame to the *robot*-frame (see Equation (4.9)), but also brings centripetal acceleration to the accelerometer measurements. This effect can cause significant error in the dead reckoning system, particularly when the robot is at high speeds. In practice, the lever arm can be measured by a digital caliper to have millimeter-level accuracy. Additionally, by knowing the models of both the wheel and the IMU, we can properly design the installation to minimize the lever arm error. In Section 4.3, we show that the lever arm can be estimated online by integrating GNSS positioning information.

The attitude misalignment, denoted as the mounting angle vector ($\boldsymbol{\varphi}_m$), can be represented by a set of Euler angles ϕ_m (roll), θ_m (pitch), and ψ_m (heading). These define the rotation from the wheel-frame to the body-frame through sequential rotations about the z-, y-, and x-axes, respectively. This misalignment weakens the effectiveness of rotational modulation and leads to an incomplete projection of the wheel's rotational angular velocity onto the gyroscope's x-axis, resulting in significant uncertainty in the computed forward velocity (see Equation (4.2)). Furthermore, Equation (4.1) remains valid only when the mounting angles are accurately compensated.

Attitude misalignment also introduces a periodic error between the estimated and true vehicle heading. Since the roll mounting angle (ϕ_m) has negligible impact on the Wheel-INS system, we focus on the pitch (θ_m) and heading (ψ_m) components. To address this, we adopt the offline calibration method proposed by [17], which estimates the attitude misalignment by analyzing Wheel-IMU data collected during straight-line motion.

In contrast, in our Wheel-GINS study (Section 4.3), we propose a novel wheel

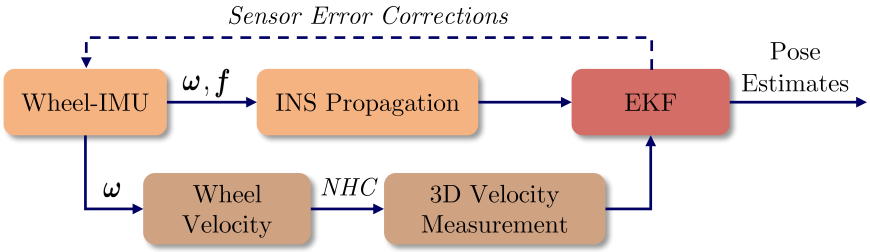


Figure 4.2: Overview of the system structure of Wheel-INS. $\boldsymbol{\omega}$ and $\boldsymbol{f}$ are the angular rate and specific force measured by Wheel-IMU, respectively; NHC refers to the non-holonomic constraints of wheeled robots.

angular velocity observation model and take advantage of the absolute positioning capability of GNSS to estimate both the position and attitude misalignment online, thereby improving system robustness and practicality.

4.1.4 Wheel-INS Pipeline

As illustrated in Figure 4.2, our proposed Wheel-INS contains four major steps. First, we perform the strapdown INS (Equation (3.2)) to propagate the robot state with the Wheel-IMU output. At the same time, we propagate the error state as the system model in an EKF (Equation (3.6)). Second, we calculate the wheel forward velocity using the gyroscope outputs and wheel radius, which is integrated with NHC to construct a 3D velocity observation. Third, we estimate the state error by fusing the observation model in our error-state EKF framework. Finally, we update the robot state and compensate the IMU outputs with the estimated state error.

The error state model is presented in Section 3.3. Here we introduce the observation model used in our Wheel-INS pipeline. Due to IMU sensor error, the wheel forward velocity calculated by the gyroscope readings of the Wheel-IMU and wheel radius also contains uncertainty, which can be written as

$$\begin{aligned} \widetilde{v}^r_{wheel} &= \widetilde{\omega}_x r - e_v = (\omega_x + \delta\omega_x) r - e_v \\ &= v^r_{wheel} + r\delta\omega_x - e_v, \end{aligned} \tag{4.5}$$

where $\widetilde{v}^r_{wheel}$ and v^r_{wheel} are the measured and true wheel forward velocity in the *robot*-frame, respectively; $\widetilde{\omega}_x$ and ω_x are the measured and true value of the angular rate in the x-axis of the Wheel-IMU; $\delta\omega_x$ is the gyroscope measurement error; r is the wheel radius, and e_v is the observation noise, modeled as Gaussian white noise. Note that the wheel radius also contains error because of the tire deformation caused by temperature and weight and pressure changes. In Wheel-INS, we don't consider the wheel radius error because there is no external absolute measurement to estimate it. In contrast, with the integration of GNSS, the wheel

radius error can be estimated and compensated for in our Wheel-GINS system (Section 4.3).

By integrating with NHC, we can obtain a 3D velocity observation which can be expressed as

$$\widetilde{\boldsymbol{v}}^r_{wheel} = \begin{bmatrix} \widetilde{v}^r_{wheel} & 0 & 0 \end{bmatrix}^\top - \boldsymbol{e}_v. \tag{4.6}$$

Since the Wheel-IMU rotates with the wheel, its roll angle relative to the *robot*-frame varies periodically. As a result, the pitch angle of the vehicle cannot be reliably determined using the Wheel-IMU alone. In other words, Wheel-INS cannot distinguish whether the vehicle is moving uphill or downhill. Therefore, it is necessary to assume that the vehicle is traveling on a horizontal surface. This assumption, however, does not introduce significant error into the dead reckoning system under typical urban conditions. For instance, if the robot travels along a road with an inclination angle of 10 °, the distance error introduced by the horizontal surface assumption can be approximated by a scale factor of $(1 - \cos(10°)) \approx 1.5\%$. From our car experiments in Section 4.1.5, we can also see that the horizontal surface assumption does not significantly affect the localization accuracy of Wheel-INS in typical urban scenarios.

According to Equation (4.1), the Euler angles of the robot with respect to the *world*-frame can be represented as

$$\boldsymbol{\varphi}^w_r = \begin{bmatrix} \phi^w_r \\ \theta^w_r \\ \psi^w_r \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \psi^w_b - \pi/2 \end{bmatrix}, \tag{4.7}$$

where $\boldsymbol{\varphi}^w_r$ is the attitude of the robot with respect to the *world*-frame; ϕ, θ, and ψ are the roll, pitch, and heading angle of the robot, respectively. The corresponding rotation matrix can be calculated by

$$\mathbf{R}^r_w = \begin{bmatrix} \cos\psi^w_r & -\sin\psi^w_r & 0 \\ \sin\psi^w_r & \cos\psi^w_r & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.8}$$

By performing the perturbation analysis, the INS-indicated wheel velocity in the *robot*-frame can be written as

$$\begin{aligned} \hat{\boldsymbol{v}}^r_{wheel} &= \hat{\mathbf{R}}^r_w \hat{\boldsymbol{v}}^w_b + \hat{\mathbf{R}}^r_w \hat{\mathbf{R}}^w_b \, (\hat{\boldsymbol{\omega}}\times) \, \boldsymbol{l}^b_{wheel} \\ &\approx \mathbf{R}^r_w (\mathbf{I} + \delta\boldsymbol{\psi}\times)(\boldsymbol{v}^w + \delta\boldsymbol{v}^w) \\ &\quad + \mathbf{R}^r_w (\mathbf{I} + \delta\boldsymbol{\psi}\times)(\mathbf{I} - \delta\boldsymbol{\phi}\times)\mathbf{R}^w_b (\boldsymbol{\omega}\times + \delta\boldsymbol{\omega}\times)\boldsymbol{l}^b_{wheel} \\ &\approx \boldsymbol{v}^r_{wheel} + \mathbf{R}^r_w \delta\boldsymbol{v}^w + \mathbf{R}^r_w \left[(\mathbf{R}^w_b (\boldsymbol{\omega}\times)\boldsymbol{l}^b_{wheel})\times \right] \boldsymbol{\phi} \\ &\quad - \mathbf{R}^r_w \left[(\boldsymbol{v}^w\times) + (\mathbf{R}^w_b (\boldsymbol{\omega}\times)\boldsymbol{l}^b_{wheel})\times \right] \delta\boldsymbol{\psi} \\ &\quad - \mathbf{R}^r_w \mathbf{R}^w_b (\boldsymbol{l}^b_{wheel}\times)\delta\boldsymbol{\omega}^b_{ib}, \end{aligned} \tag{4.9}$$

where $\mathbf{R}_w^r$ is the rotation matrix from the *world*-frame to the *robot*-frame, which can be transformed from the Euler angles $\boldsymbol{\varphi}_r^w$ by Equation (4.8); $\boldsymbol{\omega}$ is the angular velocity readings of the Wheel-IMU; $\boldsymbol{l}_{wheel}^b$ indicates the lever arm vector from the Wheel-IMU to the wheel center projected in the IMU *body*-frame; $\delta\boldsymbol{\psi}$ is the attitude error of the robot, which is only related to the heading error in the state vector; thus, it can be written as $\delta\boldsymbol{\psi} = \begin{bmatrix} 0 & 0 & \delta\psi_b^w \end{bmatrix}^\top$. Then, the velocity error observation equation in the *robot*-frame can be written as

$$\begin{aligned} \delta\boldsymbol{z}_v &= \hat{\boldsymbol{v}}_{wheel}^r - \widetilde{\boldsymbol{v}}_{wheel}^r \\ &= \mathbf{R}_w^r \delta\boldsymbol{v}^w + \mathbf{R}_w^r \left[(\mathbf{R}_b^w (\boldsymbol{\omega}\times) \boldsymbol{l}_{wheel}^b) \times \right] \boldsymbol{\phi} \\ &\quad - \mathbf{R}_w^r \left[(\boldsymbol{v}^w \times) + (\mathbf{R}_b^w (\boldsymbol{\omega}\times) \boldsymbol{l}_{wheel}^b) \times \right] \delta\boldsymbol{\psi} \\ &\quad - \mathbf{R}_w^r \mathbf{R}_b^w (\boldsymbol{l}_{wheel}^b \times) \delta\boldsymbol{\omega}. \end{aligned} \tag{4.10}$$

Although the heading error can be limited effectively by the rotation modulation, it accumulates when the vehicle remains stationary for a long time. In this case, zero-integrated heading rate measurements (ZIHR) [86] can be performed to correct the heading drift, exploiting the fact that the heading angle remains constant when the robot is stationary.

4.1.5 Experimental Results

We conducted field experiments to illustrate the performance of the proposed Wheel-INS and compare it with the conventional ODO/INS system. The experimental results support our key claims: (i) Wheel-INS achieves overall better localization accuracy compared to ODO/INS. (ii) Wheel-INS exhibits significant tolerance to constant gyroscope bias.

4.1.5.1 Experimental Setup

To illustrate the performance of the proposed Wheel-INS, we conducted field tests in two University campus, using two different wheeled vehicles. One was the Pioneer 3DX robot, a typical differential wheeled robot, and the other was a car. Figure 4.3 shows the experimental platforms.

The Wheel-IMU used in our experiments was self-developed micro-electro-mechanical system (MEMS) IMU. It includes four ICM20602 inertial sensor chips[1] (approximately 3 US dollar for each), a chargeable battery module, a micro processor, and a Bluetooth module for communication and data transmission. Since the IMU chips operate independently, we collected outputs from two of them in a single run as two separate sets of experimental data for computation and

[1] `https://invensense.tdk.com/download-pdf/icm-20602-datasheet/`

(a) Pioneer 3DX robot.

(b) Regular car.

Figure 4.3: Experimental platforms.

analysis. The Wheel-IMU was carefully mounted on the wheel to be as close as possible to the wheel center.

To compare Wheel-INS with ODO/INS, an IMU of the same type was placed on the robot body. The wheel odometer data was also recorded in the experiments. For the car, the wheel speed data came from an externally installed odometer. As shown in Figure 4.3, the two vehicles were also equipped with high-accuracy IMUs to provide a near ground truth trajectory: a tactical-grade IMU called POS320[2] (MAP Space Time Navigation Technology Co., Ltd., China) for the robot experiments and an navigation-grade IMU called LD A15[3] (Leador Spatial Information Technology Co., Ltd., China) for the car experiments. The main technique parameters of thesee IMUs are listed in Table 4.1. All reference trajectories were obtained using a smoothed post-processed kinematic (PPK)/INS integration method, which we applied to our self-collected datasets in all experiments presented throughout this thesis, except for the public datasets used in Chapter 6.

Table 4.1: Technical Parameters of the IMUs Used in the Experiments.

IMU	Gyro Bias (deg/h)	ARW* $(deg/\sqrt{h})$	Acc.*Bias (m/s^2)	VRW* $(m/s/\sqrt{h})$
LD A15	0.02	0.003	0.00015	0.03
POS320	0.5	0.05	0.00025	0.1
ICM20602	200	0.24	0.01	3

* ARW denotes the angle random walk; Acc. denotes the accelerometer; VRW denotes the velocity random walk.

We collected three sequences with our platforms. The Pioneer robot was used

[2] `http://www.whmpst.com/en/imgproduct.php?aid=32`
[3] `https://www.leador.com.cn/en/Driverless/info.aspx?itemid=1420`

Table 4.2: Vehicle Motion Information in the Experiments

Test	Sequence	Vehicle	Average speed (m/s)	Total distance (m)
1 2	I	Pioneer 3DX	1.39	≈1227
3 4	II		1.25	≈1146
5 6	III	Car	4.70	≈12199

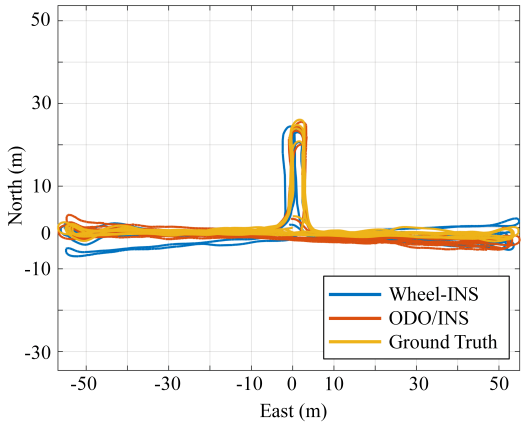
for two sequences (Seq. I and Seq. II) and the car for one sequence (Seq. III). Seq. I is a loopback trajectory in a small-scale environment, where the robot moved for approximately five times. Seq. II is a polyline trajectory with no return. Seq. III is a large loop trajectory in a University campus, where the car traveled for approximately two times. The vehicle motion information of all the three sequences is presented in Table 4.2.

We align only the initial heading of the test system with the ground truth for comparison. Additionally, the static IMU data before the vehicle started moving were used to obtain the initial roll and pitch angle, as well as the initial value of the gyroscope bias. Other inertial sensor errors were initialized as zero. The initial states of ODO/INS were determined in the same way.

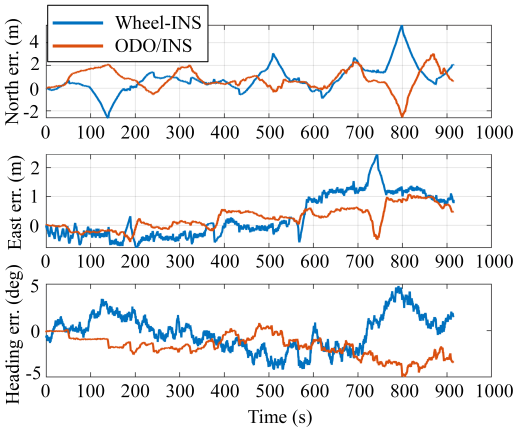
4.1.5.2 Comparison and Analysis of Localization Performance

In this section, we perform experiments to compare Wheel-INS with ODO/INS and also analyze the performance of Wheel-INS with different platforms in different environments. The results support our first key claim, that Wheel-INS achieves overall better localization accuracy compared to ODO/INS.

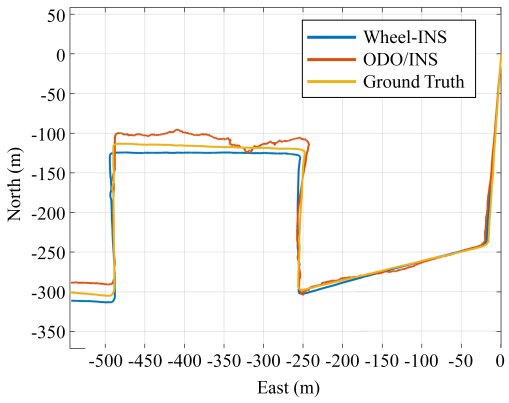
Figure 4.4 compares the trajectories and positioning and heading errors of Wheel-INS and ODO/INS in Test 1, Test 3 and Test 5, respectively. We adopted an error drift rate metric to evaluate the localization performance. First, we accumulated the moving distance of the robot by a certain increment (l). Then, we calculated the maximum horizontal position error drift rate within each distance (l, $2l$, $3l$, ...), namely, the maximum position error divided by the distance. Finally, the mean value (MEAN) and standard deviation (STD, 1σ) of these segmented drift rates were computed as the final indicator of positioning performance. This approach is similar to the odometry evaluation metric used in the KITTI dataset [31], but we segmented the trajectory only from the starting point. With regard to the heading error, the maximum (MAX) and RMSE



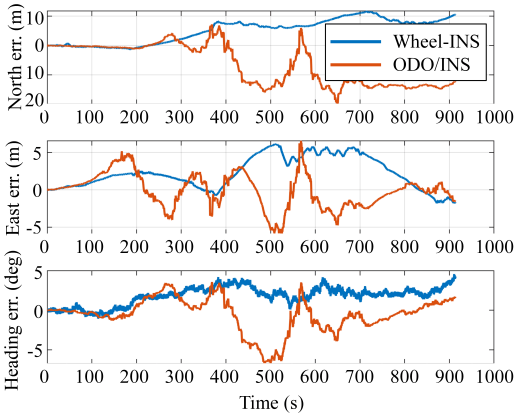
(a) Estimated trajectories against ground truth in Seq. I.



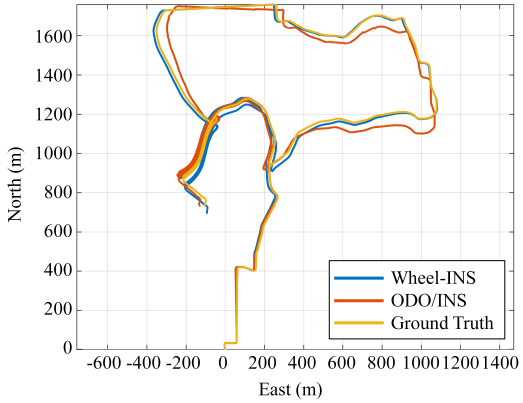
(b) Positioning and heading errors in Seq. I.



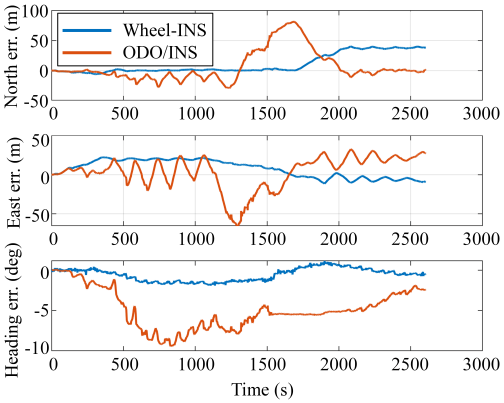
(c) Estimated trajectories against ground truth in Seq. II.



(d) Positioning and heading errors in Seq. II.



(e) Estimated trajectories against ground truth in Seq. III.



(f) Positioning and heading errors in Seq. III.

Figure 4.4: Estimated trajectories and corresponding position and heading errors of Wheel-INS and ODO/INS in three sequences.

were calculated as the evaluation criterion. In our experiments, we chose l as 100 m. Table 4.3 lists the error statistics of Wheel-INS and ODO/INS in all the six experiments.

Table 4.3: Positioning and Heading Error Statistics of Wheel-INS and ODO/INS

Test	System	Position drift rate (%)		Heading error (°)	
		MEAN	STD	RMS	MAX
1	Wheel-INS	0.59	0.30	**1.93**	4.79
	ODO/INS	**0.58**	0.45	2.11	5.11
2	Wheel-INS	1.43	0.54	3.88	7.93
	ODO/INS	**0.57**	0.32	**2.82**	4.79
3	Wheel-INS	**1.17**	0.27	**2.16**	4.56
	ODO/INS	2.25	0.68	2.88	8.00
4	Wheel-INS	**1.78**	0.26	**4.44**	10.88
	ODO/INS	2.20	0.88	5.77	9.87
5	Wheel-INS	**0.62**	0.42	**0.96**	1.91
	ODO/INS	1.14	0.65	1.38	3.30
6	Wheel-INS	**0.83**	0.43	**1.60**	4.97
	ODO/INS	1.62	1.04	2.65	6.72

From the comparison of positioning and heading accuracy between Wheel-INS and ODO/INS, we can obtain the following insights:

- Based on the mean position drift rate and heading RMSE summarized in Table 4.3, Wheel-INS improves positioning and heading accuracy by 23% and 15% on average compared to ODO/INS.
- In most experiments, Wheel-INS outperforms ODO/INS in both position and heading estimation accuracy. In Test 1, the positioning accuracy of the two systems is comparable, while Wheel-INS shows slightly better heading performance. However, in Test 2, the mean position drift rate of ODO/INS is less than half that of Wheel-INS. As previously explained, we first estimated and compensated the gyroscope bias using static IMU data before the robot began moving. Since gyroscope bias tends to remain stable over short durations (e.g., 15 minutes for Seq. I), this compensation helped reduce heading drift in ODO/INS. As a result, the advantage of Wheel-INS over ODO/INS is not significant in Test 1 and Test 2.
- In Seq. II (Test 3 and Test 4), there were many bumpy road sections. It can be observed from Figure 4.4(c) and (d) that the localization errors of ODO/INS fluctuate significantly, whereas the errors of Wheel-INS remain

comparatively smoother. In such conditions, the ground vehicle experiences strong vibrations, thereby violating the assumptions underlying the NHC. Additionally, the lever arm between the wheel center and the IMU mounted on the vehicle body is larger in ODO/INS than in Wheel-INS. This leads to more pronounced velocity projection errors during fusion with INS in ODO/INS, resulting in larger and more unstable positioning errors on uneven terrain.

- In Seq. III (Test 5 and Test 6), the experiments were conducted with a regular car. It is obvious that the accuracy of Wheel-INS in these two tests are better than that in other experiments in which the wheeled robot was used as the test platform. This improvement can be attributed to two main factors. First, the wheel and axle structure of a car is significantly more stable than that of a robot, providing a more favorable condition for Wheel-INS. Second, the car moved at a higher speed than the robot in our experiments (see Table 4.2). A relatively faster wheel speed enhances the effect of rotation modulation, thereby improving heading estimation accuracy.

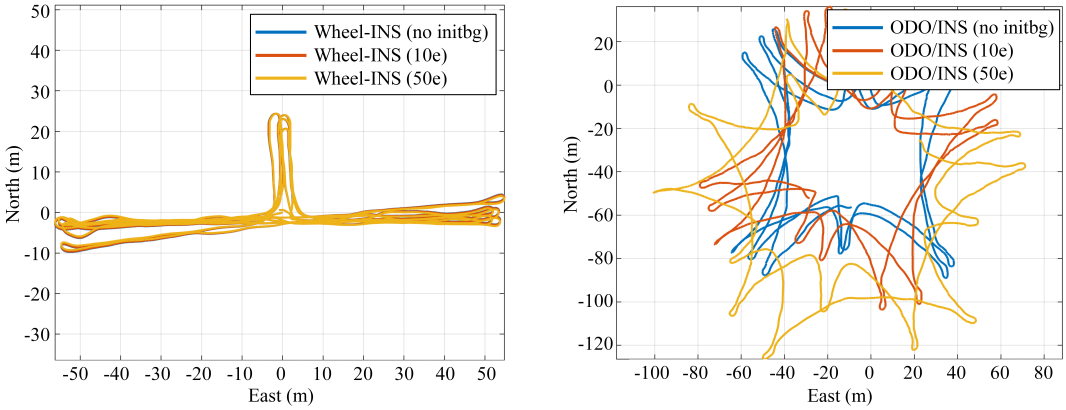
- Moreover, in Seq. III, the car moved along a large-scale trajectory with many continuous uphill and downhill roads (the maximum gradient was about 10°), but the positioning accuracy of Wheel-INS is still considerable in Test 5 and Test 6. These results indicate that although the algorithm assumes the vehicle to move on a horizontal plane, it would not introduce significant error if there are some degrees of incline in the road. That is to say, Wheel-INS can be applied to most of the regular city roads.

4.1.5.3 Comparison of Tolerance against Constant Gyroscope Bias

To illustrate the rotation modulation effect in Wheel-INS, we conducted a set of comparative experiments for Test 1 under different level of gyroscope bias. The results support our second key claim that Wheel-INS exhibits significant tolerance to constant gyroscope bias.

First, we did not estimate and compensate the initial gyroscope bias of both Wheel-INS and ODO/INS. Second, we deliberately add constant errors onto the raw gyroscope data of the Wheel-IMU and the IMU mounted on the robot body. Hereby ten times and fifty times of earth rotation angular rate ($\approx 15°/h$) were added, respectively. Then we compare the positioning performance of Wheel-INS and ODO/INS under these conditions.

As shown in Figure 4.5, the trajectories of Wheel-INS remain nearly unchanged even when the added bias reaches fifty times the Earth's rotation rate. In contrast, ODO/INS suffers from significant trajectory drift when the initial



(a) Trajectories of Wheel-INS with different gyroscope bias in Test 1.

(b) Trajectories of ODO/INS with different gyroscope bias in Test 1.

Figure 4.5: Estimated trajectories of Wheel-INS and ODO/INS given different conditions on initial gyroscope bias. "no initbg" indicates that the initial gyroscope bias was not compensated; "10e" indicates that ten times of earth rotation angular rate was manually added to the raw gyroscope data; "50e" indicates that fifty times of earth rotation angular velocity was added.

gyroscope bias is uncorrected, and the error worsens with increased bias. This is because the gyroscope bias in ODO/INS cannot be effectively estimated by integrating robot velocity measurements (i.e., forward velocity and NHC), which leads to considerable heading drift. Wheel-INS, however, is inherently robust to constant gyroscope errors due to the effect of rotational modulation. Therefore, it demonstrates significantly greater tolerance against constant gyroscope bias than ODO/INS.

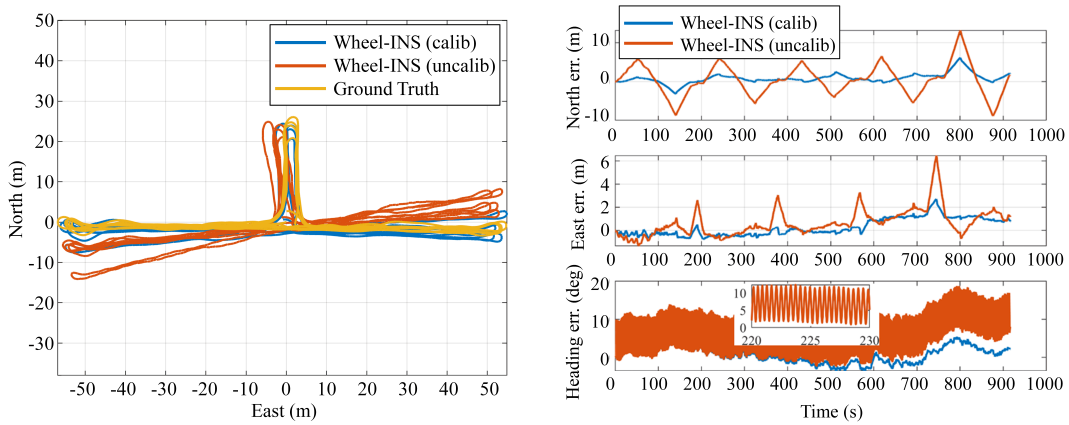
It is worth noting that although zero-velocity updates (ZUPT) [26, 49] and ZIHR [86] can help estimate gyroscope bias during vehicle stops, these signals are random and cannot be reliably used. Moreover, incorrect detection of static time frames may even degrade the stability of the system.

4.1.5.4 Discussion on the Characteristics of Wheel-INS

In this section, we present two experiments to provide deeper insights into the characteristics and design choices of the proposed Wheel-INS. The first experiment analyzes how the mounting angles of the Wheel-IMU affect the state estimation performance of Wheel-INS. The second experiment compares the system performance under different state vector configurations.

1) Influence of Mounting Angles

Figure 4.6 shows the estimated robot trajectory and the corresponding positioning and heading errors of Wheel-INS in Test 1, both before and after com-



(a) Estimated trajectories of Wheel-INS with/without mounting angle calibration against ground truth in Test 1.

(b) Positioning and heading errors of Wheel-INS with/without mounting angle calibration in Test 1.

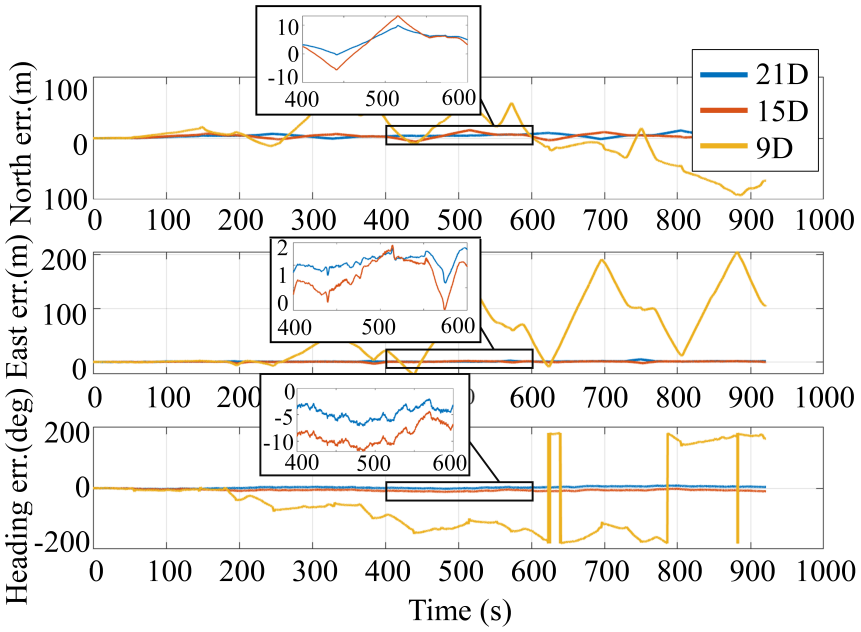
Figure 4.6: Estimated trajectories and corresponding position and heading errors under different conditions in Test 1. "calib" indicates that the Wheel-IMU mounting angles were calibrated and compensated beforehand, whereas "noncalib" indicates that the mounting angles were not handled.

pensating the Wheel-IMU mounting angles. The mounting angles were estimated using the method proposed in [17], yielding heading (ϕ_m) and pitch (θ_m) values of -4.5° and 2.5°, respectively. It is evident that the dead reckoning performance significantly deteriorates when the mounting angles are not compensated. As discussed in Section 4.1.3, the reason is that the mounting angles weaken the rotation modulation effect and invalidate Equation (4.1) and Equation (4.7), thereby introducing errors when fusing INS with the wheel velocity and NHC. Furthermore, as shown in Figure 4.6(b), when the mounting angles are not compensated, a sinusoidal signal is modulated onto the heading estimation due to the misalignment between the heading axis (x-axis) of the Wheel-IMU and the wheel's rotation axis.

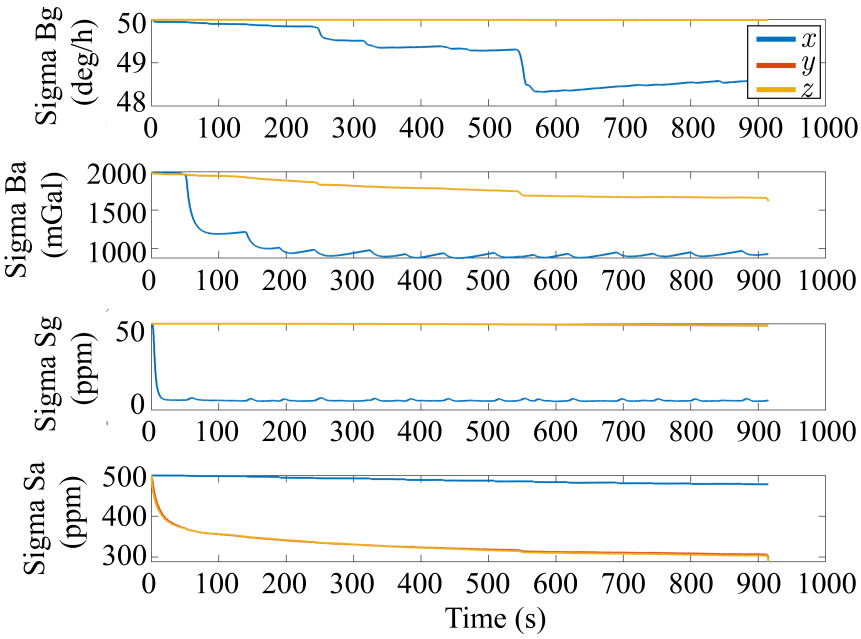
In conclusion, calibrating and compensating for the attitude misalignment of Wheel-IMU is crucial for fully leveraging the rotation modulation and improving Wheel-INS performance. In this study, we calibrate it offline, while in our Wheel-GINS study (Section 4.3), we propose an approach to estimate the mounting angles online to improve the practicality of the system.

2) Ablation Study on Different State Dimension

In our proposed Wheel-INS, we adopted a 21-state error-state EKF. We estimate not only the robot state error but also the inertial sensor error, including bias errors and scale factor errors for both the gyroscope and accelerometer. In this section, we perform an ablation study to highlight the necessity of the 21-state design.



(a) Comparison of the dead reckoning errors of Wheel-INS with different state dimensions in Test 1. "21D", "15D", and "9D" indicate the 21-, 15-, and 9-state, respectively.



(b) STD of the estimated residual inertial sensor errors in the 21-state Wheel-INS in Test 1. "ppm" means parts per million.

Figure 4.7: Comparison of the dead reckoning errors of Wheel-INS with different state dimensions and STD of the estimated residual inertial sensor errors in the 21-state Wheel-INS in Test 1.

To compare their positioning performance, we used state vectors of 21, 15, and 9 dimensions in Wheel-INS during Test 1. The 21-state vector is detailed in Equation (3.11). The 15-state vector excludes the scale factor error of the gyroscope and accelerometer, while the 9-state vector omits all IMU error terms. The corresponding state estimation errors for these three experiments are shown in Figure 4.7.

As shown in Figure 4.7, the positioning error increases significantly when the state vector dimension is reduced to 9, and the 15-state Wheel-INS also performs worse than the 21-state version. While the constant sensor bias of the IMU along the axis perpendicular to the rotation axis can be somewhat canceled, other error components still cause substantial errors if not estimated and compensated. For instance, cross-coupling errors have a more significant negative impact in Wheel-INS due to the much higher dynamic conditions of the Wheel-IMU compared to the IMU mounted on the vehicle body in traditional ODO/INS.

Figure 4.7(b) depicts the STD of the inertial sensor errors estimation of the 21-state Wheel-INS in Test 1. We selected Test 1 because it is a typical trajectory in a common environment. The Wheel-IMU errors in the y- and z-axes are mixed because they are parallel to the wheel plane and alternatively change their directions, making it difficult for the filter to distinguish them. This effect explains why the STD of the sensor error estimation in the y- and z-axes are almost coincident, as shown in all the subplots in Figure 4.7(b).

Regarding the scale factor error, since the Wheel-IMU rotates around the x-axis, the large rotation angular rate makes the gyroscope's scale factor error along the x-axis observable, allowing for quicker convergence in the filter system. Additionally, because the accelerometers in the y- and z-axes alternately perceive gravity and the vehicle is assumed to move on a horizontal plane, Wheel-INS can also estimate the scale factor errors of the accelerometer in these axes.

4.1.6 Conclusion

We propose a complete dead reckoning solution for wheeled robots using a low-cost Wheel-IMU. The key objectives of the system are: 1) to leverage the inherent rotation platform of the wheeled robot to eliminate heading drift caused by constant gyroscope bias; and 2) to use only one IMU while achieving an information fusion scheme similar to ODO/INS.

Field tests in different environments with different robots demonstrate that the maximum horizontal position drift of Wheel-INS is around 1% of the total traveled distance. The positioning and heading accuracy of Wheel-INS have respectively improved by 23% and 15% against ODO/INS. Moreover, benefit from the rotation modulation, Wheel-INS illustrates significant resilience to the gyroscope bias compared to the conventional ODO/INS.

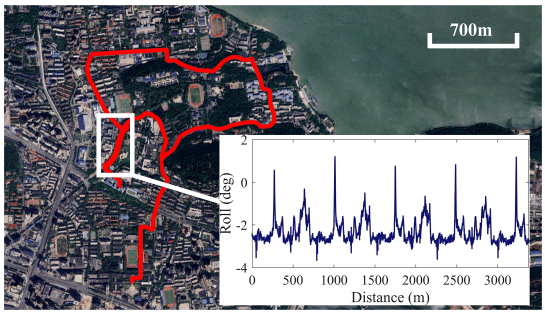
The characteristics of Wheel-INS have also been analyzed and discussed. The misalignment errors of Wheel-IMU are defined and explained; these must be compensated in advance to obtain more accurate state estimates. Additionally, the observability of the inertial sensor errors are analyzed by their covariance propagated in the EKF. Because the residual sensor errors of the Wheel-IMU in the directions perpendicular to the rotation axis are coupled to each other, they cannot be distinguished by the EKF. Experimental results show that it is necessary to incorporate the bias errors and scale factor errors of both the gyroscope and accelerometer in the error state vector of Wheel-INS to achieve better performance.

4.2 Wheel-SLAM: SLAM with Only One Wheel-IMU

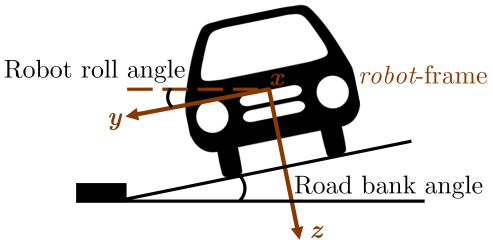
In previous section, we have discussed our proposed Wheel-INS system, which improves the accuracy compared to the conventional ODO/INS system while using only one Wheel-IMU. Although Wheel-INS exhibits excellent dead reckoning performance, it is only a relative positioning solution lacking the ability to limit error accumulation over long time. That is to say, the positioning error of Wheel-INS still drifts, especially when the stochastic error of the inertial sensor is significant. To correct the accumulated error without depending on absolute positioning techniques, a commonly used method is loop detection and relocalization.

To achieve this, an exteroceptive sensor, e.g., camera and LiDAR is typically required to extract the environmental features and build a map simultaneously. These features must be obvious and repeatable to ensure successful loop closure. From the experiments of Wheel-INS, we found that the robot's roll estimation provided distinguishable and repeatable terrain-correlated responses, as shown in Figure 4.8(a). Additionally, the robot's roll angle, to some extent, reflects the road bank angle, as shown in Figure 4.8(b). Therefore, it is feasible to encode the robot roll angle as a kind of terrain feature to enable the loo closure detection in Wheel-INS. In addition, because the IMU is mounted on the wheel which is directly contacted with the ground, the robot roll angle estimated by Wheel-INS can represent the road bank angle precisely without being affected by the suspension system of the robot which could cause error to the body-mounted IMU.

In this section, we present Wheel-SLAM, a novel SLAM system that only uses a Wheel-IMU to extract terrain features for loop closure detection. To the best of our knowledge, this is the first SLAM systems that uses only a low-cost IMU in



(a) Robot trajectory and roll angle estimation in the car experiment of Wheel-INS.



(b) Illustration of the relationship between the robot roll angle and road bank angle.

Figure 4.8: Illustration of the robot roll angle estimation in Wheel-INS and the relationship between the robot roll angle and road bank angle. In the marked area of subfigure (a), the car kept circling back and forth five times. We can see that the robot roll angle estimation (indicating the road bank angle) shows a distinguishable and repeating pattern that can be exploited to perform loop closure detection. In subfigure (b), we can see that the robot roll angle estimates reflect the road bank angle.

the literature. Specifically, Wheel-SLAM uses Wheel-INS to predict the robot's state. At the same time, we build a 2D grid terrain map with one roll angle feature in each grid. We adopt a particle filter (PF) to represent the uncertainty of the robot pose. Each particle maintain their own trajectories and terrain maps. Once a particle (or more particles) reports that the robot enters into a previously-visited grid, we compare the current roll angle sequence with the roll angle sequence stored in the map to detect the loop closure. If a loop closure is detected by a particle, we update the weights of all the particles according to the similarity between the current roll angle sequence and the stored one.

In sum, we make two key claims: (i) Wheel-SLAM can effectively detect loop closure by extracting the road bank angle features. (ii) Wheel-SLAM significantly constrains the position error drift in Wheel-INS. We show experimental results to support these claims in Section 4.2.4. We further conduct an ablation study on the number of particles to gain deeper insight into the characteristics of the proposed Wheel-SLAM system.

4.2.1 Dynamic Bayesian Network

A PF is a sequential Monte Carlo method where the basic idea is the recursive computation of relevant probability distributions using the concepts of importance sampling and approximation of probability distributions with discrete random measures [27]. In PF, the posterior distribution of the robot state is represented by a set of particles that evolve recursively with the integration of new information. Based on the technique of Rao-Blackwellization [35, 69, 70],

Wheel-SLAM decomposes the SLAM problem into a robot localization problem and a terrain mapping problem that is conditioned on the robot pose estimate.

In Wheel-SLAM, we try to estimate the posterior

$$p(\boldsymbol{x}_{1:t}, \mathbf{M} | \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) \tag{4.11}$$

which is the distribution representing the robot state $\boldsymbol{x}_{1:t}$ and the terrain map $\mathbf{M}$ based on the set of control inputs $\boldsymbol{u}_{1:t}$ which governs the robot motion and the road bank angle observations $\boldsymbol{z}_{1:t}$. The conditional independence property of the Wheel-SLAM problem implies that the posterior in Equation (4.11) can be factored as follows:

$$p(\boldsymbol{x}_{1:t}, \mathbf{M} | \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) = p(\boldsymbol{x}_{1:t} | \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) \prod_{i=1}^{N_f} p(m_i | \boldsymbol{z}_{1:t}, \boldsymbol{x}_{1:t}) \tag{4.12}$$

where m_i is the i-th terrain feature, and N_f is the total number of the features. In Wheel-SLAM, Wheel-INS is performed for the robot state estimation as well as the road bank angle (terrain feature) perception.

Wheel-SLAM uses a PF to estimate the robot trajectory distribution. For each particle, the individual terrain map is independent of each other. As a result, every particle is composed of a robot pose and a terrain map; thus, the i-th particle at time t can be represented as

$$\boldsymbol{\mathcal{X}}_i^t = \begin{bmatrix} \boldsymbol{x}_i^t & \mathbf{M}_i \end{bmatrix} \tag{4.13}$$

where $i = 1, 2, 3, ..., N_p$, is the index of the particle while N_p is the total number of the used particles. $\boldsymbol{x}_i^t$ is the pose of the robot estimated by the i-th particle at time t, and $\mathbf{M}_i$ is the terrain map maintained by the i-th particle. Here the robot pose is represented by 2D translation $p_i^t \in \mathbb{R}^2$ and heading $\mathbf{R}_i^t \in SO(2)$ as Wheel-INS cannot estimate pitch angle, namely, the robot motion in the vertical direction.

Our Wheel-SLAM algorithm consists of four main steps: 1) Sample new robot state by the motion predicted by Wheel-INS; 2) Update the terrain map and detect loop closure; 3) Update the particle weights once a convinced loop closure is reported; 4) Resample the particles when it is necessary. Algorithm 1 shows an overview of Wheel-SLAM.

The first step is to generate a new pose for each particle by sampling from the robot probabilistic motion model:

$$p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) \tag{4.14}$$

which means that the robot pose, $\boldsymbol{x}_t$, is a probabilistic function of the robot control $\boldsymbol{u}_t$ and the previous pose $\boldsymbol{x}_{t-1}$. Here, the robot state is predicted by Wheel-INS. The states of the particles are sampled from the predicted state using a normal Gaussian distribution.

Algorithm 1: Wheel-SLAM

Input: Robot pose $\boldsymbol{x}_{t-1}$, terrain map $\mathbf{M}_{t-1}$, control input $\boldsymbol{u}_t$ and measurement $\boldsymbol{z}_t$.

Output: Robot pose $\boldsymbol{x}_t$, terrain map, $\mathbf{M}_t$.

foreach $i \in \{1, \ldots, N_p\}$ **do**
  Predict the robot pose by the motion model $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ Update the terrain map $p(\mathbf{M}_t|\boldsymbol{x}_t, \boldsymbol{z}_t)$ if necessary **if** *Loop closure detected* **then**
    Update the weight of the particles according to Equation (4.15)
  end
end
Normalize the weights of the particles $\omega^i = \omega^i / \sum_{i=1}^{N_p} \omega^i$
if $1/\sum_{i=1}^{N_p} (\omega^i)^2 < 0.75$ **then**
  Perform resampling
end

4.2.2 Grid Terrain Map Construction

Along with the robot's state evolution, a grid map is built to store the measured road bank angle features. Compared to the hexagon grid map used in FootSLAM [2], we simplify the grid to square due to the relatively simple robot motion mode on the roads. As we assume that the robot is moving on the horizontal plane, our grid map is in 2D. Each grid stores a sequence of roll angles covering a short segment of the past trajectory up to the current position, sampled at equal distance intervals to eliminate the influence of the robot's speed. This scheme may introduce redundant information in the map, but it facilitates roll sequence queries and matching. Additional angle encoding can be added into the roll sequence to represent the roll sequences in different direction. Figure 4.9 illustrates the grid map constructed along with the robot trajectory evolution.

4.2.3 Loop Closure Detection and Particle Weight Update

As the robot state and map evolve simultaneously, loop closures are detected by matching road bank angle features, after which the particle weights are updated accordingly. Initially, all particles are assigned equal weights. As the robot moves, each particle develops a different trajectory and terrain map. Once a particle (or multiple particles) re-enters previously visited grids, feature matching is performed to verify true loop closures. To improve robustness, we compare

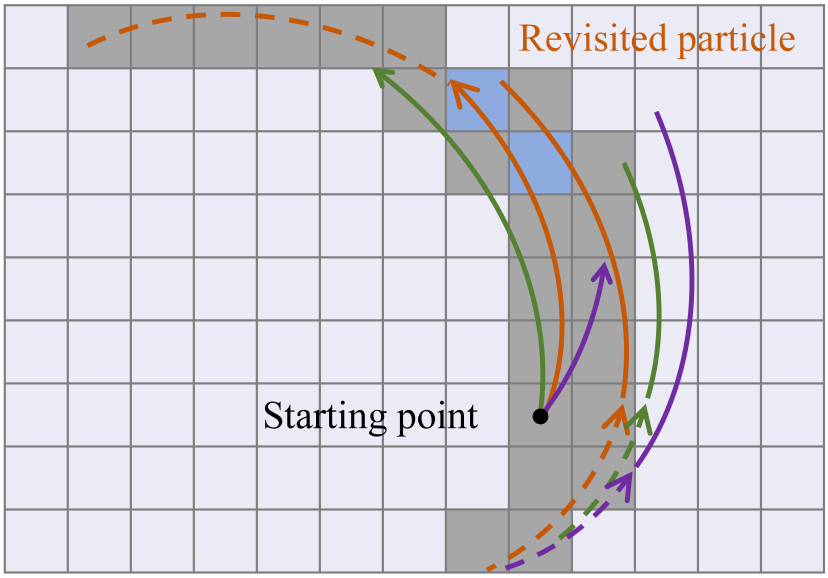


Figure 4.9: Illustration of the grid map construction and revisit recognition. Different colors of curves represent the robot path sampled by different particles. The gray grids have been visited by the robot thus they have a road bank angle measurement. Once a particle detects that the robot has continuously returned to visited grids (blue), a potential loop closure is reported for further check.

the current road bank angle sequence with the stored sequence in the grid map using the Pearson correlation coefficient [67], rather than matching a single angle. Finally, to ensure reliable loop closure detection and reduce the impact of outliers, three criteria must be satisfied:

1. The particle must consecutively revisit previously visited grids across a sliding window of length N_r.

2. For each revisit in the window, the Pearson correlation coefficient between the current and stored road bank angle sequences is computed; at least N_{thr} ($N_{\text{thr}} \leq N_r$) of these coefficients must exceed a threshold C_{thr}.

3. The correlation coefficient at the current position must also exceed C_{thr}.

If all three criteria are satisfied, a true loop closure is detected, and the particle weights are updated as follows:

$$\omega_k^i = \omega_{k-1}^i \frac{N_c}{N_r} \exp(\text{RMSE}(\boldsymbol{v}_{\text{coeff}})) \tag{4.15}$$

where k indicates the time epoch; i indicates the number of the particle; N_c ($N_c \geq N_{\text{thr}}$) is the number of the correlation coefficients larger than the threshold in the window N_r; RMSE indicates the root mean square error; $\boldsymbol{v}_{\text{coeff}}$ is the collection of the correlation coefficients larger than the threshold in the sliding

window. After that, normalization is performed to make the sum of the weights equal to one.

4.2.4 Experimental Results

4.2.4.1 Experimental Setup

To illustrate the feasibility and effectiveness of the proposed Wheel-SLAM system, we carried out five field tests using the same car used in our previous Wheel-INS experiments on a university campus. The motion characteristics of the robot during these experiments are summarized in Table 4.4.

Table 4.4: Robot motion information in the experiments

Sequence	1	2	3	4	5
Average speed (m/s)	5.41	5.47	4.93	5.16	4.89
Total distance (m)	≈2950	≈3791	≈2802	≈7235	≈9285

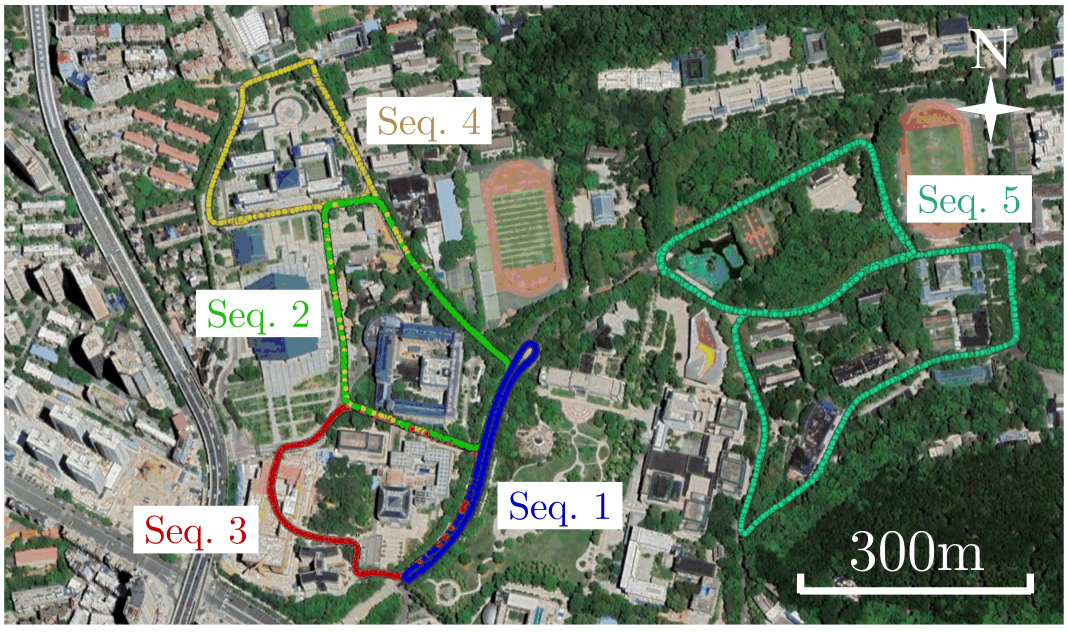


Figure 4.10: Experimental trajectories. Seq. 1, Seq. 2, and Seq. 3 are circular trajectories where the robot moved several times in one direction on the same road, while Seq. 4 and Seq. 5 are more complicated ones where the robot moved in different directions (not just turn around in one direction) in larger-scale environments.

We used the same MEMS IMU used in Wheel-INS in our Wheel-SLAM experiments. We also performed a smoothed PPK/INS integration method to get the ground truth trajectory with a high-end IMU. The static IMU data before

the car started moving were used to obtain the initial roll and pitch angle of the Wheel-IMU, as well as the initial value of the gyroscope bias. Other inertial sensor errors were set as zero.

Figure 4.10 shows the five experimental trajectories. In Sequence (Seq.) 1, the car traveled back and forth along two parallel, opposite-direction roads. Seq. 2 and Seq. 3 feature circular trajectories where the vehicle completed multiple loops in the same direction. Seq. 4 and Seq. 5 are more complex, involving two circular paths in large-scale environments. In these two sequences, the car not only traveled in the same lane and direction but also moved in opposite directions on the same road. It is also worth noting that in Seq. 1, Seq. 2, and Seq. 3, the car occasionally changed lanes during the experiments.

The key parameters of Wheel-SLAM set in the experiments are listed in Table 4.5.

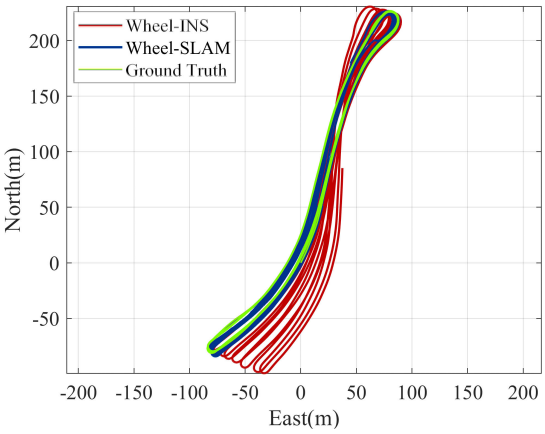
Table 4.5: Key parameters of Wheel-SLAM in the experiments.

Parameters	Value
Particle number N_p	100
Grid size of the 2D terrain map	1.5 m
Sliding window length N_r	10
Roll angle sample distance interval	0.5 m
Roll angle sequence length	50
Correlation coefficient threshold C_{thr}	0.4

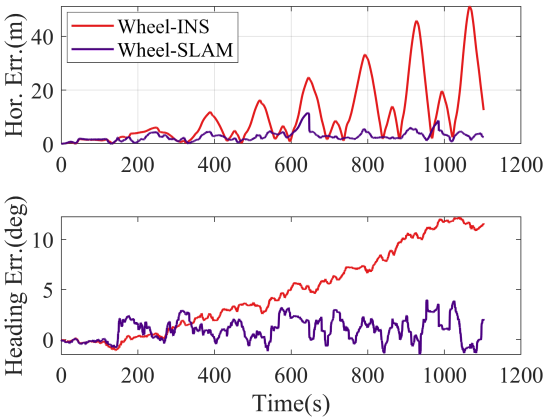
4.2.4.2 Evaluation on State Estimation Performance

Figure 4.11 compares the positioning and heading errors of Wheel-SLAM and Wheel-INS in Seq. 1, Seq. 2, and Seq. 4, respectively. Compared to Wheel-INS, Wheel-SLAM effectively suppresses error drift, even in complex scenarios where the car moves in different directions across large-scale environments, such as in Seq. 4 and Seq. 5. In contrast, Wheel-INS exhibits continuously increasing positioning errors accompanied by heading drift.

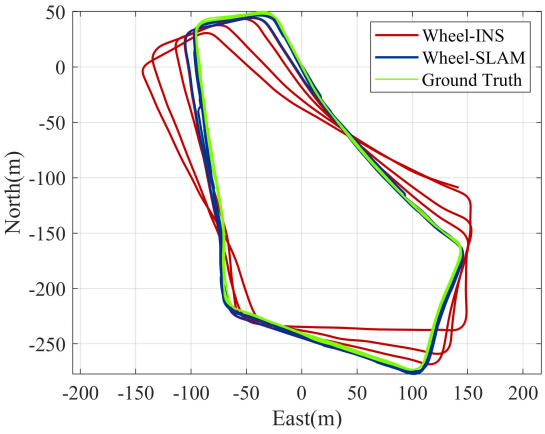
At the beginning of the trajectories, Wheel-SLAM and Wheel-INS exhibit similar drift (e.g., the first 150 s in Figure 4.11(d)) because all particles are initially sampled from the Wheel-INS result with Gaussian noise, and no external measurement is available to correct position errors until a loop closure is detected. Once loop closures are reported by some particles, their weights increase, pulling the estimated robot position back toward the previous trajectory. This correction results in sharp decreases in positioning and heading errors, such as at



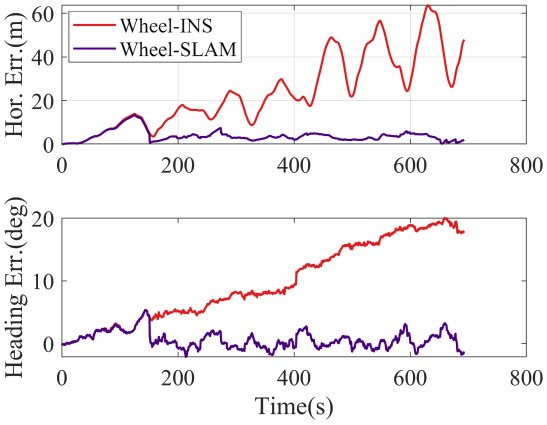
(a) Estimated trajectories against ground truth in Seq. 1.



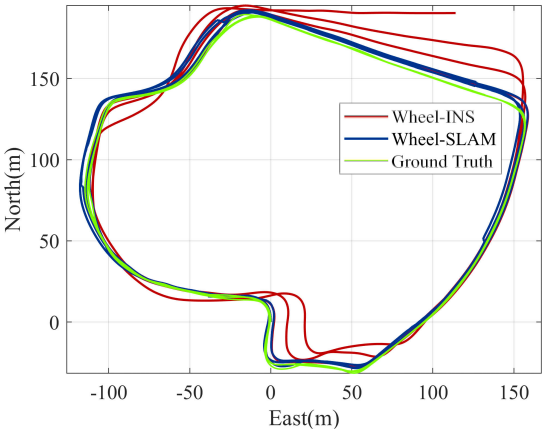
(b) Horizontal positioning and heading errors in Seq. 1.



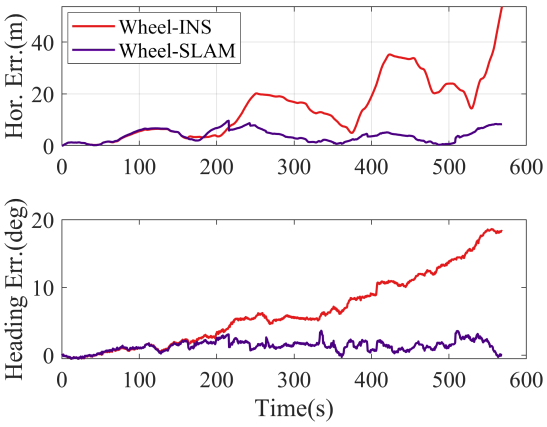
(c) Estimated trajectories against ground truth in Seq. 2.



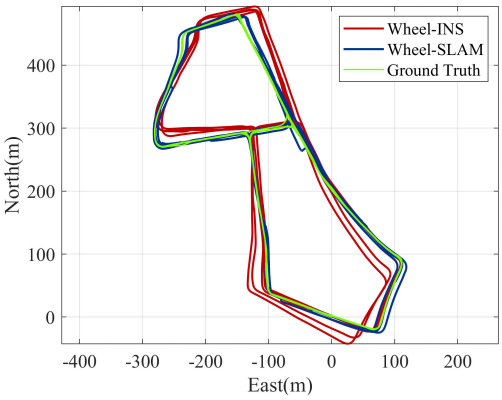
(d) Horizontal positioning and heading errors in Seq. 2.



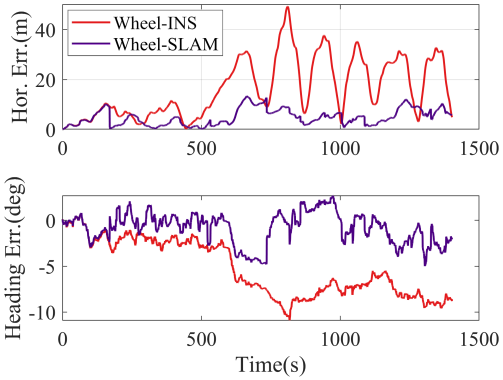
(e) Estimated trajectories against ground truth in Seq. 3.



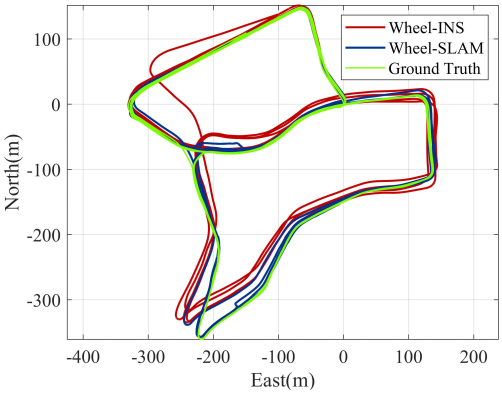
(f) Horizontal positioning and heading errors in Seq. 3.



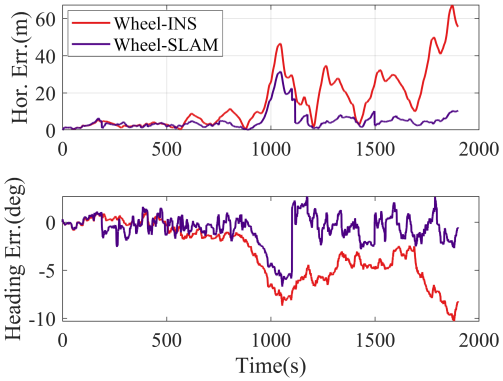
(g) Estimated trajectories against ground truth in Seq. 4.



(h) Horizontal positioning and heading errors in Seq. 4.



(i) Estimated trajectories against ground truth in Seq. 5.



(j) Horizontal positioning and heading errors in Seq. 5.

Figure 4.11: The estimated trajectories and corresponding horizontal position and heading errors of Wheel-INS and Wheel-SLAM in five experiments.

150 s in Figure 4.11(d) and 1000 s in Figure 4.11(j). Notably, we only correct the current robot state without optimizing the historical trajectory. More advanced optimization, such as applying graph-based methods like gtsam [24], could further smooth the trajectory and improve accuracy. However, this work primarily aims to demonstrate the feasibility of extracting road features from Wheel-IMU measurements for loop closure.

Taking Seq. 2 as an example, the figures show that the positioning and heading errors of Wheel-INS increase rapidly over time, while those of Wheel-SLAM remain bounded. During the first circle, before any loop closure occurs, Wheel-SLAM exhibits similar drift as Wheel-INS. However, as the robot revisits previously traveled paths, loop closures are detected, and position updates effectively constrain the accumulated drift, demonstrating the capability of the proposed method to mitigate error growth in Wheel-INS by leveraging terrain information.

Table 4.6 summarizes the state estimation error statistics of Wheel-SLAM and Wheel-INS across all five sequences. We compute the RMSE of the horizontal

Table 4.6: Positioning and heading error of Wheel-SLAM and Wheel-INS

Seq.	Horizontal Pos. RMSE(m)		Heading RMSE (°)	
	Wheel-INS	Wheel-SLAM	Wheel-INS	Wheel-SLAM
1	5.70	**2.50**	1.96	**1.00**
2	27.09	**9.38**	11.36	**3.17**
3	18.03	**8.27**	8.32	**3.83**
4	20.24	**9.21**	6.09	**2.43**
5	21.44	**14.42**	4.26	**2.95**

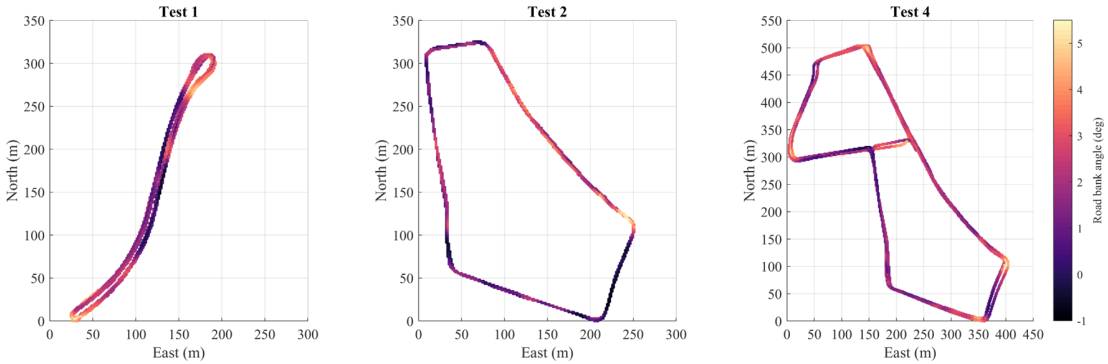


Figure 4.12: The terrain maps estimated by Wheel-SLAM in Seq. 1, Seq. 2, and Seq. 4, respectively. The colors represent the values of the road bank angles. The larger the road bank angle, the lighter the color.

position and heading (denoted as Horizontal Pos. RMSE and Heading RMSE in Table 4.6) to evaluate the estimation performance. Considering the randomness in particle sampling, the algorithm was run 100 times, and the mean value of each test was reported. The results demonstrate that, thanks to loop closure detection, Wheel-SLAM consistently outperforms Wheel-INS in both position and heading estimation. On average, Wheel-SLAM achieves a 52.6% improvement in positioning accuracy and a 53.2% improvement in heading accuracy compared to Wheel-INS.

In conclusion, the experimental results support our key claims: (i) Wheel-SLAM can effectively detect loop closure by extracting the road bank angle features. (ii) Wheel-SLAM significantly constrains the position error drift in Wheel-INS.

We present the terrain maps constructed by Wheel-SLAM in Seq. 1, Seq. 2, and Seq. 4, as shown in Figure 4.12. Since the wheel is in direct contact with the ground, the mapping is not influenced by the robot's suspension system, even under large maneuvers—unlike the case when the IMU is mounted on the robot body. Moreover, the generated terrain maps can serve as valuable resources for

monitoring road deformation and deterioration over time.

4.2.4.3 Ablation Study on Particle Number

To further evaluate the performance and stability of Wheel-SLAM, we tested different numbers of particles in Seq. 1. For each configuration, the algorithm was run 100 times. Figure 4.13 presents the results.

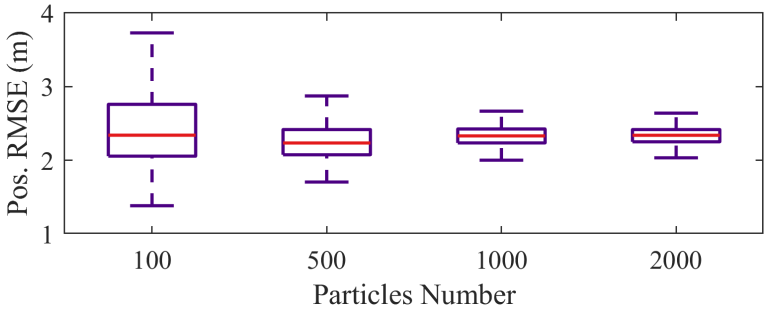


Figure 4.13: The positioning RMSE of Wheel-SLAM with different particle numbers in Seq. 1. Medians are indicated by the red lines, while the bottom and top edges of the boxes indicate the first quartile and third quartile, respectively, and the whiskers show the maximum (upper) and minimum (lower).

It can be observed that as the number of particles increases, the position errors of Wheel-SLAM become more concentrated, indicating improved stability. This is because with more particles, the system has a higher probability of detecting true loop closures and becomes less sensitive to anomalies. However, increasing the particle number from 1000 to 2000 yields little additional benefit due to diminishing marginal returns. Moreover, the accuracy of Wheel-SLAM also depends on the initial position error when the robot revisits a location, meaning that simply adding more particles can only marginally improve positioning accuracy.

Additionally, the overall positioning performance does not significantly improve with a higher number of particles. Although a few outliers appear when the particle number is small (e.g., 100), the system remains robust in recognizing loop closures, thanks to the strong dead reckoning capability of Wheel-INS.

4.2.4.4 Discussion

The core principles behind Wheel-SLAM can be summarized as follows:

1. Particles are spread to sample the possible robot states and detect loop closures based on each particle's maintained trajectory and grid terrain map;

2. Road bank angle sequence matching results are used to update particle weights and identify the most reliable particle(s).

The roll sequence matching strategy plays a central role in Wheel-SLAM. It must be robust enough to retain outstanding particles while filtering out false positives. Therefore, we adopt strict loop closure detection criteria to enhance robustness.

However, Wheel-SLAM also has two major limitations. First, the robot must physically revisit previously traveled areas over a sufficient length; unlike vision-based SLAM systems that leverage exteroceptive sensors (e.g., cameras or LiDARs) for remote feature sensing, Wheel-SLAM relies on terrain features extracted by the Wheel-IMU, which can only be obtained upon actual contact. Second, successful loop closure depends on the variation of the road bank angle sequence. If the robot operates on extremely smooth roads with negligible bank angle fluctuations—for instance, in indoor environments—loop closure detection becomes challenging for Wheel-SLAM.

4.2.5 Conclusion

We propose a SLAM system using only one Wheel-IMU by exploiting the environmental perception ability of the Wheel-IMU. To be specific, we extend our previous Wheel-INS system to Wheel-SLAM by extracting the terrain feature from the robot roll angle estimates to enable loop closure detection. The system is implemented with a Rao-Blackwellized particle filter where each particle maintains its own robot state and grid map. Experimental results show that the proposed method can effectively suppress the error drift of Wheel-INS by closing loops. The positioning and heading accuracy has been averagely improved by 52.6% and 53.2%, respectively, against Wheel-INS.

Given the characteristics of Wheel-SLAM, it is especially suitable for robots that perform repeated operations within predefined areas, such as sweeping robots or patrol robots operating in restricted environments.

4.3 Wheel-GINS: GNSS/Wheel-IMU Integrated Localization System

In previous two sections, we have introduced our proposed dead reckoning solution (Wheel-INS) and SLAM solution (Wheel-SLAM) using only one low-cost Wheel-IMU for wheeled robot. Taking advantage of the inherent rotation modulation platform of the wheeled roobot, Wheel-INS improved the robustness and accuracy in state estimation compared to ODO/INS using only a low-cost Wheel-IMU.

Building on top of Wheel-INS, Wheel-SLAM can effectively detect loop closure by extracting and matching terrain features to significantly suppress the error drift in Wheel-INS.

However, Wheel-SLAM is only applicable for those applications where the robot moves in constrained environments while failing in scenarios where the robot does not have the opportunity to revisit the places it has been before, for example, a self-driving car driving from one city to another. Therefore, external correction signals are necessary to limit the long-term error drift of Wheel-INS.

GNSS has been a widely used aid to integrate with INS for mobile robot navigation because of the complementary advantages of GNSS and INS. On the one hand, GNSS can provide accurate absolute positions to correct the error drift of INS. On the other hand, INS can provide continuous state estimation when GNSS is not available. In this section, we present Wheel-GINS, a novel GNSS/INS integration navigation system using a Wheel-IMU, along with a study of its characteristics and an evaluation of its performance. To the best of our knowledge, this is the first GNSS/INS integrated navigation system in the literature that uses a low-cost Wheel-IMU.

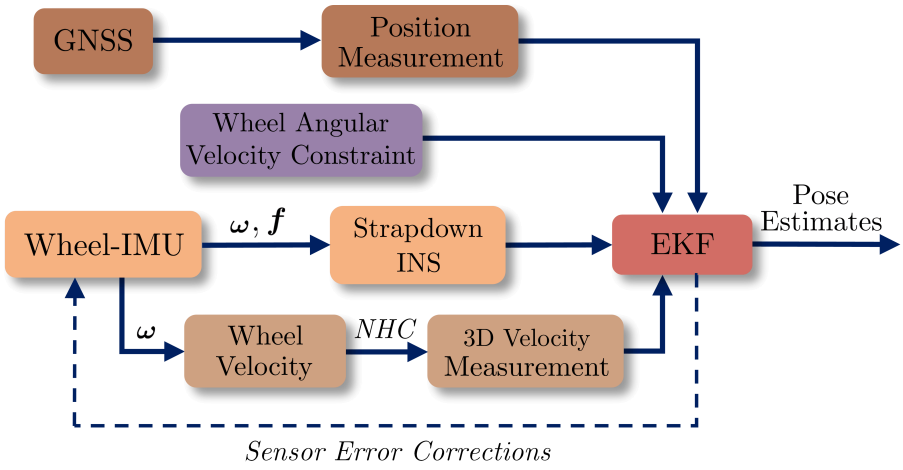


Figure 4.14: System overview of Wheel-GINS. $\boldsymbol{\omega}$ and $\boldsymbol{f}$ are the angular velocity and specific force measured by the Wheel-IMU, respectively.

Specifically, we integrate the GNSS position measurement into the EKF pipeline building on top of Wheel-INS; thus, Wheel-GINS can achieve a similar information fusion scheme as the conventional GNSS/Odometer/INS integrated navigation system (ODO-GINS). However, Wheel-GINS is more promising than ODO-GINS due to the advantages of Wheel-INS. Additionally, the misalignment between the GNSS and the Wheel-IMU is different from that between the GNSS and the normal IMU, which is usually placed on the top or in the body of the robot. To handle this issue and improve the practicality of the proposed Wheel-GINS, we perform detailed modeling of the Wheel-IMU installation parameters,

including the Wheel-IMU lever arm (position misalignment) and mounting angle (attitude misalignment), and the wheel radius scale error to estimate them online. Furthermore, we propose a wheel angular velocity constraint model to accelerate the convergence of the Wheel-IMU mounting angle online estimation. Figure 4.14 overviews the algorithm structure of the proposed Wheel-GINS.

In sum, we make two key claims: (i) Wheel-GINS has significantly reduced the position error drift compared to ODOGINS during GNSS outages; (ii) Wheel-GINS can effectively estimate the Wheel-IMU installation parameters, including the Wheel-IMU lever arm and mounting angle and the wheel radius scale error online, thus improving the pose estimation accuracy. Our experimental evaluation backs up these claims.

4.3.1 Misalignment Parameters

In Section 4.1.3, we have discussed the Wheel-IMU misalignment parameters, including the Wheel-IMU lever arm and mounting angle, as shown in Figure 4.1. The lever arm is the vector from the Wheel-IMU center to the wheel center expressed in the Wheel-IMU *body*-frame, while the mounting angle is the rotation angle of the Wheel-IMU *body*-frame with respect to the *wheel*-frame. In our previous Wheel-INS study, we measured the lever arm manually and calibrate the mounting angle offline. However, this approach is not practical for real-world applications. Therefore, we propose to estimate the Wheel-IMU installation parameters online in Wheel-GINS. In addition to the Wheel-IMU lever arm and mounting angle, we also estimate the wheel radius scale error online. The wheel radius scale error is the ratio of the actual wheel radius to the nominal wheel radius. The nominal wheel radius is the value we measured beforehand, which differs from the actual value because the vehicle weight, temperature, and tire pressure would all cause wheel deformation. It is necessary to compensate for this error because it plays an important role in computing the wheel velocity in Wheel-GINS.

In Wheel-GINS, we model the Wheel-IMU lever arm and mounting angle as 2D vectors, while the wheel radius scale error is modeled as a scalar.

For lever arm, we take only the components in the y-axis and z-axis into consideration. The reason is threefold. First, the lever arm error on the wheel plane is more important. Second, the lever arm error in the direction of the rotation axis is trivial if the attitude misalignment is compensated. Third, reducing one dimension in the state vector can save some computation resources. Therefore, we have

$$\boldsymbol{l}_{wheel}^{b} = \begin{bmatrix} l_y & l_z \end{bmatrix}^{\top}. \tag{4.16}$$

The Wheel-IMU mounting angle ($\boldsymbol{\phi}_m$) can be represented by a set of Euler

angles φ_m (roll mounting angle), θ_m (pitch mounting angle), and ψ_m (heading mounting angle). Because the wheel continuously rotates with the x-axis, the roll mounting angle is negligible according to the definition of the *wheel*-frame. Therefore, we only account for the pitch mounting angle and the heading mounting angle, namely,

$$\boldsymbol{\phi}_m = \begin{bmatrix} \theta_m & \psi_m \end{bmatrix}^\top. \tag{4.17}$$

Note that we consider the 3D vector of both the lever arm and mounting angle when we derive the observation model equations to simplify the expression. In practice, we extract the components from the equations corresponding to the actual misalignment error to construct the matrix for the final computation in the EKF (see Section 4.3.3).

With regard to the wheel radius scale error, we model it as a scalar, which is the ratio of the actual wheel radius to the nominal wheel radius, namely,

$$\hat{r} = (1 + s_r)r, \tag{4.18}$$

where $\hat{r}$ is the measured wheel radius; s_r is the wheel radius scale error, augmented into the state vector for online estimation; r is the true wheel radius, which is unknown.

4.3.2 Error State Model

In Wheel-INS, we simplify the INS propagation model by omitting less significant terms, e.g., earth rotation, as it is only a local dead reckoning system using a low-cost inertial sensor. This simplification reduces computational overhead without compromising accuracy. In contrast, Wheel-GINS employs a full strapdown INS in the *earth*-frame to integrate precise global position data from GNSS. For detailed information on the strapdown INS model, refer to [83, 86].

As in Wheel-INS, we adopt an error-state model in EKF to mitigate nonlinear errors in Wheel-GINS. In addition to the robot state and the IMU sensor errors, we augment the Wheel-IMU installation parameter errors (including the Wheel-IMU lever arm and mounting angle and the wheel radius scale error) into the error state vector, which can be written as

$$\boldsymbol{x} = \begin{bmatrix} \delta\boldsymbol{p}^\top & \delta\boldsymbol{v}^\top & \boldsymbol{\phi}^\top & \boldsymbol{b}_g^\top & \boldsymbol{b}_a^\top & \boldsymbol{s}_g^\top & \boldsymbol{s}_a^\top & \delta\boldsymbol{l}_{wheel}^{b\top} & \delta\boldsymbol{\phi}_m^\top & s_r \end{bmatrix}^\top, \tag{4.19}$$

where δ indicates the error of the robot state, particularly, $\delta\boldsymbol{p}$, $\delta\boldsymbol{v}$ and $\boldsymbol{\phi} \in \mathfrak{so}(3)$ indicate the position, velocity, and attitude errors of INS in the n-frame, respectively; $\boldsymbol{b}_g$ and $\boldsymbol{b}_a$ represent the residual bias errors of the gyroscope and the accelerometer, respectively; $\boldsymbol{s}_g$ and $\boldsymbol{s}_a$ represent the residual scale factor errors of the gyroscope and accelerometer, respectively; $\delta\boldsymbol{l}_{wheel}^b \in \mathbb{R}^2$ is the residual Wheel-IMU lever arm error; $\delta\boldsymbol{\phi}_m \in \mathbb{R}^2$ is the residual Wheel-IMU mounting angle error;

$s_r \in \mathbb{R}$ is the scale error of the wheel radius. Therefore, $\boldsymbol{x}$ is a vector with 26 dimensions. The IMU sensor errors are modeled using the first-order Gauss-Markov process, while the Wheel-IMU lever arm error, mounting angle error, and wheel radius scale error are modeled using random walks.

We use the detailed error-state model [36, 86] in Wheel-GINS instead of the simplified version used in Wheel-INS. Because Wheel-GINS aims to provide accurate positioning results in large-scale (kilometer-level) environments, instead of medium-scale (hundred-meters level) environments for Wheel-INS, we have to take more terms into consideration, for example, the earth rotation, the rotation rate of the *n*-frame with respect to the *earth*-frame and the change of gravity. The robot state error state model is given by

$$\begin{cases} \delta\dot{\boldsymbol{p}} = -\boldsymbol{\omega}_{en}^{n} \times \delta\boldsymbol{p} + \delta\boldsymbol{\phi}_{en} \times \boldsymbol{v} + \delta\boldsymbol{v} \\ \delta\dot{\boldsymbol{v}} = \mathbf{R}_b^n \boldsymbol{f}^b \times \boldsymbol{\phi} + \mathbf{R}_b^n \delta\boldsymbol{f}^b + \boldsymbol{v} \times (2\delta\boldsymbol{\omega}_{ie}^{n} + \delta\boldsymbol{\omega}_{en}^{n}) \\ \qquad - (2\boldsymbol{\omega}_{ie}^{n} + \boldsymbol{\omega}_{en}^{n}) \times \delta\boldsymbol{v} + \delta\boldsymbol{g}^n \\ \dot{\boldsymbol{\phi}} = -\boldsymbol{\omega}_{in}^{n} \times \boldsymbol{\phi} + \delta\boldsymbol{\omega}_{in}^{n} - \mathbf{R}_b^n \delta\boldsymbol{\omega} \end{cases}, \tag{4.20}$$

where $\delta\boldsymbol{\phi}_{en}$ is the rotation vector describing the error of the INS-indicated *n*-frame; $\boldsymbol{\omega}_{ie}^{n}$ is the earth's rotation rate described in the *n*-frame, while $\delta\boldsymbol{\omega}_{ie}^{n}$ is its error; $\boldsymbol{\omega}_{en}^{n}$ is the rotation rate of the *n*-frame with respect to the *earth*-frame, while $\delta\boldsymbol{\omega}_{en}^{n}$ is its error; $\boldsymbol{\omega}_{in}^{n}$ is the rotation rate of the *n*-frame, while $\delta\boldsymbol{\omega}_{in}^{n}$ is its error; $\mathbf{R}_b^n$ is the rotation matrix from the *body*-frame to the *n*-frame; τ_{b_g}, τ_{b_a}, τ_{s_g}, and τ_{s_a} are the correlation time in the first-order Gauss-Markov model of the gyroscope bias, accelerometer bias, gyroscope scale factor, and accelerometer scale factor, respectively; $\boldsymbol{w}_{b_g}$ and $\boldsymbol{w}_{b_a}$ denote the driving white noise of the residual bias errors of the gyroscope and accelerometer, respectively; $\boldsymbol{w}_{s_g}$ and $\boldsymbol{w}_{s_a}$ denote the driving white noise of the scale factor errors of the gyroscope and accelerometer, respectively. More details of the error-state model can be found in Shin [86]. The IMU sensor error state model is the same as in Equation (3.6).

The lever arm error, mounting angle error, and wheel radius scale error are modeled as random walks [86], leading to the following propagation model:

$$\begin{cases} \delta\dot{\boldsymbol{l}}_{wheel}^{b} = \boldsymbol{w}_{l_w} \\ \delta\dot{\boldsymbol{\phi}}_m = \boldsymbol{w}_{\phi_m}, \\ \dot{s}_r = w_r \end{cases} \tag{4.21}$$

where $\boldsymbol{w}_{l_w}$, $\boldsymbol{w}_{\phi_m}$ and w_r denote the driving white noise of the lever arm error, mounting angle error, and wheel radius scale error, respectively.

4.3.3 Observation Model

In this section, we present a detailed derivation of the observation models, including the vehicle velocity observation, the GNSS position observation, and the wheel angular velocity observation used in the EKF scheme of the proposed Wheel-GINS. Different from Wheel-INS, we incorporate the Wheel-IMU installation parameters into the state vector for online estimation. Note that the Wheel-IMU lever arm and mounting angle errors are modeled as 2D vectors, while the wheel radius scale error is modeled as a scalar. Although our derivation considers the full 3D vector of the Wheel-IMU lever arm and mounting angle, we extract only the elements relevant to the state vector for matrix construction in practical computation.

4.3.3.1 Vehicle Velocity Observation

To compute the wheel velocity, we first need to know the rotation speed of the wheel. Therefore, we need to transform the Wheel-IMU angular velocity from the *body*-frame to the *wheel*-frame, namely,

$$\hat{\boldsymbol{\omega}}^{wheel} = \hat{\mathbf{R}}_{b}^{wheel}\hat{\boldsymbol{\omega}}, \tag{4.22}$$

where $\hat{\boldsymbol{\omega}}^{wheel}$ is the estimated angular velocity of the wheel in the *wheel*-frame; $\hat{\mathbf{R}}_{b}^{wheel}$ is the rotation matrix from the *body*-frame to the *wheel*-frame; $\hat{\boldsymbol{\omega}}$ is the angular velocity measurement of the Wheel-IMU in the *body*-frame. Because we only need the wheel angular velocity in the x-axis of the *wheel*-frame to calculate the wheel velocity, we can simplify the equation as

$$\hat{\omega}_{x}^{wheel} = \hat{\mathbf{R}}_{b(1,:)}^{wheel}\hat{\boldsymbol{\omega}}, \tag{4.23}$$

where $\hat{\omega}_{x}^{wheel}$ is the estimated angular velocity of the wheel in the x-axis of the *wheel*-frame; $\hat{\mathbf{R}}_{b(1,:)}^{wheel}$ is the first row of the rotation matrix from the *body*-frame to the *wheel*-frame, where $\hat{\mathbf{R}}_{b(1,:)}^{wheel} = \mathbf{R}_{b(1,:)}^{wheel} + \delta\mathbf{R}_{b(1,:)}^{wheel}$, and $\delta\mathbf{R}_{b(1,:)}^{wheel}$ is the error of the first row of the rotation matrix, which is governed by the Wheel-IMU mounting angle error $\delta\boldsymbol{\phi}_m$.

Further, taking into consideration the wheel radius scale error, we can calculate the forward wheel velocity and derive the error as

$$\begin{aligned}
\widetilde{v}_{wheel}^{r} &= \hat{\omega}_{x}^{wheel}\hat{r} - e_v = \hat{\mathbf{R}}_{b(1,:)}^{wheel}\hat{\boldsymbol{\omega}}\hat{r} - e_v \\
&= (\mathbf{R}_{b(1,:)}^{wheel} + \delta\mathbf{R}_{b(1,:)}^{wheel})(\boldsymbol{\omega} + \delta\boldsymbol{\omega})(1 + s_r)r - e_v \\
&= v_{wheel}^{r} + \mathbf{R}_{b(1,:)}^{wheel}r\delta\boldsymbol{\omega} + \mathbf{A}\delta\boldsymbol{\phi}_m + \mathbf{R}_{b(1,:)}^{wheel}\boldsymbol{\omega}rs_r - e_v,
\end{aligned} \tag{4.24}$$

where $\widetilde{v}_{wheel}^{r}$ and v_{wheel}^{r} are the observed and true wheel velocity in the *robot*-frame, respectively; $\delta\boldsymbol{\omega}$ is the error of the angular velocity measurement, which

is $\delta\boldsymbol{\omega} = \boldsymbol{b}_g + \text{diag}(\boldsymbol{\omega})\boldsymbol{s}_g + \boldsymbol{e}_\omega$, where $\boldsymbol{e}_\omega$ is the gyroscope noise and $\text{diag}(\cdot)$ is the diagonal matrix form of a vector; r is the wheel radius; s_r is the wheel radius scale error, and e_v is the observation noise, modeled as Gaussian white noise; $\mathbf{A}$ is the coefficent matrix of the mounting angle error, with dimensions 1×2. Comparing Equation (4.24) with the velocity observation model in Wheel-INS (Equation (4.5)), we can see that the Wheel-IMU mounting angle error and wheel radius scale error are incorporated to the observation model too be estimated online in our Wheel-GINS pipeline.

Now, we derive the matrix $\mathbf{A}$ in Equation (4.24). Given the two Wheel-IMU mounting angles, θ_m (pitch mounting angle), and ψ_m (heading mounting angle), the rotation matrix from the Wheel-IMU *body*-frame to the *wheel*-frame can be expressed as

$$\mathbf{R}_b^{wheel} = \begin{bmatrix} \cos\theta_m \cos\psi_m & -\sin\psi_m & \sin\theta_m \cos\psi_m \\ \cos\theta_m \sin\psi_m & \cos\psi_m & \sin\theta_m \sin\psi_m \\ -\sin\theta_m & 0 & \cos\theta_m \end{bmatrix}. \tag{4.25}$$

In Equation (4.24), $\mathbf{R}_{b(1,:)}^{wheel}$ indicates the first row of $\mathbf{R}_b^{wheel}$. We only use the first row because we only need the wheel angular velocity in the x-axis to compute the wheel velocity. Because the estimated Wheel-IMU mounting angle contains errors, we have the estimated value of $\mathbf{R}_{b(1,:)}^{wheel}$ as

$$\begin{aligned} \hat{\mathbf{R}}_{b(1,:)}^{wheel} &= \mathbf{R}_{b(1,:)}^{wheel} + \delta\mathbf{R}_{b(1,:)}^{wheel} \\ &= \begin{bmatrix} \cos\hat{\theta}_m \cos\hat{\psi}_m & -\sin\hat{\psi}_m & \sin\hat{\theta}_m \cos\hat{\psi}_m \end{bmatrix} \\ &= \begin{bmatrix} \cos(\theta_m + \delta\theta_m)\cos(\psi_m + \delta\psi_m) \\ -\sin(\psi_m + \delta\psi_m) \\ \sin(\theta_m + \delta\theta_m)\cos(\psi_m + \delta\psi_m) \end{bmatrix}^\top. \end{aligned} \tag{4.26}$$

After the expansion of the terms on the right side of the equation and ignoring the second-order small terms, we have

$$\delta\mathbf{R}_{b(1,:)}^{wheel} = \begin{bmatrix} -\sin\theta_m \cos\psi_m \delta\theta_m - \cos\theta_m \sin\psi_m \delta\psi_m \\ -\cos\psi_m \delta\psi_m \\ \cos\theta_m \cos\psi_m \delta\theta_m - \sin\theta_m \sin\psi_m \delta\psi_m \end{bmatrix}^\top. \tag{4.27}$$

Then matrix $\mathbf{A}$ can be expressed as

$$\mathbf{A} = \begin{bmatrix} -\omega_x^b r \sin\theta_m \cos\psi_m + \omega_z^b r \cos\theta_m \cos\psi_m \\ -\omega_x^b r \cos\theta_m \sin\psi_m - \omega_y^b r \cos\psi_m - \omega_z^b r \sin\theta_m \sin\psi_m \end{bmatrix}^\top, \tag{4.28}$$

where $\omega_x^b, \omega_y^b, \omega_z^b$ represent the components of the Wheel-IMU angular rate measurement $\boldsymbol{\omega}$ along the x, y, and z axes, namely, $\boldsymbol{\omega} = \begin{bmatrix} \omega_x^b & \omega_y^b & \omega_z^b \end{bmatrix}^\top$.

By combining the forward wheel velocity with NHC, we can formulate the complete 3D vehicle velocity model as

$$\widetilde{\boldsymbol{v}}^r_{wheel} = \begin{bmatrix} \widetilde{v}^r_{wheel} & 0 & 0 \end{bmatrix}^\top - \boldsymbol{e}_v. \tag{4.29}$$

Simultaneously, through perturbation analysis, the wheel velocity in the *robot*-frame indicated by the INS can be expressed as

$$\begin{aligned} \hat{\boldsymbol{v}}^r_{wheel} &= \hat{\mathbf{R}}^r_n \hat{\boldsymbol{v}} + \hat{\mathbf{R}}^r_b \left(\hat{\boldsymbol{\omega}}\times\right) \hat{\boldsymbol{l}}^b_{wheel} \\ &\approx \mathbf{R}^r_n (\mathbf{I} + \delta\boldsymbol{\psi}\times)(\boldsymbol{v} + \delta\boldsymbol{v}) \\ &\quad + (\mathbf{I} - \delta\boldsymbol{\phi}_m\times)\mathbf{R}^r_b(\boldsymbol{\omega}\times + \delta\boldsymbol{\omega}\times)(\boldsymbol{l}^b_{wheel} + \delta\boldsymbol{l}^b_{wheel}) \\ &\approx \boldsymbol{v}^r_{wheel} + \mathbf{R}^r_n \delta\boldsymbol{v}^n - \mathbf{R}^r_n(\boldsymbol{v}\times)\delta\boldsymbol{\psi} - \mathbf{R}^r_b(\boldsymbol{l}^b_{wheel}\times)\delta\boldsymbol{\omega} \\ &\quad + \mathbf{R}^r_b(\boldsymbol{\omega}\times)\delta\boldsymbol{l}^b_{wheel} + \left(\mathbf{R}^r_b(\boldsymbol{\omega}\times)\boldsymbol{l}^b_{wheel}\right)\times\delta\boldsymbol{\phi}_m, \end{aligned} \tag{4.30}$$

where $\mathbf{R}^r_n$ is the rotation matrix from the *n*-frame to the *robot*-frame; $\mathbf{R}^r_b$ is the rotation matrix from the *body*-frame to the *robot*-frame; $\boldsymbol{\omega}$ is the angular velocity measurement of the Wheel-IMU; $(\cdot)\times$ indicates the skewsymmetric matrix of a vector; $\boldsymbol{l}^b_{wheel}$ indicates the lever arm vector from the Wheel-IMU to the wheel center expressed in the *body*-frame, while $\delta\boldsymbol{l}^b_{wheel}$ indicates its residual error; $\delta\boldsymbol{\psi}$ is the attitude error of the vehicle. Because Wheel-INS cannot estimate the vehicle pitch angle, it is assumed that the vehicle is moving on a horizontal plane in Wheel-INS, which means the vehicle attitude only includes heading angle as the other two components are zero. Therefore, the vehicle attitude error is only related to the heading error of the attitude error in the state vector, which can be written as $\delta\boldsymbol{\psi} = \begin{bmatrix} 0 & 0 & \delta\psi \end{bmatrix}^\top$.

Then, the vehicle velocity measurement model can be expressed as

$$\begin{aligned} \delta\boldsymbol{z}_v &= \hat{\boldsymbol{v}}^r_{wheel} - \widetilde{\boldsymbol{v}}^r_{wheel} \\ &= \mathbf{H}_v\boldsymbol{x} + \boldsymbol{e}_v, \end{aligned} \tag{4.31}$$

where $\mathbf{H}_v$ is a 3×26 matrix of the form

$$\mathbf{H}_v = \begin{bmatrix} \mathbf{0} & \mathbf{R}^r_n & -\mathbf{R}^r_n(\boldsymbol{v}\times) & -\mathbf{R}^r_b(\boldsymbol{l}^b_{wheel}\times) & \mathbf{0} & \mathbf{B} & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{E} \end{bmatrix}, \tag{4.32}$$

and

$$\begin{cases} \mathbf{B} = -\mathbf{R}^r_b(\boldsymbol{l}^b_{wheel}\times)\mathrm{diag}(\boldsymbol{\omega}) \\ \mathbf{C} = \mathbf{R}^r_b(\boldsymbol{\omega}\times)_{(:,2:3)} \\ \mathbf{D} = \left(\mathbf{R}^r_b(\boldsymbol{\omega}\times)\boldsymbol{l}^b_{wheel}\right)\times_{(:,2:3)} - \begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \end{bmatrix}^\top \\ \mathbf{E} = \begin{bmatrix} -\mathbf{R}^{wheel}_{b(1,:)}\boldsymbol{\omega}r & 0 & 0 \end{bmatrix}^\top \end{cases} \tag{4.33}$$

where $_{(:,2:3)}$ indicates the second and third columns of a matrix. Note that we take only the last two columns of the matrix corresponding to the actual misalignment error to build $\mathbf{C}$ and $\mathbf{D}$ in $\mathbf{H}_v$ because the Wheel-IMU lever arm error and mounting angle error are in 2-dimension as can be seen in Equation (4.16) and Equation (4.17).

4.3.3.2 GNSS Observation

We calculate the difference between the observed GNSS antenna position from the receiver and the INS-predicted antenna position to build the GNSS observation model. Given the IMU position propagated by the strapdown INS, the GNSS antenna's position in the *earth*-frame can be derived as

$$\hat{\boldsymbol{p}}_{gnss}^{e} = \hat{\boldsymbol{p}}_{b}^{e} + \mathbf{R}_{n}^{e}\hat{\mathbf{R}}_{b}^{n}\hat{\boldsymbol{l}}_{gnss}^{b}, \tag{4.34}$$

where $\mathbf{R}_{n}^{e}$ is the rotation matrix from the local navigation frame to the *earth*-frame; $\boldsymbol{l}_{gnss}^{b}$ is the GNSS lever arm, equaling to the vector from the Wheel-IMU center to the GNSS antenna center expressed in the IMU *body*-frame.

Unlike the traditional GNSS/INS integrated navigation system where the lever arm between the GNSS antenna and the IMU is fixed, the GNSS lever arm expressed in the IMU *body*-frame changes in Wheel-GINS because the Wheel-IMU continuously rotates with the wheel. Therefore, the GNSS lever arm has to be computed in real-time in Wheel-GINS.

Given that the origin of the *robot*-frame is aligned with the wheel center, the position of the GNSS antenna is constant in the *robot*-frame, which can be measured beforehand. Then, the GNSS lever arm w.r.t the IMU *body*-frame can be calculated as

$$\hat{\boldsymbol{l}}_{gnss}^{b} = \hat{\boldsymbol{l}}_{wheel}^{b} + \hat{\mathbf{R}}_{r}^{b}\boldsymbol{l}_{gnss}^{r}. \tag{4.35}$$

Consequently, we can get the GNSS observation equation

$$\begin{aligned}
\delta\boldsymbol{z}_{gnss} &= \mathbf{R}_{e}^{n}\left(\hat{\boldsymbol{p}}_{gnss}^{e} - \tilde{\boldsymbol{p}}_{gnss}^{e}\right) \\
&= \delta\boldsymbol{p} + \left(\mathbf{R}_{b}^{n}\left(\boldsymbol{l}_{wheel}^{b} + \mathbf{R}_{r}^{b}\boldsymbol{l}_{gnss}^{r}\right)\right)\times\boldsymbol{\phi} \\
&\quad - \mathbf{R}_{b}^{n}\mathbf{R}_{v}^{b}\left(\boldsymbol{l}_{gnss}^{r}\times\right)\delta\boldsymbol{\phi}_{m} + \mathbf{R}_{b}^{n}\delta\boldsymbol{l}_{wheel}^{b} \\
&= \mathbf{H}_{gnss}\boldsymbol{x} + \boldsymbol{e}_{gnss},
\end{aligned} \tag{4.36}$$

where $\mathbf{H}_{gnss}$ is a 3×26 matrix of the form

$$\mathbf{H}_{gnss} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{F} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}_{b(:,2:3)}^{n} & \mathbf{G} & \mathbf{0} \end{bmatrix}, \tag{4.37}$$

and

$$\begin{cases} \mathbf{F} = \left(\mathbf{R}_{b}^{n}\left(\boldsymbol{l}_{wheel}^{b} + \mathbf{R}_{v}^{b}\boldsymbol{l}_{gnss}^{r}\right)\right)\times \\ \mathbf{G} = -\mathbf{R}_{b}^{n}\mathbf{R}_{v}^{b}\left(\boldsymbol{l}_{gnss}^{r}\times\right)_{(:,2:3)} \end{cases}. \tag{4.38}$$

4.3.3.3 Wheel Angular Velocity Observation

To effectively estimate the Wheel-IMU mounting angle online, we construct and integrate a wheel angular velocity observation model into Wheel-GINS. Note that the wheel angular velocity constraint is only used for the Wheel-IMU mounting angle estimation, not to help with the robot state estimation.

Assuming that the vehicle travels in a straight line without turning and ignoring the Earth's rotation, the wheel only has angular velocity along the rotation axis, namely,

$$\widetilde{\boldsymbol{\omega}}^{wheel} = \begin{bmatrix} \widetilde{\omega}_x^{wheel} & 0 & 0 \end{bmatrix}^\top - \boldsymbol{e}_\omega, \tag{4.39}$$

where $\widetilde{\boldsymbol{\omega}}^{wheel}$ indicates the angular velocity of the wheel in the *wheel*-frame; $\widetilde{\omega}_x^{wheel}$ is the x-axis component of $\widetilde{\boldsymbol{\omega}}^{wheel}$; $\boldsymbol{e}_\omega$ indicates the measurement noise. As a result, the angular velocity sensed by the y-axis and z-axis of the Wheel-IMU should equal zero if there is no attitude misalignment error. Therefore, we can construct an observation model based on this fact to estimate the attitude misalignment between the Wheel-IMU and the wheel. The angular velocity of the wheel in the *wheel*-frame computed with the Wheel-IMU gyroscope readings can be expressed as

$$\begin{aligned} \hat{\boldsymbol{\omega}}^{wheel} &= \hat{\mathbf{R}}_b^{wheel} \hat{\boldsymbol{\omega}} \\ &= (\mathbf{I} - \boldsymbol{\phi}_m \times) \mathbf{R}_b^{wheel} (\boldsymbol{\omega} + \delta\boldsymbol{\omega}), \end{aligned} \tag{4.40}$$

where $\mathbf{R}_b^{wheel}$ indicates the rotation matrix from the *body*-frame to the *wheel*-frame. Then, the wheel angular velocity observation model can be written as

$$\begin{aligned} \delta \boldsymbol{z}_\omega &= \hat{\boldsymbol{\omega}}^{wheel} - \tilde{\boldsymbol{\omega}}^{wheel} \\ &= \mathbf{R}_b^{wheel} \delta\boldsymbol{\omega} + (\mathbf{R}_b^{wheel} \boldsymbol{\omega}) \times \delta\boldsymbol{\phi}_m \\ &= \mathbf{H}_\omega \boldsymbol{x} + \boldsymbol{e}_\omega, \end{aligned} \tag{4.41}$$

where $\mathbf{H}_\omega$ is a 2×26 matrix of the form

$$\mathbf{H}_\omega = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}_b^{wheel} & \mathbf{0} & \mathbf{R}_b^{wheel} \mathrm{diag}(\boldsymbol{\omega}) & \mathbf{0} & \mathbf{0} & (\mathbf{R}_b^{wheel} \boldsymbol{\omega}) \times_{(:,2:3)} & \mathbf{0} \end{bmatrix}. \tag{4.42}$$

Note that we use only the bottom two rows of the matrices on both sides of Equation (4.41) to construct the observation model because only the y-axis and z-axis component of the wheel angular velocity in the *wheel*-frame should equal zero. Therefore, we take only the last two rows of $\mathbf{H}_\omega$.

One may argue that the assumption of the vehicle moving along a straight line is strong because the vehicle inevitably turns during motion. The turning of the vehicle will project angular velocity to the y-axis and z-axis of the wheel, making Equation (4.39) not strictly valid. However, in practice, we can detect the vehicle turning with the Wheel-IMU. Thus, we can employ the wheel angular velocity observation model to estimate the Wheel-IMU mounting angle specifically when the vehicle is moving straight and subsequently fix it once convergence is achieved. Our experimental results have shown that the Wheel-IMU mounting angle converges fast. In addition, the occasional normal turning of the vehicle has negligible impact on both the Wheel-IMU mounting angle and vehicle pose estimation.

Table 4.7: Vehicle motion information in the experiments

Sequence	Vehicle	Average speed (m/s)	Total distance (m)
1	Pioneer 3DX	1.25	≈1146
2	Trolley	1.41	≈1990
3	Car	6.00	≈6566

4.3.4 Experimental Results

4.3.4.1 Experimental Setup

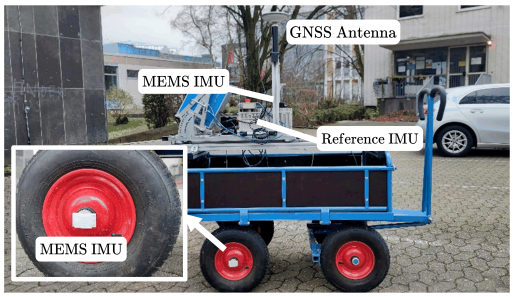
To evaluate the performance of Wheel-GINS, we collected real-world data in three different places with three different wheeled vehicles. One was the Pioneer 3DX robot used in our Wheel-INS experiments (see Section 4.1.5), the other was a trolley, and the third was a regular street car. Figure 4.15 shows the experimental trolley and car and the corresponding trajectories. We used the same Wheel-IMU as used in the experiments of Wheel-INS and Wheel-SLAM. An IMU of the same model was mounted on the top of the vehicle to perform ODO-GINS for comparison. Because we didn't have an external odometer and it was difficult to access the wheel encoder of the car, we calibrated the Wheel-IMU and compensated its error in advance to get the wheel velocity for ODO-GINS, but we didn't do this for Wheel-GINS. Therefore, the wheel velocity used in ODO-GINS is more accurate than that calculated in Wheel-GINS.

We used the same dataset collected in Sequence (Seq.) 1. The other two experimental platforms and sequences are shown in Figure 4.15. Both the trolley and regular car were equipped with an IMAR iNav-FJI[4], a navigation-grade IMU for a near ground truth trajectory. We performed a smoothed PPK/INS integration method to compute this near ground truth trajectory with the high-end reference IMU. We used the same PPK GNSS position for both Wheel-GINS and ODO-GINS. The satellite numbers in the three sequences over time are shown in Figure 4.16.

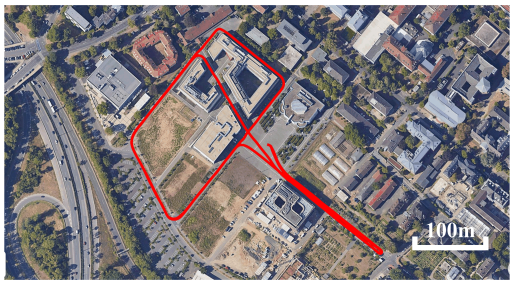
As shown in Figure 4.4(a), Seq. 1 is a one-way polyline trajectory with no return on an experimental field at a university. As shown in Figure 4.15, Seq. 2 is a loop trajectory on a university campus, and Seq. 3 is a large-scale loop trajectory where the vehicle drove twice on the same road. The average speed and the total traveled distance of the vehicles in the three sequences are listed in Table 4.7.

We set the initial heading and position of both Wheel-GINS and ODO-GINS with the reference system. In real applications, the system initialization problem

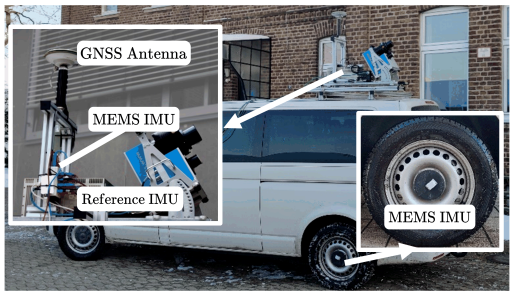
[4] `https://www.imar-navigation.de/downloads/NAV_FJI_001-J_en.pdf`



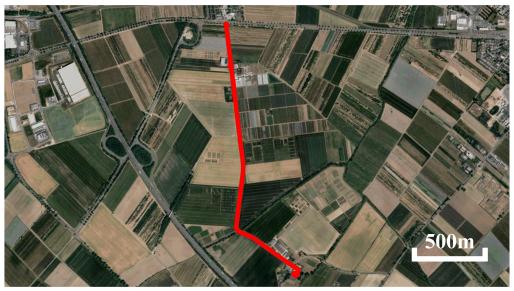
(a) Experimental trolley and the devices used in Seq. 2.



(b) Seq. 2: loop trajectory with overlap.



(c) Experimental car and the devices used in Seq. 3.



(d) Seq. 3: large-scale trajectory with twice repetition.

Figure 4.15: The experimental platforms and trajectories in Seq. 2 and Seq. 3.

can be solved by online alignment approaches [16]. We calibrated the attitude misalignment between the reference IMU and the vehicle, as well as between the Body-IMU and the vehicle in advance using the method proposed in Chen et al. [18]. Furthermore, we set the initial gyroscope bias using the static IMU data collected before the vehicle started moving. Other inertial sensor errors and the Wheel-IMU installation parameter errors were initialized as zero. The GNSS update frequency was set to 1 Hz while the velocity update frequency was set to 2 Hz for both Wheel-GINS and ODO-GINS.

4.3.4.2 Positioning Performance Evaluation

Table 4.8 presents the position and heading RMSE of the proposed Wheel-GINS and ODO-GINS in the three sequences. It can be observed from Table 4.8 that Wheel-GINS achieves comparable accuracy to ODO-GINS: the positioning error is at the centimeter level, and the heading error is mostly less than 1°. In Seq. 1, the heading RMSE of Wheel-GINS is much larger than ODO-GINS. This is because the unevenness of the road caused significant vibration of the robot when it was moving. At the same time, the reference IMU was placed on top of the robot, not on the wheel. As a consequence, there is a larger error in the Wheel-GINS attitude estimates.

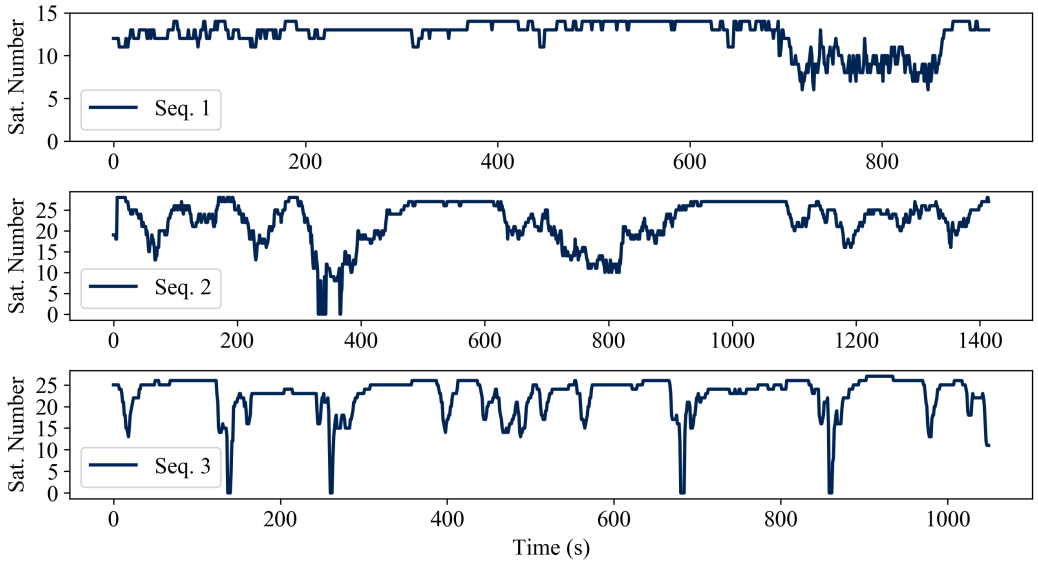


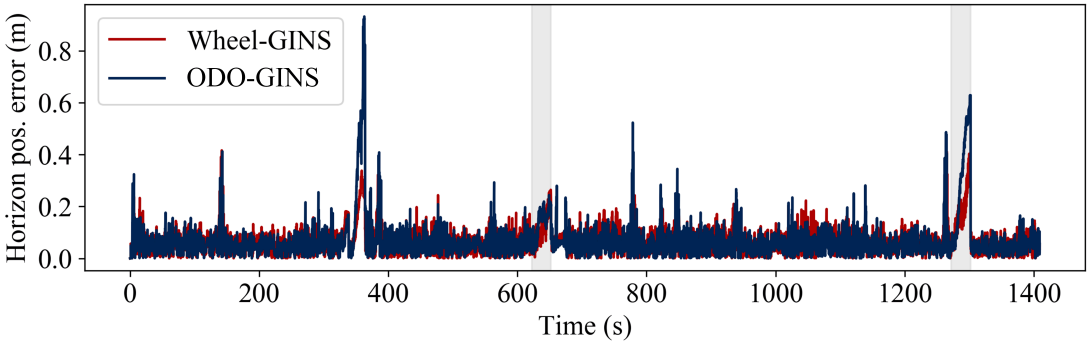
Figure 4.16: The number of GNSS satellites used in the three sequences.

We can also see that with the integration of GNSS position observation, Wheel-GINS has improved Wheel-INS from a 2D dead reckoning system to a full 3D positioning system with accurate height estimation. In addition, Wheel-GINS does not show significant advantages compared to ODO-GINS when GNSS is always available. This is because the high-quality GNSS position observation has effectively suppressed the error drift of Wheel-INS and the traditional odometer-aided INS and helped estimate the IMU sensor errors.

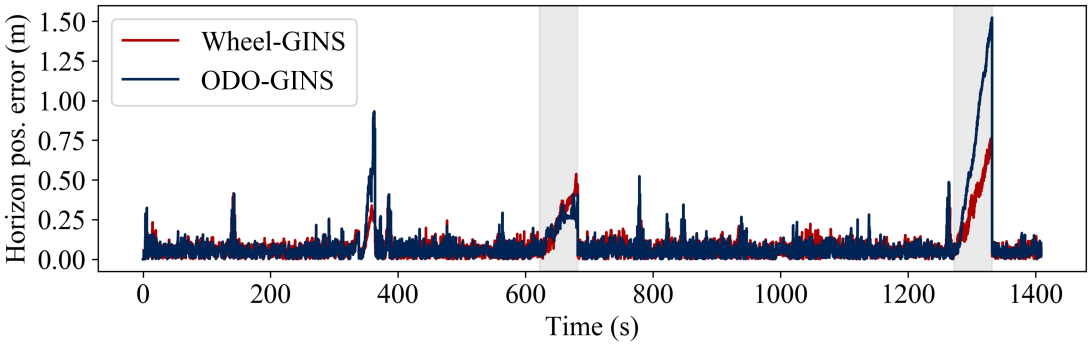
Table 4.8: Pose accuracy statistics of Wheel-GINS and ODO-GINS

Seq.	System	Horizon pos. RMSE (m)	Height RMSE (m)	Heading RMSE (°)
1	Wheel-GINS	**0.04**	**0.03**	1.15
	ODO-GINS	0.05	0.05	**0.61**
2	Wheel-GINS	**0.07**	**0.04**	**0.39**
	ODO-GINS	0.10	**0.04**	**0.39**
3	Wheel-GINS	0.10	0.26	**0.35**
	ODO-GINS	**0.09**	**0.13**	0.38

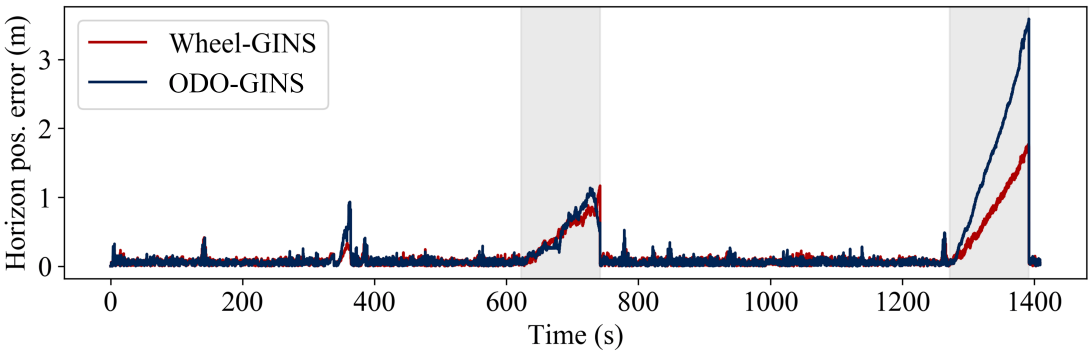
A commonly used method to evaluate the performance of a GNSS-aided integrated navigation system is to investigate the error drift when GNSS is blocked [13, 41, 74, 127]. To this end, we manually set three different lengths (namely, 30 s, 60 s, and 120 s) of GNSS outages in the three sequences to compare the performance of Wheel-GINS with ODO-GINS. We set two outages for each



(a) Horizon position errors of Wheel-GINS and ODO-GINS with two 30 s GNSS outages in Seq. 2.



(b) Horizon position errors of Wheel-GINS and ODO-GINS with two 60 s GNSS outages in Seq. 2.



(c) Horizon position errors of Wheel-GINS and ODO-GINS with two 120 s GNSS outages in Seq. 2.

Figure 4.17: Horizon position errors of Wheel-GINS and ODO-GINS with different lengths of GNSS outages in Seq. 2. The gray areas indicate the GNSS outage period.

sequence. Figure 4.17 shows the horizon positioning error of Wheel-GINS and ODO-GINS with the three different lengths of GNSS outage in Seq. 2. Table 4.9 lists the mean RMSE and MAX horizontal position error of the two systems during GNSS outages in the three sequences.

We can see from Figure 4.17 that without the global position information during the GNSS outages, the horizon positioning errors of both Wheel-GINS and ODO-GINS are growing. The horizon position error accumulation increases with the increase of the GNSS outage period. Due to the random error characteristics of the IMU, the drift rate is different at different points of the sequence with the same length of GNSS outage period. Note that the error drifts at around 350 s in Figure 4.17 are caused by the inferior GNSS conditions where the trolley was surrounded by high buildings.

Table 4.9: Comparison of position error between Wheel-GINS and ODO-GINS during GNSS outages

Seq.	System	Horizontal position (m)					
		RMSE	MAX	RMSE	MAX	RMSE	MAX
Outage time		30 s		60 s		120 s	
1	Wheel-GINS	**0.16**	**0.27**	**0.26**	**0.62**	**0.68**	**1.28**
	ODO-GINS	0.32	0.49	0.44	0.70	0.93	1.86
2	Wheel-GINS	**0.16**	**0.33**	**0.34**	**0.65**	**0.73**	**1.47**
	ODO-GINS	0.25	0.44	0.53	0.97	1.23	2.36
3	Wheel-GINS	0.38	**0.67**	**0.75**	**1.37**	**1.37**	**2.23**
	ODO-GINS	**0.37**	0.71	0.92	1.91	2.45	4.46

In addition, we can see from Table 4.9 that Wheel-GINS exhibits higher accuracy than ODO-GINS during GNSS outages. Compared to ODO-GINS, the horizontal position RMSE of Wheel-GINS in 30 s, 60 s, and 120 s GNSS outages has been averagely reduced by 28%, 32%, and 37%, respectively. The reason for this is that Wheel-INS exhibits a lower error drift rate than the traditional odometer-aided INS, as illustrated in [76]. Even though GNSS helps to limit the error drift when it is available, Wheel-GINS outperforms ODO-GINS during GNSS outages thanks to the inherent rotation modulation effect.

4.3.4.3 Online Wheel-IMU Installation Parameters Estimation

In this section, we delve deeper into Wheel-GINS's capacity for online installation parameters estimation to give better insights into the system characteristics. The

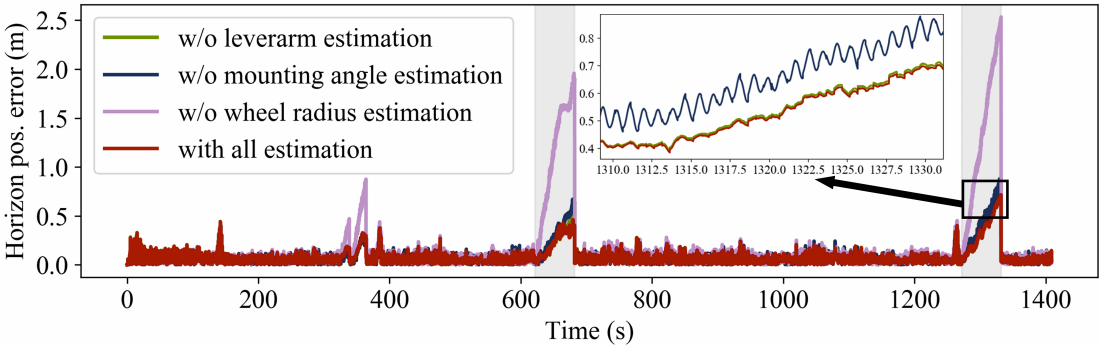


Figure 4.18: Horizontal position error drift of Wheel-GINS during 60 s GNSS outages with and without online estimation of the Wheel-IMU installation parameters in Seq. 2.

results support our second claim that Wheel-GINS can effectively estimate Wheel-IMU installation parameters, including the Wheel-IMU lever arm and mounting angle and the wheel radius scale error online, thus greatly improving the pose estimation accuracy.

First, we compare the positioning accuracy of Wheel-GINS during GNSS outages with and without the online estimation of the Wheel-IMU installation parameters to qualitatively illustrate the necessity of online estimation of the Wheel-IMU installation parameters. We set two 60 s GNSS outages in Seq. 2. Figure 4.18 shows the horizon position error of Wheel-GINS with and without the online estimation of the Wheel-IMU installation parameters in Seq. 2.

We can see that the positioning accuracy of Wheel-GINS is significantly improved with the online estimation of the Wheel-IMU installation parameters. Specifically, the influence of the wheel radius scale error is more evident in this experiment. This is because the wheel radius scale error directly affects the wheel velocity estimation, which is crucial when GNSS is unavailable. In addition, we can see that the Wheel-IMU lever arm and mounting angle errors also introduce positioning errors if not appropriately compensated. The Wheel-IMU mounting angle modulates a sine signal onto the positioning error because of the continuous rotation of the wheel. The influence of the Wheel-IMU lever arm is not significant in this experiment because it is less than 5 cm (see Figure 4.19). We can constrain the Wheel-IMU lever arm error within this level by carefully installing the IMU. In conclusion, each Wheel-IMU installation error causes position error for Wheel-GINS at different levels if not calibrated properly. In the following section, we show experimental results to analyze the online estimation of each Wheel-IMU installation parameters in the proposed Wheel-GINS, respectively.

1) Online lever arm Estimation

We first investigate the online estimation of the Wheel-IMU lever arm in Wheel-GINS. Because it is difficult to get the ground truth value of the Wheel-

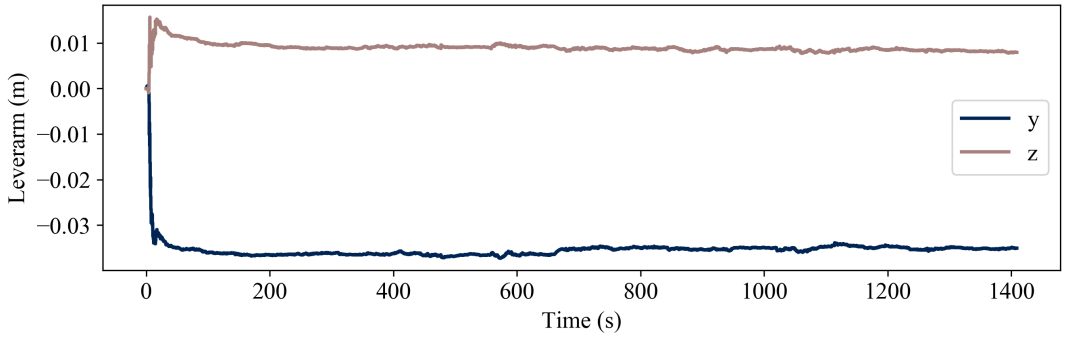


Figure 4.19: Online Wheel-IMU leverarm estimation results in Seq. 2. y and z represent the two leverarm components in the y-axis and z-axis of the *body*-frame (see Equation (4.16)).

IMU lever arm, we can only evaluate the effectiveness of the lever arm online estimation by looking at the convergence of the error. Figure 4.19 plots the Wheel-IMU lever arm estimation result in Seq. 2. The figure shows that the Wheel-IMU lever arm error can be effectively estimated in Wheel-GINS, which converges in around 90 s. Because it is difficult to accurately measure the Wheel-IMU lever arm in practice, the online estimation of the Wheel-IMU lever arm is essential for Wheel-GINS to achieve high positioning accuracy.

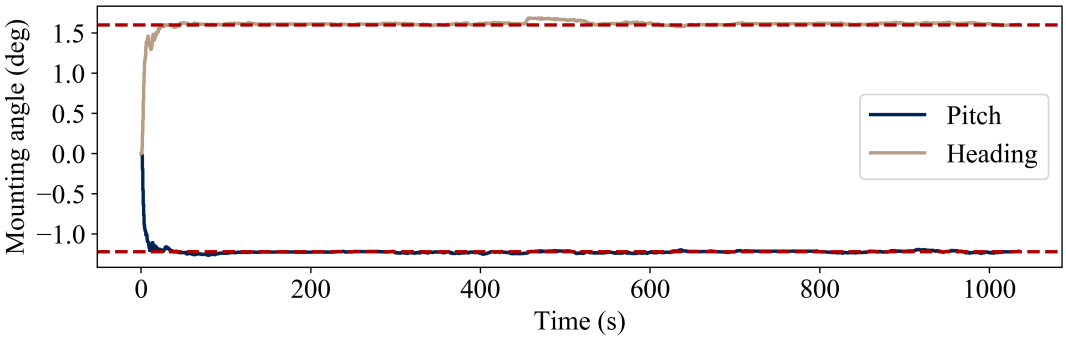


Figure 4.20: Online Wheel-IMU mounting angle estimation results in Seq. 3. The red dashed line represents the reference value calculated by the offline calibration method [17] (pitch mounting angle: -1.22°, heading mounting angle: 1.60°).

2) Online Mounting Angle Estimation

Experimental results in Wheel-INS [76] and Tan [92] have illustrated that the Wheel-IMU mounting angle error causes significant pose error if it is not estimated and compensated properly. In this section, we analyze the online estimation of the Wheel-IMU mounting angle in Wheel-GINS. Figure 4.20 compares the online estimation results of the Wheel-IMU mounting angle in Wheel-GINS with the offline calibration results [17] in Seq. 3. We can see that the Wheel-IMU

mounting angle error converges to the reference value in around 30 s in Wheel-GINS. After convergence, it remains stable even when the vehicle occasionally turns. Figure 4.15 (d) shows that there are some turns and even u-turns in Seq. 3 because the vehicle traversed the same road back and forth twice. We can see from Figure 4.20 that it does not disrupt the Wheel-IMU mounting angle estimation. These results back up our claim in Section 4.3.3.3 that the Wheel-IMU mounting angle estimation converges fast in Wheel-GINS, and the occasional normal turning of the vehicle has negligible impact on the Wheel-IMU mounting angle estimation.

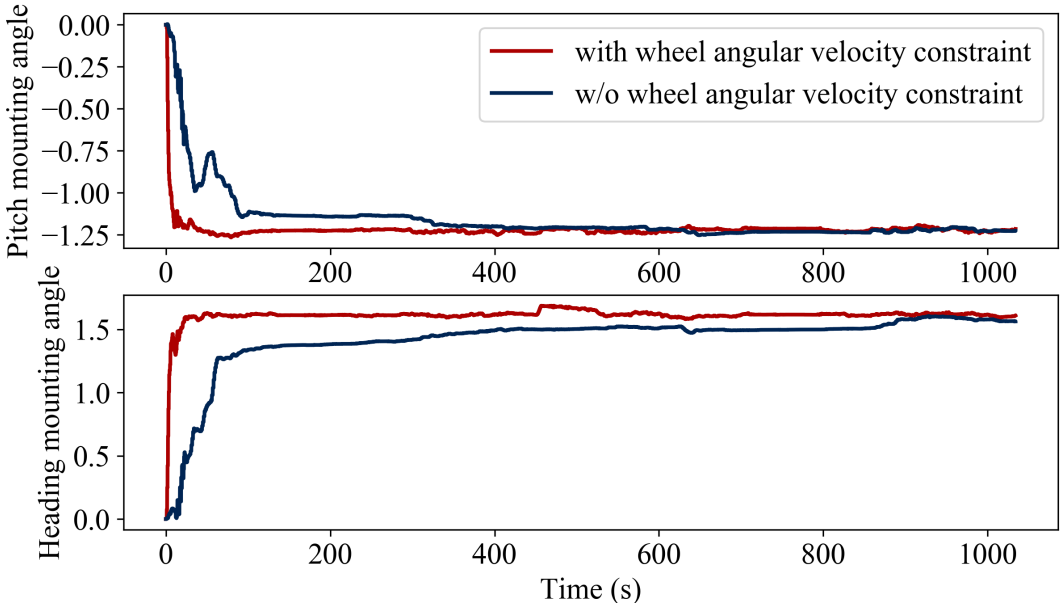


Figure 4.21: Online Wheel-IMU mounting angle estimation with/without the proposed wheel angular velocity constraint in Seq. 3. (Unit: degree).

One may argue that the GNSS position can also help estimate the Wheel-IMU heading misalignment as it provides absolute heading information for the vehicle; thus, it is unnecessary to integrate the proposed wheel angular velocity constraint model. We conduct a comparison experiment to show that the proposed angular velocity measurement can significantly accelerate the convergence of the Wheel-IMU mounting angle error. Figure 4.21 shows the results. As we can see, with the integration of the proposed wheel angular velocity constraint, the convergence time of the Wheel-IMU pitch and heading mounting angle estimation has been reduced from around 400 s and 900 s to 30 s, respectively.

Although the GNSS position indirectly reflects the vehicle heading, which helps to estimate the Wheel-IMU mounting angle, it takes a long time to achieve convergence. In addition, the accuracy of the GNSS position is also critical. When the accuracy of GNSS positioning is poor, such as in complex environments, it even impairs the estimation of the Wheel-IMU mounting angle. However, the

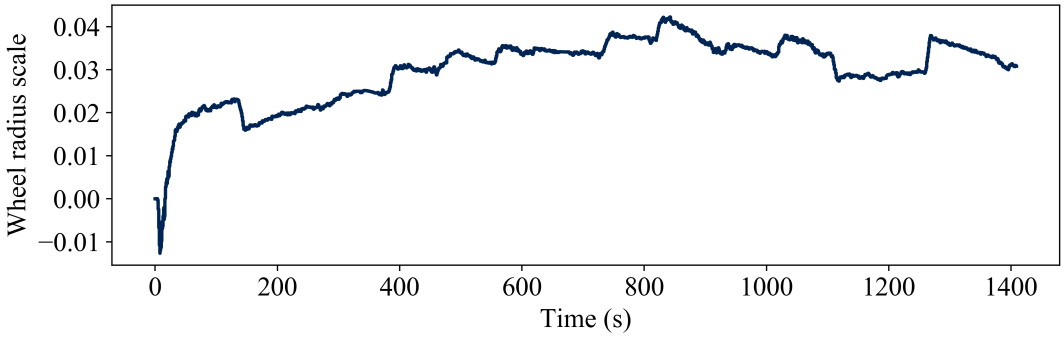


Figure 4.22: Online wheel radius scale error estimation results of Wheel-GINS in Seq. 2.

proposed wheel angular velocity constraint is not affected by the environment. Therefore, the proposed wheel angular velocity measurement plays a key role in estimating the Wheel-IMU mounting angle to improve the pose accuracy in Wheel-GINS.

3) Online Wheel Radius Scale Error Estimation

We now conduct experiments to illustrate that the wheel radius scale error can be effectively estimated in Wheel-GINS. Figure 4.22 plots the online estimation result of the wheel radius scale error in Seq. 2. From the figure, we can see that the wheel radius scale error drifts at the beginning because of the coupling effect from other installation parameters and IMU errors. Still, it soon converges to a reasonable range. Due to the pneumatic nature and softness of the tires in the trolley used in Seq. 2, deformation is prone to occur on uneven road surfaces. Consequently, there are some variations of the wheel radius scale estimation along with the sequence. It is also difficult to accurately calibrate the wheel radius scale error offline because it varies due to the terrain, tire pressure, vehicle weight, and so on. Therefore, the online estimation of the wheel radius scale error is necessary for Wheel-GINS to achieve high positioning accuracy, especially for the robots with pneumatic tires prone to deformation.

4.3.5 Conclusion

Our goal in this study was to build an accurate and robust localization system for wheeled robots in large-scale environments (e.g., kilometer level). For that, we proposed Wheel-GINS, a GNSS/INS integrated navigation system using a Wheel-IMU. Based on Wheel-INS, we integrated the GNSS position observation into the EKF pipeline. To take full advantage of the absolute position information from GNSS, we augmented Wheel-IMU installation parameters, including the Wheel-IMU lever arm and mounting angle and the wheel radius scale error, into the state vector to be estimated online. Furthermore, we proposed a novel wheel

angular velocity observation model to accelerate the convergence of the Wheel-IMU mounting angle error.

Real-world experimental results have illustrated that the proposed Wheel-GINS can achieve similar localization performance compared to the conventional ODO-GINS when GNSS is always available, while it significantly outperforms ODO-GINS during GNSS outages. Additionally, the Wheel-IMU installation parameters, including the Wheel-IMU lever arm and mounting angle and wheel radius scale error, can be effectively estimated online, thus improving the localization accuracy and the practicality of the system.

Chapter 5

State Estimation for Legged Robots Using Multiple Leg-Mounted IMUs

In the previous chapter, we presented our novel state estimation systems utilizing a custom Wheel-IMU, tailored for wheeled robots. We started with Wheel-INS, a dead reckoning solution that outperforms traditional ODO/INS system in both accuracy and robustness using only one low-cost IMU. Then, we extended Wheel-INS from a localization system to a full SLAM system which extracts and matches terrain features for loop closure detection. Lastly, we integrated GNSS with Wheel-INS to limit the long-term error drift and estimate the Wheel-IMU installation parameters online.

Although wheeled robots have been widely used in various real-world applications, legged robots offer greater application potential due to their enhanced mobility and have garnered significant attention from both academia and industry these years. Legged robots rely on accurate state estimation to keep balance and navigate challenging environments [5, 9, 62, 117]. To this end, proprioceptive sensors, such as IMUs and joint encoders, play an essential role. On one hand, they provide high-frequency egomotion information that can be fused with the exteroceptive sensors, such as LiDARs and cameras [77, 105, 118], to enhance localization accuracy. On the other hand, they serve as a backup solution when the exteroceptive sensors fail, such as in dark environments or under canopy conditions in agricultural applications.

Current approaches to this problem primarily use the robot body-mounted IMU (Body-IMU), joint encoders and foot force sensors [4, 5, 39, 62]. In these systems, the Body-IMU initially propagates the robot's main body state, which is then refined by integrating leg kinematics estimated from joint encoders upon detecting foot contact using the foot force sensor. The most widely used ap-

proach for foot contact detection relies on empirically thresholding force sensor readings, but this method is susceptible to errors due to slippage and sensor degradation over time. In addition, the foot positions during swing phase cannot be estimated [5] and need to be relocated when the foot regains contact with the ground, introducing further error into the robot state estimation.

Recent studies have attempted to mount IMUs on robot legs (Leg-IMUs) to detect foot contact and improve state estimation. Yang et al. [117] proposed to mount multiple IMUs on the robot feet to provide additional foot velocity measurements within the similar extended Kalman filter (EKF) structure as in Bloesch et al. [4, 5]. However, the state estimation capability of the Leg-IMUs had not been fully utilized because IMU can provide six degrees of freedom pose estimation, not only velocity measurement.

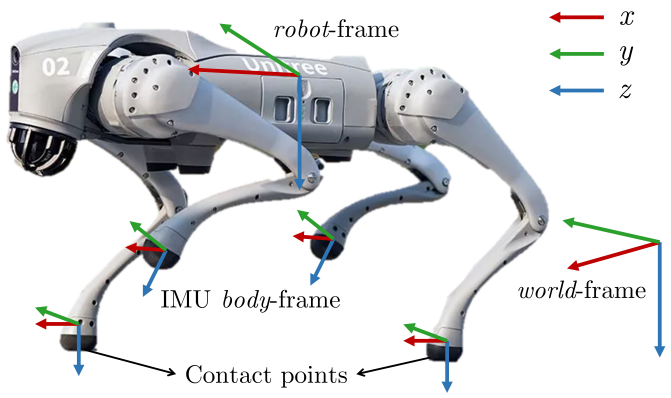


Figure 5.1: Illustration of the installation of the multiple Leg-IMUs and the coordinate systems of our proposed DogLegs on a quadruped robot. Our approach fuses the measurements from a Body-IMU, multiple Leg-IMUs and joint encoders. The Leg-IMUs are mounted on the lower legs of the robot and close to the feet. The *robot*-frame is defined with the origin at the center of the robot body and aligns with the coordinates of the Body-IMU. The *world*-frame is defined with the body-frame at the initial position.

In this chapter, we introduce DogLegs, our proposed framework that fuses the full state estimation information from all the individual IMU systems, including the Body-IMU and multiple Leg-IMUs, through an EKF. Figure 5.1 illustrates the installation of multiple Leg-IMUs and the coordinate systems involved within our proposed DogLegs framework, using a quadruped robot as an example.

In summary, we make two major claims:

- Doglegs achieves better performance in state estimation compared to the traditional EKF-based leg odometry (using only a Body-IMU and joint encoders) across different terrains.
- DogLegs reduces the error drift of the individual IMU systems by incorporating the relative position constraints between the multiple IMUs.

5.1 Overview

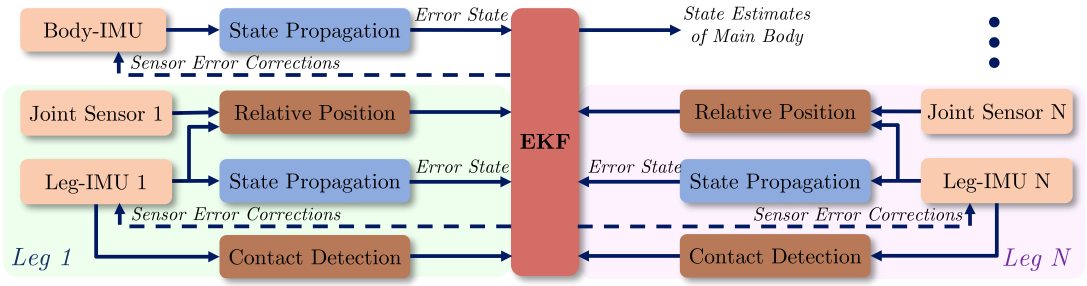


Figure 5.2: Overview of our proposed DogLegs system. We fuse the measurements from a Body-IMU, joint encoders, and multiple Leg-IMUs to estimate the state of the main body for legged robots via an error-state EKF. The system model of our EKF framework contains the error states from all IMUs. N represents the number of legs. Joint encoders on each leg provide the relative position constraints between the Body-IMU and the corresponding Leg-IMU. Additionally, we refine the IMU outputs by compensating for estimated sensor errors.

Figure 5.2 illustrates the algorithmic structure of our proposed DogLegs system. Our method is implemented using an error-state EKF to fuse the measurements from joint encoders and all the IMUs. The Leg-IMUs propagate the states of their mounting points on the legs, while the Body-IMU propagates the state of the main body. Simultaneously, the Leg-IMUs detect foot contact, i.e., zero velocity, to update their respective states. Additionally, relative position constraints between the Body-IMU and Leg-IMUs (estimated with joint encoders) are incorporated to mitigate error accumulation across all IMUs. We use the estimated IMU sensor errors to correct the IMU readings.

5.2 Foot Contact Detection

We detect the robot foot contact by analyzing the Leg-IMU readings [98] rather than force sensor measurements. The main reason is that IMUs provide richer motion information, capturing six degrees of freedom, whereas foot force sensors are typically limited to measuring only the normal contact force in many robots, e.g., Unitree Go1 and Go2 robot[1].

Given the fact that the foot acceleration equals to zero at the moment of contact, we employ the generalized likelihood ratio test [53, 72, 88] using a batch of accelerometer measurements as the foot contact detection criterion, which is given by

[1] `https://shop.unitree.com/`

$$\frac{1}{2M+1}\sum_{i=k-M}^{k+M}\left\|\boldsymbol{f}_i-\boldsymbol{g}\frac{\bar{\boldsymbol{f}}}{\|\bar{\boldsymbol{f}}\|}\right\|\leq\boldsymbol{\gamma},\tag{5.1}$$

where, $2M+1$ is the batch size; $\boldsymbol{f}_i$ is the accelerometer measurement at time i; $\boldsymbol{g}$ is the gravity vector; $\bar{\boldsymbol{f}}$ is the average accelerometer measurement within the batch; $\|\cdot\|$ is the L2 norm of a vector; $\boldsymbol{\gamma}$ is an empirically determined threshold.

5.3 Error State Model

Given gyroscope and acceleration measurements at each time step, the state of the rigid platform on which the IMU is mounted can be propagated by integrating the IMU data. In this paper, we employ the strapdown inertial navigation system (INS) [36, 86, 107] for each IMU to propagate its own state. Meanwhile, to mitigate system nonlinearity, we adopt the error-state representation [76] as the system model within the EKF framework. The whole error-state vector maintained in the EKF is given by

$$\boldsymbol{\mathcal{X}}=\left[\delta\boldsymbol{x}_0^\top\ ,\ \delta\boldsymbol{x}_1^\top\ ,\ \cdots\ ,\ \delta\boldsymbol{x}_N^\top\right]^\top,\tag{5.2}$$

where N represents the number of legs, and $\delta\boldsymbol{x}$ represents the error state of each IMU.

In Section 3.3, we introduced the INS error propagation model including the robot state error and IMU bias and scale factor error. We used this 21-state model in our Wheel-INS and Wheel-GINS framework. However, the IMU scale factor error in DogLegs is not observable due to insufficient execution. Therefore, to reduce the state dimension and consequently improve computation efficiency, we eliminate the IMU scale factor error from the error state in Equation (3.11) and only estimate the 15-state for each IMU, which is defined as

$$\delta\boldsymbol{x}=\left[\delta\boldsymbol{p}^\top\ ,\ \delta\boldsymbol{v}^\top\ ,\ \boldsymbol{\phi}^\top\ ,\ \boldsymbol{b}_g^\top\ ,\ \boldsymbol{b}_a^\top\right]^\top\in\mathbb{R}^{15},\tag{5.3}$$

where $\delta\boldsymbol{p}$, $\delta\boldsymbol{v}$ and $\boldsymbol{\phi}$ denote the position, velocity, and attitude errors respectively, and $\boldsymbol{b}_g$ and $\boldsymbol{b}_a$ denote the residual bias errors of the accelerometer and gyroscope, respectively. We integrate the error states from each IMU into a single vector, which serves as the system model in our EKF pipeline [109]. Consequently, the total dimension of the error state vector is $15\,(N+1)$.

5.4 Observation Model

5.4.1 Zero-Velocity Update

Generally, when the foot is in contact with the ground without slippage, its velocity in the *world*-frame should be zero. This condition can be formulated as

a measurement model [112] and incorporated into the Leg-IMU state estimation. However, it is important to note that only the velocity of the contact point on the foot is strictly zero, while other points on the foot may still be in motion. Since the Leg-IMU measures velocity at its mounting location rather than at the contact point, we must transform the velocity estimated by the Leg-IMU to the contact point.

The velocity of the contact point on the robot foot in the *world*-frame indicated by the Leg-IMU can be expressed as

$$\hat{\boldsymbol{v}}_f^w = \hat{\boldsymbol{v}}_{\text{limu}}^w + \hat{\mathbf{R}}_{\text{limu}}^w \left(\hat{\boldsymbol{\omega}}_{\text{limu}}\times\right) \hat{\boldsymbol{l}}_f^{\text{limu}}, \tag{5.4}$$

where $\hat{\boldsymbol{v}}_{\text{limu}}^w$ is the velocity estimates of the Leg-IMU in the *world*-frame; $\hat{\mathbf{R}}_{\text{limu}}^w$ is the rotation matrix of the Leg-IMU; $\hat{\boldsymbol{\omega}}_{\text{limu}}\times$ is the skewsymmetric matrix of the angular velocity measurement of the Leg-IMU; $\hat{\boldsymbol{l}}_f^{\text{limu}}$ is the position of the foot contact point in the Leg-IMU frame, as shown in Figure 5.1. This position misalignment between the Leg-IMU and the foot contact point can be measured in advance.

When the foot contacts the ground, the observed velocity of the contact point in the *world*-frame is

$$\tilde{\boldsymbol{v}}_f^w = \mathbf{0} - \boldsymbol{n}_{v_f^w}, \tag{5.5}$$

where $\boldsymbol{n}_{v_f^w}$ indicates the noise. Consequently, we can build the zero velocity measurement model as

$$\begin{aligned} \delta \boldsymbol{z}_v &= \hat{\boldsymbol{v}}_f^w - \tilde{\boldsymbol{v}}_f^w \\ &= \delta \boldsymbol{v}_{\text{limu}}^w + \mathbf{R}_{\text{limu}}^w \left(\hat{\boldsymbol{\omega}}_{\text{limu}}\times\right) \boldsymbol{\phi}_{\text{limu}} - \mathbf{R}_{\text{limu}}^w \left(\boldsymbol{l}_f^{\text{limu}}\times\right) \boldsymbol{b}_g, \end{aligned} \tag{5.6}$$

where $\delta \boldsymbol{v}_{\text{limu}}^w$, $\boldsymbol{\phi}_{\text{limu}}$, and $\boldsymbol{b}_g$ indicate the velocity error, attitude error, and gyroscope bias error of the Leg-IMU, respectively.

This zero-velocity update is applied to the Leg-IMUs only when foot contact is detected. However, if all IMUs, including the Body-IMU, report zero velocity, the robot is considered to be in a static state, and the zero-velocity update is performed for all IMUs.

5.4.2 Relative Position Constraints

In our proposed DogLegs system, all the Body-IMU and Leg-IMUs perform strapdown INS propagartion to predict their respective states. However, rather than moving independently, these IMUs are physically constrained within the robot's kinematic chain. Their relative positions can be computed in real time using joint encoder readings and known robot parameters. This information helps to constrain the error drift of individual IMUs, improving overall state estimation accuracy.

Since the multiple IMUs are not aligned together, each has its own independent initial reference frame in which the IMU pose is expressed. As shown in Figure 5.1, the *world*-frame is defined with the *robot*-frame at the initial position. Additionally, we assume the Body-IMU *body*-frame aligns with the *robot*-frame. Therefore, to determine the Leg-IMU pose in the *world*-frame, we must compute the transformation between the initial Leg-IMU frame and the initial Body-IMU frame.

Taking one Leg-IMU as an example, its transformation w.r.t. the *world*-frame is represented by the rotation matrix $\mathbf{R}^w_{\text{limu}}$ and the translation vector $\boldsymbol{p}^w_{\text{limu}}$. The roll and pitch components of $\mathbf{R}^w_{\text{limu}}$ can be determined by IMU data during the initial stationary period, while the translation $\boldsymbol{p}^w_{\text{limu}}$ can be obtained from leg kinematic model during the same period. For the yaw component of $\mathbf{R}^w_{\text{limu}}$, we assume it to be zero, given that the IMUs are carefully installed and aligned during setup. Therefore, given this transformation and considering the position of the foot contact point in the Leg-IMU frame $\hat{\boldsymbol{l}}^{\text{limu}}_f$, the Leg-IMU can estimate the position of the foot in the *world*-frame, denoted as $\hat{\boldsymbol{p}}^w_f$.

Consequently, the position of the Body-IMU in the *world*-frame, as estimated by the Leg-IMU, can be expressed as

$$\hat{\boldsymbol{p}}^w_b = \hat{\boldsymbol{p}}^w_f - \hat{\mathbf{R}}^w_b \hat{\boldsymbol{p}}^b_f, \tag{5.7}$$

where $\hat{\boldsymbol{p}}^w_b$ is the position of the Body-IMU in *world*-frame, and $\hat{\boldsymbol{p}}^b_f$ is the position of the foot in the *body*-frame. By rearranging the equation above, we obtain the relative position of the foot in the *body*-frame, indicated by the Leg-IMU and Body-IMU, as

$$\hat{\boldsymbol{p}}^b_f = \hat{\mathbf{R}}^b_w \left(\hat{\boldsymbol{p}}^w_f - \hat{\boldsymbol{p}}^w_b \right). \tag{5.8}$$

At the same time, for each leg of the robot, the position of the foot in the *body*-frame can be calculated by the leg kinematics model as

$$\tilde{\boldsymbol{p}}^b_f = g(\hat{\boldsymbol{\phi}}) = \boldsymbol{p}^b_f - \boldsymbol{n}_{p^b_f}, \tag{5.9}$$

where $\hat{\boldsymbol{\phi}}$ is the vector containing all joint angle measurements from the joint encoders of the leg, $g(\cdot)$ is the kinematics function of the leg, $\boldsymbol{p}^b_f$ indicates the true position of the foot in the *body*-frame, and $\boldsymbol{n}_{p^b_f}$ indicates the uncertainty of the relative position measurement, which is determined by the noise of the joint encoders. Consequently, we can build the relative position constraint model as

$$\delta \boldsymbol{z}_p = \hat{\boldsymbol{p}}^b_f - \tilde{\boldsymbol{p}}^b_f. \tag{5.10}$$

Whenever a Leg-IMU detects foot contact, we perform both zero velocity updates and relative position updates between that Leg-IMU and Body-IMU.

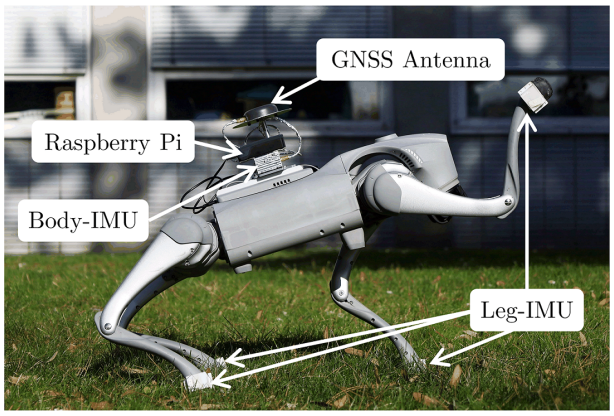


Figure 5.3: Experimental setup showcasing the Unitree Go2 quadruped robot and the devices used in the experiments.

5.5 Experimental Results

This section presents real world experimental results to support our key claims, namely, (i) Doglegs achieves better performance in state estimation compared to traditional EKF-based leg odometry across different terrains, and (ii) DogLegs reduces the error drift of the individual IMU systems by incorporating the relative position constraints between the multiple IMUs.

(a) Asphalt road. (b) Parking lots. (c) Construction site. (d) Offroad. (e) Grass.

Figure 5.4: The different experimental environments of the five sequences.

5.5.1 Experimental Setup

We conducted experiments using a Unitree Go2 quadruped, which was equipped with a Body-IMU, four Leg-IMUs, a GNSS/IMU integrated navigation system, and a raspberry pi for data collection, as shown in Figure 5.3. The Body-IMU and Leg-IMUs were of the same type as we used in our Wheel-INS experiments. The Leg-IMUs were mounted on the lower legs of the robot and close to the feet, while the Body-IMU was mounted on the robot body.

We collected datasets in five different outdoor environments, including asphalt road, parking lots, construction site, offroad, and grass. Figure 5.4 shows the five different environments. The robot was set to walking mode for all experiments. We performed a smoothed GNSS post-processed kinematic (PPK)/IMU integration method to compute the near-ground-truth trajectories for the first four sequences. However, for the grass experiment, GNSS signal were blocked by trees and buildings, rendering them unusable. Instead, we controlled the robot return to its starting point, allowing us to evaluate the loop closure error. The IMU and joint encoder data were collected at 200 Hz in all the experiments.

Table 5.1: Quantitative comparison on the pose accuracy in four sequences

Sequence	Method	RPE tra.	RPE rot.	APE tra.	APE rot.
Asphalt road	Foot-INS	1.21	0.31	6.09	2.54
	LegOdom	**0.70**	0.22	5.76	2.12
	DogLegs	**0.70**	**0.17**	**4.07**	**1.82**
Parking lots	Foot-INS	1.26	0.34	4.46	2.22
	LegOdom	**0.69**	0.22	**2.94**	2.01
	DogLegs	0.76	**0.21**	3.51	**1.91**
Construction site	Foot-INS	1.05	0.28	2.29	2.22
	LegOdom	0.73	0.20	2.15	2.33
	DogLegs	**0.68**	**0.18**	**2.01**	**2.05**
Offroad	Foot-INS	1.11	0.31	2.72	2.52
	LegOdom	**0.48**	0.17	2.47	1.83
	DogLegs	0.54	**0.12**	**2.04**	**1.45**

* RPE tra. and RPE rot. denote the relative translation error (%) and relative rotation error (deg/m), respectively. APE tra. and APE rot. denote the absolute translation error (m) and absolute rotation error (deg), respectively [37]. The best performance is highlighted in bold.

5.5.2 State Estimation Performance Evaluation

In this section, we analyze the pose estimation accuracy of our method and its generalizability across different terrains. We compare our method with two baselines:

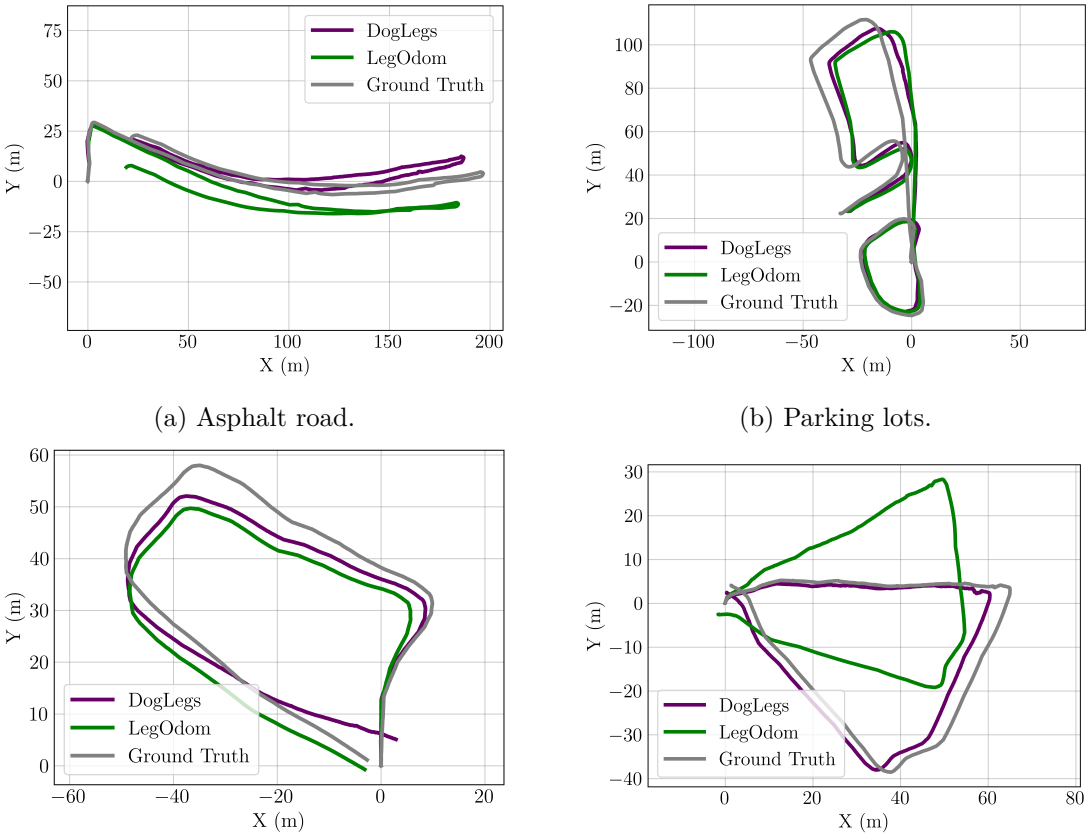
- **LegOdom**. The traditional EKF-based leg odometry[2] using a Body-IMU

[2] `https://github.com/YibinWu/leg-odometry`

and joint encoders [4, 5].

- **Foot-INS**. Single Foot-IMU based localization system [72]. See details in Section 2.2.

Because the Leg-IMUs have different roll and pitch angles to the main body, we assign the roll and pitch angles estimated by DogLegs to Foot-INS for a fair comparison. In addition, we report the mean value of the four individual systems on the four legs for Foot-INS. Table 5.1 lists the relative pose error (RPE) and absolute pose error (APE) calculated using evo [37] in four sequences. All the estimated trajectories were aligned with the ground truth using the transformation computed by evo [37] before metric evaluation.



(a) Asphalt road.

(b) Parking lots.

(c) Construction site.

(d) Offroad.

Figure 5.5: The estimated trajectories of LegOdom and our proposed DogLegs against ground truth in four sequences. As shown, DogLegs demonstrates more accurate trajectory estimation with reduced error drift compared to LegOdom.

Figure 5.5 compares the estimated trajectories of LegOdom and our proposed DogLegs against the ground truth across the four sequences. To illustrate the actual absolute error drift, in this figure, we didn't compute the transformation to align the whole estimated trajectories with the ground truth as done in evo [37].

As shown in the asphalt road sequence (a) and offroad sequence (d), DogLegs demonstrates significantly smaller error drift compared to LegOdom.

In the grass experiment, the GNSS signal was obstructed by trees and buildings, preventing the GNSS/IMU integrated navigation system from providing an accurate trajectory reference. Therefore, we evaluated the robot's loop closure error when it returned to the starting point. The results, shown in Figure 5.6, indicate that DogLegs achieves a loop closure distance error of approximately 0.1 m, which is smaller than half of LegOdom's error.

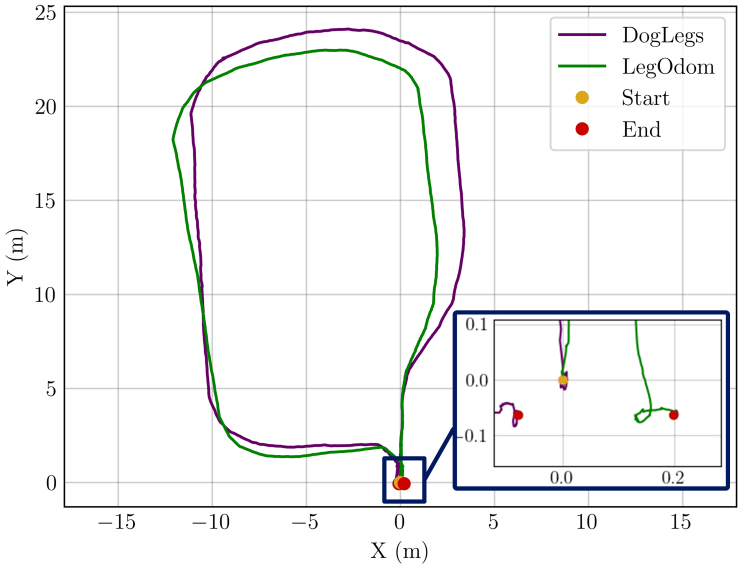


Figure 5.6: The estimated trajectories of LegOdom and our proposed DogLegs in the grass experiment. Because this area is surrounded by buildings and trees, GNSS could not provide useful positioning results. We evaluate the loop closure error in this experiment. As can be seen, DogLegs shows a smaller error than LegOdom when the robot returns to its starting point.

In addition to position and yaw angle, accurate estimation of roll and pitch angles is crucial for legged robots to maintain balance and execute dynamic movements effectively. Table 5.2 presents the statistic results of the roll and pitch angle errors for LegOdom and DogLegs across the four sequences. We report both the RMSE and maximum absolute error (MAX). From the table, we can see that overall, DogLegs exhibits lower roll and pitch estimation error than LegOdom. Moreover, DogLegs provides more stable roll and pitch estimates, as indicated by its smaller MAX error. For instance, in both the asphalt road and parking lots sequences, the maximum roll and pitch errors of LegOdom are around three times larger than those of DogLegs.

In conclusion, these experimental results support our first claim that Doglegs achieves better performance in state estimation compared to traditional EKF-based leg odometry across different terrains.

Table 5.2: Quantitative Comparison on the Roll and Pitch Accuracy in Four Sequences

Sequence	Method	Roll (°)		Pitch (°)	
		RMSE	**MAX**	**RMSE**	**MAX**
Asphalt road	LegOdom	2.79	5.95	**0.67**	5.41
	DogLegs	**0.59**	**1.46**	0.75	**1.36**
Parking lots	LegOdom	1.52	3.50	1.03	3.50
	DogLegs	**0.81**	**1.71**	**0.57**	**1.47**
Construction site	LegOdom	0.69	1.71	0.92	2.01
	DogLegs	**0.50**	**1.19**	**0.74**	**1.45**
Offroad	LegOdom	0.90	2.00	**0.63**	1.80
	DogLegs	**0.49**	**1.17**	0.85	**1.43**

5.5.3 Relative Position Constraints Evaluation

In this section, we evaluate the effectiveness of our proposed relative position constraint between multiple IMUs. Specifically, we compare the state estimation accuracy of DogLegs against individual IMU systems in the same sequence. This analysis supports our second claim: DogLegs reduces the error drift of the individual IMU systems by incorporating the relative position constraints between the multiple IMUs.

Figure 5.7 compares the trajectories of the individual Leg-IMU systems estimated by Foot-INS [72], namely, front-left foot (FL), front-right foot (FR), rear left foot (RL), rear right foot (RR) alongside DogLegs and the ground truth in the asphalt road experiment. The results show that the DogLegs trajectory falls approximately in the middle of the multiple independent Leg-IMU trajectories. Due to the random sensor errors inherent in IMUs, the individual Leg-IMU systems exhibit varying levels of error drift in different directions, even using the same type of sensor. By integrating diverse motion information from multiple IMUs, DogLegs effectively reduces random errors from individual IMU systems, thereby providing more accurate and stable state estimates overall.

5.5.4 Foot Contact Detection Evaluation

To evaluate the foot contact detection performance of our proposed DogLegs system, we compare its front-left foot contact detection results with those of the force threshold-based method in the asphalt road sequence. As shown in Figure 5.8, DogLegs detects more stable foot contact intervals, while force threshold-based

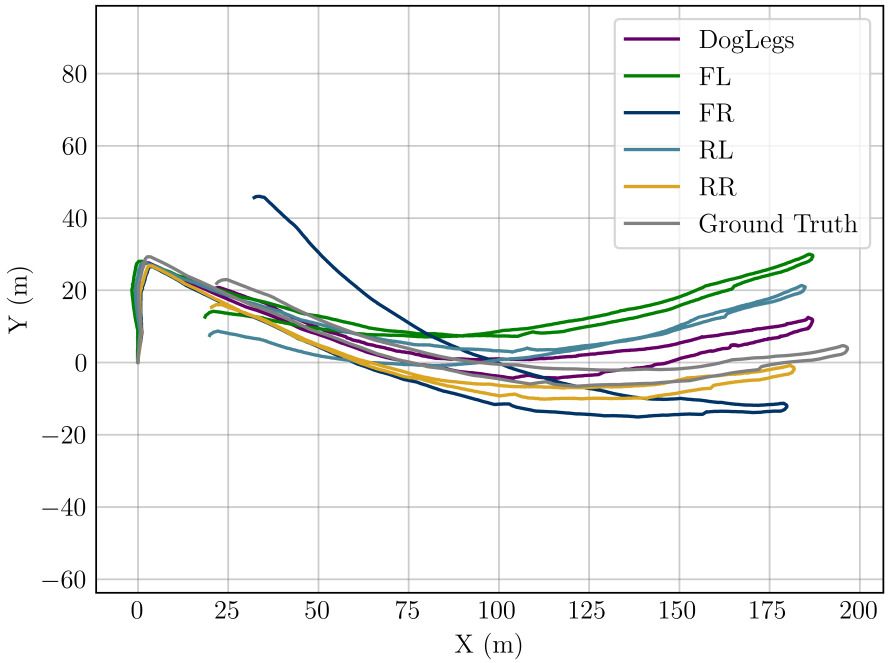


Figure 5.7: The estimated trajectories of the individual Leg-IMU systems (FL, FR, RL, RR) and our proposed DogLegs compared with ground truth in the asphalt road sequence.

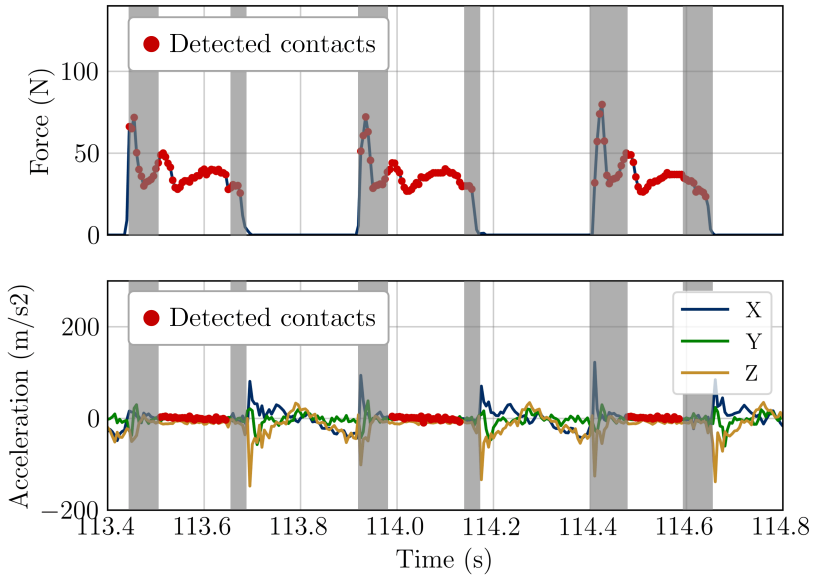


Figure 5.8: Detection results of front-left foot contact moments with the force threshold-based method and DogLegs in the asphalt road sequence. The upper and lower sub-figures show the detected results using a force sensor (with a threshold set to 20) and Leg-IMU accelerometer measurements, respectively. The gray areas indicate false positives (upper) and true negatives intervals (lower) in contact detection. As shown, our proposed DogLegs improves the robustness of contact detection by effectively rejecting outliers.

method introduces outliers due to its reliance on a single-dimensional force measurement.

5.6 Conclusion

Our goal in this study was to develop a robust proprioceptive state estimator for legged robots while minimizing additional hardware and computational costs. To this end, we proposed DogLegs, a framework that fuses the measurements from a Body-IMU, joint encoders, and multiple Leg-IMUs via an error-state EKF. Real world experiments have shown that DogLegs achieves better performance in state estimation compared to the traditional EKF-based leg odometry in different challenging terrains. In addition, DogLegs effectively reduces the error drift of the individual IMU systems by incorporating the relative position constraints between the multiple IMUs.

We emphasize that the IMU chips used in our system are commercial-grade components, costing approximately 3 US dollars only. Although in our current experiments, the IMU are attached to the exterior of the quadruped robot's lower legs, they can be seamlessly integrated into the internal structure without modifying the existing design. Additionally, power supply and data transmission are not limiting factors, as the lower legs already accommodate force sensors. Therefore, our approach can be directly applied to existing legged robots.

Chapter 6

Generic Robot State Estimation by LiDAR-Inertial Fusion

In previous two chapters, we have presented our proposed specific state estimation methods with specially mounted IMUs for both wheeled robots and legged robots. However, these methods are specially designed to take advantage of the special locomotion of the wheeled robots and legged robots, which are not applicable to more general platforms, such as handheld devices and aerial robots. In this chapter, we focus on more general aided-inertial state estimation algorithms that are not constrained by the robot motion model and can be applied to different platforms. To achieve this, we exploit the 3D spatial geometry information from LiDAR to aid IMU.

We motivated in our Wheel-INS and DogLegs studies that it is essential to improve the proprioceptive state estimation of the robot when exteroceptive sensors fail. However, integrating it with exteroceptive sensors is also crucial to further improve the overall state estimation accuracy and robustness. On one hand, such a reliable proprioceptive state estimation can provide accurate state prediction for the exteroceptive sensors, which can be used to improve the data association and reduce the number of outliers and consequently reduce the depends on the exteroceptive sensors. On the other hand, the exteroceptive sensors can be used to correct the drift of the proprioceptive state estimation system.

Note that in this study, both the LiDAR and IMU are both placed on the main body the robot. We use neither Wheel-IMUs and Leg-IMUs, thus it is not limited by special robot motion profile. However, it is possible to integrate LiDAR with Wheel-IMU and Leg-IMU to further improve the state estimation accuracy and robustness.

Specifically, we proposed a tightly-coupled LiDAR-inertial odometry (LIO) system named LIO-EKF that uses only the classical point-to-point registration and the EKF scheme. To further improve the generalizability and efficiency of

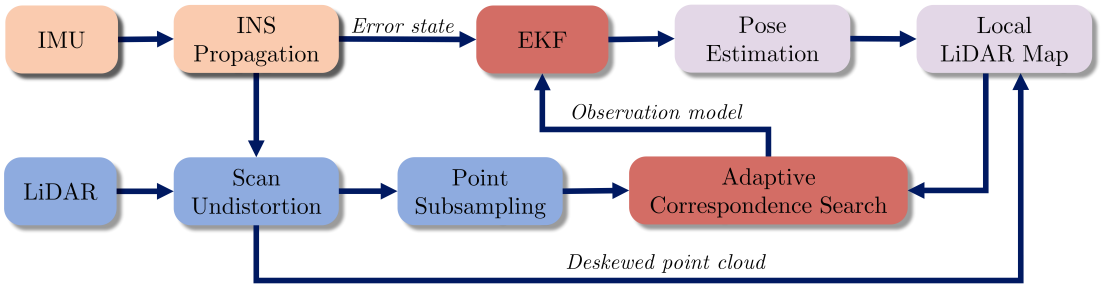


Figure 6.1: Overview of LIO-EKF. We integrate the most basic front end (point-to-point association) and back end (EKF) to build a simple, small yet accurate, generic, and high frequency LIO system.

the LiDAR-inertial fusion system, we proposed an adaptive data association that considers the relative pose uncertainty, the map discretization errors, and the LiDAR noise. In this way, we can substantially reduce the parameters to tune for a given type of platform and environment.

In sum, we make three key claims: LIO-EKF (i) is on par with the state-of-the-art LIO systems in terms of estimation accuracy, (ii) can provide an accurate pose estimate in different environments and vehicle motion profiles, (iii) can provide pose estimates at close-to-IMU frame rate. These claims are backed up by our experimental evaluation.

6.1 Overview

We aim to incrementally estimate the pose of a mobile robot by fusing sensor measurements obtained from a LiDAR scanner and an IMU in a tightly-coupled manner via EKF. We employ a conventional strapdown inertial navigation system (INS) for state prediction using the IMU readings. The predicted state is then corrected using the LiDAR scans in an observation model based on point-to-point residuals. To find reliable correspondences in the observation model, we propose an adaptive threshold that takes into account the predicted state uncertainty, map discretization errors, and LiDAR range noise. Figure 6.1 shows the overview of our proposed approach.

6.2 LiDAR Observation Model

We adopt the same strapdown INS model as used in Wheel-INS and DogLegs to predict the robot state. Meanwhile, we use the 15-dimension error state model used in DogLegs (Equation (5.3)) in our LIO-EKF pipeline.

LiDAR points are usually sampled sequentially while the sensor undergoes continuous motion. This procedure creates significant motion distortion influ-

encing the performance of the state estimation, especially when the motion is severe [116]. IMU can be used to mitigate this problem by providing high-frequency relative pose prediction within one LiDAR frame. Therefore, every time we process an incoming scan, we first deskew the point cloud of the scan using the IMU predicted pose. After scan deskewing, we employ the sub-sampling strategy proposed in KISS-ICP [97] to reduce the number of 3D points in the LiDAR point cloud for the sake of efficiency. We then transform the downsampled LiDAR frame to the robot body frame via the extrinsic calibration matrix between LiDAR and IMU. We construct a local map of the LiDAR point cloud using a voxel grid. Please refer to Vizzo et al. [97] for the details of map maintenance and update. The correspondences between the current downsampled LiDAR frame and the local map are computed via the nearest neighbor search using voxel hashing, resulting in a correspondence set:

$$\mathcal{C} = \{\, (i,j) \,:\, \min_{i,j} \|\boldsymbol{m}_i^w - \boldsymbol{q}_j^w\|_2 < \tau \}, \tag{6.1}$$

where $\boldsymbol{m}_i^w$ is the i-th point in the downsampled LiDAR cloud, which has been transformed to the *world*-frame by the extrinsic parameters between the LiDAR and IMU and the IMU pose, $\boldsymbol{q}_j^w$ is the j-th point in the local map, and τ is the maximum correspondence distance allowed for an inlier association.

We assume the extrinsic between the LiDAR and IMU is known and fixed. The observation model for the i-th measured point $\boldsymbol{m}_i$ is:

$$\begin{aligned} h_{ij}(\delta\boldsymbol{x}) &= \boldsymbol{m}_i^w - \boldsymbol{q}_j^w \\ &= (\mathbf{R}\boldsymbol{m}_i) \times \boldsymbol{\phi} + \delta\boldsymbol{p}, \end{aligned} \tag{6.2}$$

where $\mathbf{R}$, $\boldsymbol{p}$ represent the robot pose in the global frame after integrating the IMU readings between the previous and current LiDAR scan using Equation (3.2); $\boldsymbol{m}_i$ is the position of the i-th LiDAR point transformed into the IMU *body*-frame using the extrinsic parameters.

The Jacobian $\mathbf{J}_{ij}(\delta\boldsymbol{x})$ of $h_{ij}(\delta\boldsymbol{x})$ is:

$$\mathbf{J}_{ij}(\delta\boldsymbol{x}) = \begin{bmatrix} \mathbf{I} & \mathbf{0} & [\mathbf{R}\,\boldsymbol{m}_i]_\times & \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{3\times 15}. \tag{6.3}$$

We can then compute the error state vector and corresponding covariance using the alternative Kalman gain formulation proposed by Xu et al. [116]. In practice, to avoid explicitly forming the Kalman gain matrix, we rearrange the terms of the Kalman update equation as

$$\delta\boldsymbol{x} = \left(\mathbf{H}(\delta\boldsymbol{x}) + \hat{\mathbf{P}}^{-1}\right)^{-1} \boldsymbol{b}(\delta\boldsymbol{x}), \tag{6.4}$$

$$\mathbf{P} = \left(\mathbf{I} - \left(\mathbf{H}(\delta\boldsymbol{x}) + \hat{\mathbf{P}}^{-1}\right)^{-1} \mathbf{H}(\delta\boldsymbol{x})\right) \hat{\mathbf{P}}, \tag{6.5}$$

where

$$\mathbf{H}(\delta \boldsymbol{x}) = \sum_{i,j \in \mathcal{C}} \mathbf{J}_{ij}^{\top}(\delta \boldsymbol{x})\, \mathbf{R}_{p_i}^{-1}\, \mathbf{J}_{ij}(\delta \boldsymbol{x}), \tag{6.6}$$

$$\boldsymbol{b}(\delta \boldsymbol{x}) = \sum_{i,j \in \mathcal{C}} \mathbf{J}_{ij}^{\top}(\delta \boldsymbol{x})\, \mathbf{R}_{p_i}^{-1}\, h_{ij}(\delta \boldsymbol{x}), \tag{6.7}$$

and $\mathbf{R}_{p_i}$ is the observation noise covariance of point $\boldsymbol{p}_i$; $\hat{\mathbf{P}}$ is the predicted state covariance. In this way, we avoid storing and multiplying the complete Jacobian $\mathbf{J}(\delta \boldsymbol{x}) \in \mathbb{R}^{3n \times 15}$ of the n scan points.

Once the error state vector has been computed via Equation (6.4), the position, velocity, attitude, and biases at the current timestamp are corrected with the error state. After that, the error state is set to zero and subsequently estimated when a new LiDAR scan is integrated.

6.3 Adaptive Data Association

In Equation (6.1), we seek to establish associations between current LiDAR points and local map points through the nearest neighbor search. To mitigate the effect of outlier correspondences, we introduce a threshold denoted as τ; it limits the maximum allowed distance between corresponding points. Instead of determining this threshold empirically, our approach aims to automatically estimate τ by considering the uncertainty of the initial pose prediction, map discretization errors, and sensor noise.

6.3.1 Pose Prediction

The most critical factor in determining good correspondences is the quality of the pose prediction, namely the distance between the initial estimate and the real pose of the robot. Since, in our case, we compute the pose prediction using IMU measurements, we can compute the uncertainty of the relative pose between two successive scans by integrating the noise from the accelerometer and gyroscope over the time interval between the two LiDAR scans. We employ the strapdown INS detailed in Equation (3.2) for pose prediction. Consequently, we obtain a pose covariance matrix $_{r,k}$ that characterizes the uncertainty associated with the relative motion at time k. Our next step involves projecting this uncertainty into the space of point-to-point distances generated by the relative motion of the robot. These point-to-point distances can be approximated with an upper bound using the formula proposed by KISS-ICP [97]:

$$d(\mathbf{R}, \boldsymbol{t}) = 2\ r_{\max} \sin\left(\frac{1}{2}\boldsymbol{\Theta}(\mathbf{R})\right) + |\boldsymbol{t}|_2. \tag{6.8}$$

Here, $\boldsymbol{\Theta}(\mathbf{R})$ represents the angle corresponding to the rotation matrix $\mathbf{R}$, and $r_{\max}$ signifies the maximum range of the LiDAR. We can subsequently apply the unscented transform [47] in conjunction with Equation (6.8) to obtain the variance σ^2_{p2p} of the point-to-point distances from $\mathbf{P}_{r,k}$.

6.3.2 Map Discretization Errors

Our pipeline employs a voxel grid as the map representation, with a predetermined maximum number of points m per voxel. This choice speeds up the nearest neighbor search but introduces a discretization error in the voxel grid. We account for this error by modeling a Gaussian over the distances between a scan point and m points uniformly distributed on a surface patch contained in a voxel of size v. The variance of this Gaussian can be then computed as:

$$\sigma^2_{\text{map}} = \frac{v^2}{m}. \tag{6.9}$$

6.3.3 Sensor Noise

We incorporate sensor noise that affects the range measurements from the LiDAR using the variance σ^2_{range}.

The threshold τ at time k can then be computed by assuming the three Gaussians to be independent and employing the three-sigma bound:

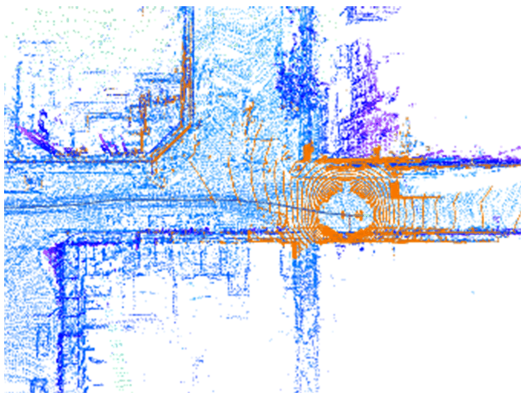
$$\tau = 3\sqrt{\sigma^2_{\text{p2p}} + \sigma^2_{\text{map}} + \sigma^2_{\text{range}}}. \tag{6.10}$$

6.4 Experimental Results

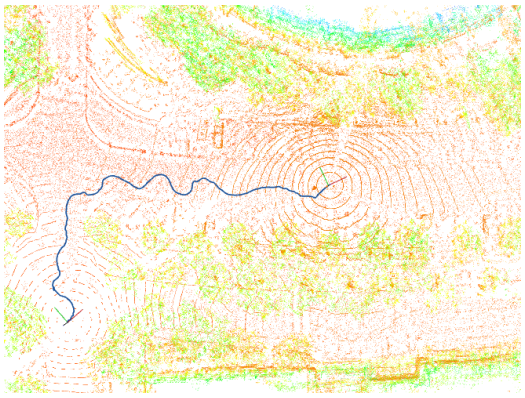
6.4.1 Experimental Setup

We compare LIO-EKF with two state-of-the-art LIO systems, FAST-LIO2 [115] and LIO-SAM [85] in four different datasets. FAST-LIO2 is a tightly-coupled iterated EKF-based LIO system using point-to-plane registration in the front-end, while LIO-SAM is a tightly-coupled factor graph-based LIO system using feature (line and plane) matching in the front-end. We used the default parameters in the open-source code of FAST-LIO2 and LIO-SAM, respectively. We disabled the loop closure module in LIO-SAM for the sake of fairness. The four datasets that we selected are:

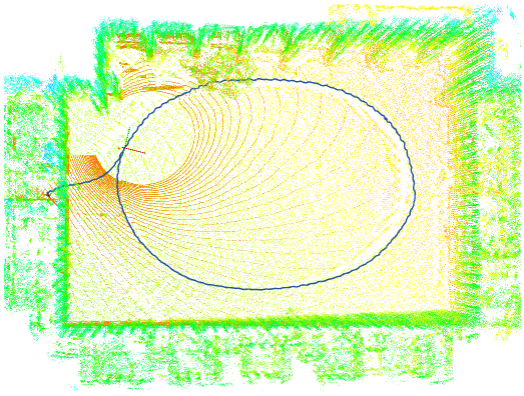
- *UrbanNav* [44] an autonomous driving dataset composed of three sequences recorded in Hong Kong.



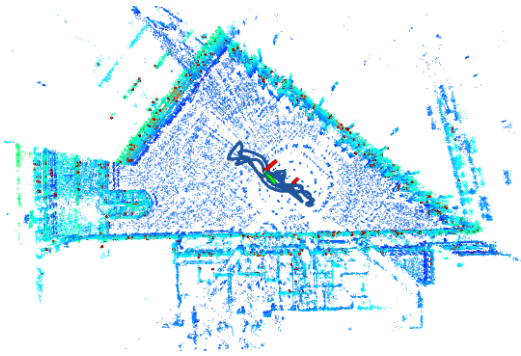
(a) UrbanNav.



(b) M2dgr.



(c) NewerCollege.



(d) NTU VIRAL.

Figure 6.2: Illustration of LIO-EKF running on three datasets within different environments using different platforms. The orange points represent the current downsampled LiDAR scan, while the others represent the local map accumulated over time.

- *M2dgr* [120] a dataset composed of seven sequences recorded with a wheeled mobile robot on a university campus in Shanghai.
- *NewerCollege* [81] a dataset composed of two sequences recorded with a handheld device on the Oxford university campus.
- *NTU VIRAL* [71] a dataset composed of three sequences recorded with a drone on a university campus at Singapore.

The IMUs used in the datasets are all consumer-grade MEMS IMUs. Please refer to the dateset websites for more details. To evaluate the different methods we use the Absolute Trajectory Error (ATE) [97] and the KITTI metric [31] for the relative error.

For all the following experiments, we use the same system configuration. The list of tunable parameters of our pipeline are:

- Max. points per voxel m

- Voxel size v
- Initial state covariance $\mathbf{P}_0$
- Point observation covariance $\mathbf{R}_{p_i}$

Notice that we do not consider as parameters the noise covariance of the IMU measurements and the process covariance related to the bias $\mathbf{Q}_k$, and the range noise σ^2_{range} as those can be determined from the sensors datasheets.

6.4.2 Pose Accuracy Evaluation

In the first experiment, we analyze the odometry accuracy of our method and its generalizability to different environments and motion profiles. In Table 6.1, we present the comparative results of LIO-EKF with the two baselines in all datasets. We average the results for the sequences "street_01" to "street_05" in *M2dgr* due to space limitation. Unfortunately, we were not able to run LIO-SAM on the NewerCollege dataset because additional attitude estimation output from the IMU is needed to initialize the system. Additionally, we were not able to evaluate the attitude accuracy in the NTU VIRAL dataset as the ground truth position was tracked with a total station, which did not provide attitude information.

Figure 6.2 illustrates LIO-EKF running with the four datasets. The quantitative results in Table 6.1 show that the proposed approach performs on par with the state-of-the-art LIO systems. This illustrates that, with a single configuration (without parameter fine-tuning), LIO-EKF can handle different motion profiles of the platform and different structures of the environment. Moreover, these results show that a classical EKF scheme is sufficient to fuse IMU and LiDAR data to estimate an accurate robot pose without relying on more complex state estimation schemes such as factor graphs or iterated EKF. This evaluation supports the first two claims made in this study.

6.4.3 Computation Efficiency Evaluation

In this section, we analyze the computational speed of our method. The results support our third claim that LIO-EKF can compute the robot's ego motion at close-to-IMU frequency. We compute the average processing time of the correction step of the filter once a new LiDAR scan has been received. We chose this because the correction step is the most time-consuming part of the pipeline, including scan preprocessing, data association, and pose update. Furthermore, we also compare the time of both FAST-LIO2 and LIO-SAM in terms of the scan processing in all the selected datasets by averaging the values over the different

Table 6.1: Quantitative results on the three different datasets

Sequence	Method	Avg. tra.	Avg. rot.	ATE. tra.	ATE. rot.
UrbanNav 20210517	FAST-LIO2	4.11	1.68	**17.62**	4.45
	LIO-SAM	**3.18**	**1.40**	20.74	**4.20**
	LIO-EKF	3.20	1.45	24.73	5.05
UrbanNav 20210518	FAST-LIO2	2.73	1.30	23.02	3.27
	LIO-SAM	2.52	1.31	**20.37**	**2.98**
	LIO-EKF	**2.20**	**1.14**	22.44	7.46
UrbanNav 20210521	FAST-LIO2	3.56	1.63	47.29	5.18
	LIO-SAM	**2.94**	1.62	**30.98**	4.51
	LIO-EKF	2.96	**1.54**	34.97	**4.47**
M2dgr street_01-05	FAST-LIO2	**1.62**	**0.79**	**5.05**	1.79
	LIO-SAM	3.15	1.52	10.19	4.27
	LIO-EKF	1.68	0.83	5.33	**1.70**
M2dgr street_06	FAST-LIO2	3.41	**1.55**	**8.93**	2.29
	LIO-SAM	3.65	1.65	9.04	2.41
	LIO-EKF	**3.37**	1.56	9.05	**2.27**
M2dgr street_08	FAST-LIO2	**1.10**	**1.65**	**2.12**	**1.71**
	LIO-SAM	3.73	5.78	4.21	6.36
	LIO-EKF	1.28	1.85	2.22	1.88
NewerCollege short_exp	FAST-LIO2	1.05	1.01	5.14	2.49
	LIO-EKF	**0.63**	**0.73**	**4.16**	**1.75**
NewerCollege long_exp	FAST-LIO2	1.09	1.33	6.33	4.22
	LIO-EKF	**0.74**	**0.91**	**5.14**	**2.34**
NTU VIRAL eee_01	FAST-LIO2	5.37	–	12.51	–
	LIO-EKF	**5.33**	–	**12.45**	–
NTU VIRAL nya_01	FAST-LIO2	4.58	–	**4.86**	–
	LIO-EKF	**4.56**	–	**4.86**	–
NTU VIRAL tnp_01	FAST-LIO2	**6.29**	–	11.55	–
	LIO-EKF	6.32	–	**11.52**	–

Avg. tra. and Avg. rot. denote the relative translation error (%) and relative rotation error (deg/m) using KITTI [31] metrics, respectively. ATE. tra. and ATE. rot. denote the absolute rotation error (deg) and absolute translation error (m) [97], respectively. The best performance is highlighted in bold.

sequences. All the methods have been run on a desktop machine with an Intel i7-10700 CPU at 2.90 GHz and 32 GB of RAM. The results are shown in Table 6.2. As we can see from the results, LIO-EKF achieves the fastest processing time for the scan processing, which is close to a regular IMU stream rate (100 Hz). In particular, it is about two times faster than FAST-LIO2 and four times faster than LIO-SAM in most scenarios. This is because our method uses a classical EKF scheme, which is very efficient as the optimization does not require multiple iterations.

Table 6.2: Comparison of the average processing time [ms]

	UrbanNav	M2dgr	NewerCollege
LIO-SAM	41.68	60.01	-
FAST-LIO2	24.03	29.96	9.42
LIO-EKF	**12.37**	**13.65**	**7.36**

Average processing time for the scan processing in all the selected datasets. The numbers are reported in milliseconds [ms].

6.4.4 Ablation Study

In this section, we perform two ablation studies to give better insights into the design choices of our pipeline. In particular, we show the impact of using the EKF scheme compared to an iterative EKF (IEKF) scheme with different numbers of iterations. Furthermore, we showcase that our novel adaptive data association threshold significantly impacts the generalization capabilities of our proposed LIO pipeline.

1) Multiple Iterations

We now investigate if multiple iterations can improve the odometry estimate of LIO-EKF. We replaced the used EKF scheme in our system with an IEKF, where we selected 10 and 100 max iterations for comparison. We use the "street_05" sequence of *M2dgr* and the "short_exp" sequence of *NewerCollege* for this ablation study. We compare the relative error according to the KITTI metric and the average processing time of the scan. The results are shown in Table 6.3.

Additionally, we compare the estimated trajectories of our proposed LIO-EKF with different numbers of iterations in the "street_01" sequence of *M2dgr* in Figure 6.3. We only align the estimated trajectories to the ground truth at the beginning to better show and compare the drift of the system.

Table 6.3: Comparison of LIO-EKF with different number of iterations

Sequence	# iterations	Avg. tra.	Avg rot.	Processing Time (ms)
M2dgr street_05	1	2.64	0.80	11.37
	10	2.61	0.77	35.10
	100	2.61	0.76	63.90
NewerCollege short_exp	1	0.63	0.73	12.41
	10	0.60	0.62	26.92
	100	0.55	0.58	84.21

Avg. tra. and Avg. rot. denote the relative translation error (%) and relative rotation error (deg/m) using KITTI metrics, respectively.

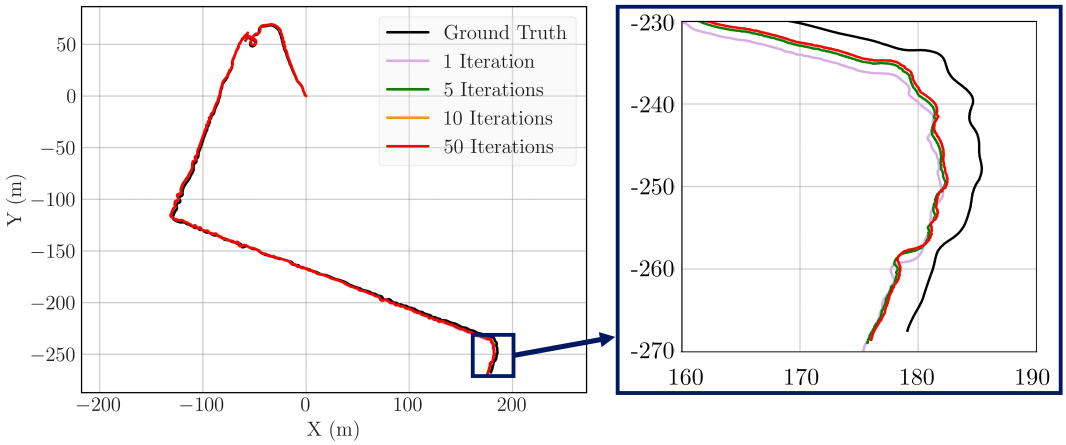


Figure 6.3: Comparison of LIO-EKF trajectory with different number of iterations in the "street_01" sequence of *M2dgr*. We set the maximum number of iterations to 1, 5, 10, and 50. It can be seen that there is no significant improvement in the trajectory accuracy with more iterations.

As we can see from Table 6.3, there is no significant improvement in the odometry accuracy with more iterations, with a maximum gap of 0.08% in relative translation error at 100 iterations in "short_exp". Conversely, the processing time is dramatically increased by a factor of 8. This can also be illustrated by Figure 6.3. The reasons are two-fold. First, IMU has excellent short-term stability, and it can provide an accurate pose prediction for every LiDAR scan so that the estimation can converge to an accurate pose after one update. Second, our error state formulation better handles the nonlinearity of the system dynamics. This shows that our design decision to use a classical EKF effectively computes the robot pose without relying on more complex state estimation schemes.

2) Adaptive Threshold

Table 6.4: Comparison of LIO-EKF with different threshold

Sequence	**Threshold** (m)	**Avg. tra.**	**Avg rot.**
UrbanNav 20210518	0.3	2.23	1.25
	1	**2.18**	**1.12**
	KISS-ICP	2.24	1.14
	Ours	2.20	1.14
UrbanNav 20210521	0.3	2.96	1.72
	1	2.95	1.55
	KISS-ICP	**2.95**	1.59
	Ours	2.96	**1.54**
M2dgr street_01	0.3	4.25	2.26
	1	1.46	0.86
	KISS-ICP	1.79	1.04
	Ours	**1.44**	**0.85**
M2dgr street_08	0.3	2.66	2.58
	1	1.33	1.93
	KISS-ICP	2.06	2.50
	Ours	**1.28**	**1.85**

Avg. tra. and Avg. rot. denote the relative translation error (%) and relative rotation error (deg/m) using KITTI metrics, respectively. In bold the best performing system configuration.

In the second ablation study, we show the generalization capabilities and effectiveness of the proposed adaptive threshold model. We compare fixed threshold settings and the adaptive threshold scheme proposed in KISS-ICP [97]. For this comparative analysis, we use two *UrbanNav* sequences and two *M2dgr* sequences. In this way, we aim to showcase that the proposed adaptive threshold can be generalized to different motion profiles. Table 6.4 lists the results.

We can see in Table 6.4 that the proposed adaptive threshold achieves the best generalization capability overall. To be specific, for *UrbanNav*, 1 m seems to be as good as the proposed. However, it is worse in the *M2dgr*. That means a handcrafted threshold cannot be guaranteed to work well with different vehicle motions in different environments. In addition, KISS-ICP [97] uses historical LiDAR registrations to estimate the average deviation between the constant velocity motion prediction and the corrected pose to indicate the threshold statistically. However, it is just an approximation of the motion prediction error, which cannot

represent the exact uncertainty of the initial pose guess. In the proposed algorithm, we directly use the uncertainty of the IMU-predicted initial pose, which is more accurate than the statistical model proposed in KISS-ICP. Furthermore, we consider the range noise of the LiDAR sensor and the map discretization error, which better models the data association problem.

6.5 Conclusion

This chapter presents LIO-EKF, a LiDAR-inertial odometry system based on a classical EKF scheme. Our approach exploits the classical point-to-point metric with an EKF to build a generic, tightly-coupled LIO system. Thanks to our error state formulation, we can better handle the non-linearities of the problem and converge to an accurate robot pose with a single correction step without relying on multiple iterations, as in IEKF schemes. Additionally, we propose a novel adaptive threshold model considering the pose uncertainty, map discretization error, and sensor noise for a more robust and effective data association.

Extensive real-world experiments conducted with different platforms in different environments show that LIO-EKF can achieve on par odometry performance compared to the more sophisticated state-of-the-art systems with a significantly more straightforward system design. Additionally, LIO-EKF can compute the pose at close to the IMU frame rate (100 Hz), which is much faster than other state-of-the-art methods. Furthermore, our proposed LIO-EKF achieved second place in the LiDAR-Inertial Track of the SLAM Challenge at the 2023 IEEE/CVF International Conference on Computer Vision (ICCV).

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Robust and accurate state estimation is the key for autonomous vehicles to achieve full autonomy. In particular, an IMU-based state estimation system is essential for providing high-frequency 6-DOF egomotion information that can be integrated with exteroceptive sensors, while also ensuring continuous pose estimation when other sensors fail due to environmental disturbances. The main contribution of this thesis are novel approaches that exploit different aids to improve the accuracy, efficiency, and robustness of the IMU-based state estimation, not only the proprioceptive odometry estimation, but also the fusion of IMU with other exteroceptive sensors. We divided the thesis into three chapters.

In Chapter 4, we presented novel state estimation algorithms specially designed for wheeled robots that use a Wheel-IMU to take advantage of the continuous rotation of the wheel.

First, we presented Wheel-INS in Section 4.1, a novel proprioceptive state estimation system for wheeled robots using only a Wheel-IMU. There are two major advantages of Wheel-INS: First, it achieves a similar information fusion scheme as ODO/INS by only one inertial sensor. Second, the continuous rotation of the Wheel-IMU autonomously and significantly reduces the heading drift. Extensive real-world experiments with different wheeled platforms illustrated that Wheel-INS achieved competitive pose estimation performance with only one IMU. Specifically, the positioning and heading accuracy of Wheel-INS had been improved by 23% and 15% respectively over ODO/INS. Moreover, benefiting from the rotation modulation, Wheel-INS showed significant resilience to the constant gyro bias error.

Then, in Section 4.2, we extended Wheel-INS to Wheel-SLAM, a novel SLAM system that uses the terrain features measured by the Wheel-IMU for loop closure

detection. The main contribution of our Wheel-SLAM is the implementation of such a SLAM system using only a Wheel-IMU, which is the first of its kind. We conducted real-world experiments to illustrate the feasibility of the idea to detect loop closure using the road bank angles measured by the Wheel-IMU. The results showed that the positioning and heading accuracy of Wheel-SLAM has been averagely improved by 52.6% and 53.2%, respectively, against Wheel-INS.

After that, we presented Wheel-GINS in Section 4.3, a GNSS/INS integration system built upon Wheel-INS. The main contribution of Wheel-GINS is the integration of GNSS with a Wheel-IMU for long-term state estimation. In this study, we not only achieved the fusion of GNSS and a Wheel-INS, but also estimated the Wheel-IMU installation parameters online. To accelerate the Wheel-IMU attitude misalignment error estimation, we incorporated a novel wheel angular velocity observation model. Real-world experimental results have illustrated that the proposed Wheel-GINS can achieve similar localization performance compared to the conventional ODO-GINS when GNSS is always available, while it significantly outperforms ODO-GINS during GNSS outages. Additionally, the Wheel-IMU installation parameters, including the Wheel-IMU leverarm and mounting angle and wheel radius scale error, can be effectively estimated online, thus improving the localization accuracy and the practicality of the system.

In Chapter 5, we presented a novel proprioceptive state estimation system (called DogLegs) for legged robots that fuses a Body-IMU, joint encoders, and multiple Leg-IMUs. The main contribution of this study is the use of Leg-IMU to detect foot ground contact and the integration of multiple IMUs with the forward kinematic model of the legged robot. Real world experiments have shown that DogLegs achieves better performance in state estimation compared to the traditional EKF-based leg odometry in different challenging terrains. In addition, DogLegs effectively reduces the error drift of the individual IMU systems by incorporating the relative position constraints between the multiple IMUs.

In Chapter 6, we presented LIO-EKF, a high-frequency yet accurate LiDADR-inertial odomtry system that adopts a simple and efficient design. We also proposed an empirical adaptive data association scheme to improve the generalizability of the system. The main contribution of LIO-EKF is that we showed that even with the basic point-to-point registration and EKF (no iterations) pipeline, we can achieve competitive performance compared to the state-of-the-art LiDAR-inertial odometry systems in different environments using different mobile platforms without parameter tuning. Additioinally, our LIO-EKF was more than two times faster than the state-of-the-art systems thanks to the simple and efficient design.

Overall, this thesis presented novel approaches to exploiting different aids to improve the accuracy and robustness of the IMU-based state estimation, not

only the proprioceptive odometry estimation, but also fusing IMU with other exteroceptive sensors, i.e., using IMU to aid other sensors. It showed promising results of the aided inertial navigation system and provided practical examples of how to exploit aid information from different mobile robots to improve the IMU-based state estimation system. We emphasize that the IMUs used in all the proposed approaches are low-cost MEMS sensors, which are widely available and can be easily integrated into different mobile platforms. Particularly, the Wheel-IMU and Leg-IMU used in Chapter 4 and Chapter 5 costs around three euros. We additionally released all the implementation of our methods presented in this thesis to benefit the community for facilitating further research.

7.2 Future Work

In this thesis, we proposed several aided-inertial navigation systems to improve the accuracy, efficiency, and robustness of the state estimation for various types of mobile robots. Despite the promising results, there remain several directions for future research that could further enhance the performance, generality, and applicability of the proposed methods.

First, for the proposed Wheel-INS system introduced in Section 4.1, future work can focus on advancing information fusion techniques to further improve localization accuracy and consistency. In the current implementation, the robot velocity is computed by the Wheel-IMU's angular velocity and the wheel radius, which introduces correlation with the system model of the EKF and may lead to inconsistency since this violates the EKF's independence assumption. To address this, advanced nonlinear filtering techniques such as particle filters can be investigated, as they can better handle system nonlinearities and correlated uncertainties, potentially improving both accuracy and consistency.

Another promising direction is to detect and handle wheel slippage [33], which can degrade the performance of Wheel-INS. Learning-based approaches could be used to identify slippage events directly from Wheel-IMU measurements. Once a slippage is detected, adaptive strategies such as dynamically adjusting the covariance of the velocity measurement model could be applied to mitigate its effects. Training end-to-end models to predict the velocity measurement covariance from Wheel-IMU data [6] also represent a promising hybrid design that combines model-based and learning-based advantages for enhanced robustness in challenging environments.

For the proposed Wheel-SLAM system presented in Section 4.2, future improvements can be made in loop closure detection. The current particle-based approach relies on the trajectory and grid map to detect potential loop closures, which can be ineffective in large-scale environments or with a limited number of

particles. To address this, global search-based matching methods can be incorporated to improve robustness and success rate [8, 80] of loop closure detection. To further improve the efficiency and accuracy, frequency domain matching techniques can be explored. For example, transforming roll angle sequences into the frequency domain using methods such as the fast Fourier transform (FFT) [29] can provide more efficient and discriminative matching features compared to time-domain correlation.

Additionally, the current approach focuses on estimating only the latest robot state without optimizing historical states, namely, bundle adjustment. Incorporating factor graph optimization frameworks [24, 104] would allow joint optimization of the entire trajectory and map, resulting in improved consistency and accuracy of both state estimation and terrain reconstruction.

In the proposed Wheel-GINS system (Section 4.3), we directly fuse the 2D localization from Wheel-INS with the global 3D GNSS position measurements. This can introduce inconsistencies in height and pitch angle estimation, particularly when the robot travels on roads with steep inclines or declines. A promising future direction is to introduce a virtual pitch angle propagation model to capture the pitch variation within the Wheel-INS framework. The parameters of this virtual model could be jointly estimated within the EKF, leading to more consistent 3D localization results.

Beyond improving the accuracy and efficiency of the proprioceptive state estimation, integrating the proposed Wheel-IMU systems with exteroceptive sensors such as cameras or LiDARs is also an important research direction. Since Wheel-INS already provides accurate odometry in real time with a low cost IMU, the system can operate with lightweight visual or LiDAR modules that only store sparse keyframes for loop closure detection (without the complex computation for visual odometry or LiDAR odometry). This approach enables low-cost yet efficient navigation for robots. Notably, Dijk et al. [96] demonstrated that even tiny robots with limited computational resources can navigate autonomously using online odometry estimation and visual route-following. Therefore, combining Wheel-INS with visual route-following or lightweight exteroceptive perception methods could lead to highly efficient and low-cost navigation systems for wheeled robots.

For the proposed multiple Leg-IMUs based state estimation system for legged robots presented in Chapter 5, one key direction is to develop online calibration methods to estimate the extrinsic parameters between the Leg-IMUs and robot limbs. In practice, Leg-IMUs cannot be perfectly aligned with the leg frames or precisely positioned at the foot ends. Real-time calibration of these offsets can significantly improve the system's accuracy and usability. Another research avenue is to model and incorporate the uncertainty of foot contact detection [66]

into the state estimation framework, thereby improving the robustness of state estimation in rough and slippery terrains.

Additionally, leveraging Leg-IMU data to detect foot slippage and missteps in real time, and feeding this information into the robot's control loop [103], could greatly enhance the stability and adaptability of legged robots in complex terrains. Given that IMUs measure full 6-DoF motion, they can effectively detect slippage events and enable legged robots to respond adaptively in challenging conditions.

Finally, for the LIO-EKF system introduced in Chapter 6, future work could focus on developing a hybrid front-end that combines point-to-point and point-to-plane registration methods [56]. The current implementation uses pure point-to-point registration for efficiency and generality, but at some cost to accuracy. Incorporating point-to-plane registration selectively, for points lying on planar surfaces such as walls, could achieve a better balance between computational efficiency and estimation accuracy.

Bibliography

[1] K. Abdul Rahim. *Heading drift mitigation for low-cost inertial pedestrian navigation*. PhD thesis, Department of Civil Engineering, University of Nottingham, Nottingham, UK, 2012.

[2] M. Angermann and P. Robertson. FootSLAM: Pedestrian simultaneous localization and mapping without exteroceptive sensors-hitchhiking on human perception and cognition. *Proc. of the IEEE*, 100:1840–1848, 2012.

[3] A. Angrisano. *GNSS/INS integration methods*. PhD thesis, Department of Applied Sciences, Parthenope University of Naples, Naples, Italy, 2010.

[4] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart. State estimation for legged robots on unstable and slippery terrain. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6058–6064, 2013.

[5] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart. State estimation for legged robots: Consistent fusion of leg kinematics and IMU. In *Proc. of Robotics: Science and Systems*, pages 17–24, 2013.

[6] M. Brossard, A. Barrau, and S. Bonnabel. AI-IMU dead-reckoning. *IEEE Transactions on Intelligent Vehicles*, 5(4):585–595, 2020.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. 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.

[8] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. 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.

[9] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini. Probabilistic contact estimation and impact detection for state estimation of quadruped robots. *IEEE Robotics and Automation Letters*, 2(2):1023–1030, 2017.

[10] M. Camurri, M. Ramezani, S. Nobili, and M. Fallon. Pronto: A multi-sensor state estimator for legged robots in real-world scenarios. *Frontiers in Robotics and AI*, 7, 2020.

[11] L. Carlone, A. Kim, T. Barfoot, D. Cremers, and F. Dellaert, editors. *SLAM Handbook. From Localization and Mapping to Spatial Intelligence*. Cambridge University Press, 2025.

[12] J. Chang, Y. Zhang, S. Fan, F. Huang, D. Xu, and L.-T. Hsu. An anti-spoofing model based on MVM and MCCM for a loosely-coupled GNSS/INS/LiDAR Kalman filter. *IEEE Transactions on Intelligent Vehicles*, 9(1):1744–1755, 2024.

[13] L. Chang, X. Niu, T. Liu, J. Tang, and C. Qian. GNSS/INS/LiDAR-SLAM integrated navigation system based on graph optimization. *Remote Sensing*, 11(9), 2019.

[14] C. Chen, X. Lu, A. Markham, and N. Trigoni. Ionet: Learning to cure the curse of drift in inertial odometry. *Proc. of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[15] C. Chen and X. Pan. Deep learning for inertial positioning: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 25(9):10506–10523, 2024.

[16] Q. Chen, H. Lin, J. Kuang, Y. Luo, and X. Niu. Rapid initial heading alignment for MEMS land vehicular GNSS/INS navigation system. *IEEE Sensors Journal*, 23(7):7656–7666, 2023.

[17] Q. Chen, X. Niu, J. Kuang, and J. Liu. IMU mounting angle calibration for pipeline surveying apparatus. *IEEE Transactions on Instrumentation and Measurement*, 69(4):1765–1774, 2019.

[18] Q. Chen, Q. Zhang, and X. Niu. Estimate the pitch and heading mounting angles of the IMU for land vehicular GNSS/INS integrated system. *IEEE Transactions on Intelligent Transportation Systems*, 22(10):6503–6515, 2021.

[19] C. Chi, X. Zhang, J. Liu, Y. Sun, Z. Zhang, and X. Zhan. GICI-LIB: A GNSS/INS/camera integrated navigation library. *IEEE Robotics and Automation Letters*, 8(12):7970–7977, 2023.

[20] J. Collin. MEMS IMU carouseling for ground vehicles. *IEEE Transactions on Vehicular Technology*, 64(6):2242–2251, 2014.

[21] J. Collin, M. Kirkko-Jaakkola, and J. Takala. Effect of carouseling on angular rate sensor error processes. *IEEE Transactions on Instrumentation and Measurement*, 64(1):230–240, 2014.

[22] E. H. Coulter, P. M. Dall, L. Rochester, J. P. Hasler, and M. H. Granat. Development and validation of a physical activity monitor for use on a wheelchair. *Spinal Cord*, 49(3):445–450, 2011.

[23] B. Della Corte, I. Bogoslavskyi, C. Stachniss, and G. Grisetti. A general framework for flexible multi-cue photometric point cloud registration. In *Proc. of the IEEE International Conference on Robotics and Automation*, 2018.

[24] F. Dellaert and G. Contributors. borglab/gtsam, May 2022.

[25] L. Di Giammarino, L. Brizi, T. Guadagnino, C. Stachniss, and G. Grisetti. MD-SLAM: Multi-Cue Direct SLAM. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[26] G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte. The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. *IEEE Transactions on Robotics and Automation*, 17(5):731–747, 2001.

[27] P. Djuric, J. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. Bugallo, and J. Miguez. Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38, 2003.

[28] S. Du. *Rotary inertial navigation system with a low-cost MEMS IMU and its integration with GNSS*. PhD thesis, Department of Geomatics Engineering, University of Calgary, Calgary, Canada, 2015.

[29] P. Duhamel and M. Vetterli. Fast fourier transforms: A tutorial review and a state of the art. *Signal Processing*, 19(4):259–299, 1990.

[30] E. Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

[31] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[32] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang. OpenVINS: A research platform for visual-inertial estimation. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 4666–4672, 2020.

[33] C. L. Gentil, D. Lisus, and T. D. Barfoot. Do we still need to work on odometry for autonomous driving? *arXiv preprint arXiv:2505.04438*, 2025.

[34] B. Gersdorf and U. Freese. A Kalman filter for odometry using a wheel mounted inertial sensor. In *Proc. of the International Conference on Informatics in Control, Automation and Robotics*, 2013.

[35] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[36] P. D. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, London, United Kingdom, 2013.

[37] M. Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017.

[38] T. Guadagnino, X. Chen, M. Sodano, J. Behley, G. Grisetti, and C. Stachniss. Fast Sparse LiDAR Odometry Using Self-Supervised Feature Selection on Intensity Images. *IEEE Robotics and Automation Letters*, 7(3):7597–7604, 2022.

[39] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle. Contact-aided invariant extended Kalman filtering for robot state estimation. *International Journal of Robotics Research*, 39(4):402–430, 2020.

[40] R. Hartley, M. G. Jadidi, J. Grizzle, and R. M. Eustice. Contact-aided invariant extended Kalman filtering for legged robot state estimation. In *Proc. of Robotics: Science and Systems*, 2018.

[41] V. Havyarimana, D. Hanyurwimfura, P. Nsengiyumva, and Z. Xiao. A novel hybrid approach based-SRG model for vehicle position prediction in multi-GPS outage conditions. *Information Fusion*, 41:1–8, 2018.

[42] V. Havyarimana, Z. Xiao, A. Sibomana, D. Wu, and J. Bai. A fusion framework based on sparse gaussian–wigner prediction for vehicle localization using GDOP of GPS satellites. *IEEE Transactions on Intelligent Transportation Systems*, 21(2):680–689, 2020.

[43] S. Herath, D. Caruso, C. Liu, Y. Chen, and Y. Furukawa. Neural inertial localization. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6604–6613, June 2022.

[44] L.-T. Hsu, F. Huang, H.-F. Ng, G. Zhang, Y. Zhong, X. Bai, and W. Wen. Hong Kong UrbanNav: An open-source multisensory dataset for benchmarking urban navigation algorithms. *NAVIGATION: Journal of the Institute of Navigation*, 70(4), 2023.

[45] T. Hua, L. Pei, T. Li, J. Yin, G. Liu, and W. Yu. M2C-GVIO: motion manifold constraint aided GNSS-visual-inertial odometry for ground vehicles. *Satellite Navigation*, 4(13), 2023.

[46] G. Huang. Visual-inertial navigation: A concise review. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 9572–9582, 2019.

[47] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

[48] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the bayes tree. *International Journal of Robotics Research*, 31(2):216–235, 2012.

[49] C. Kilic, J. N. Gross, N. Ohi, R. Watson, J. Strader, T. Swiger, S. Harper, and Y. Gu. Improved planetary rover inertial navigation and wheel odometry performance through periodic use of zero-type constraints. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 552–559, 2019.

[50] J.-H. Kim, S. Hong, G. Ji, S. Jeon, J. Hwangbo, J.-H. Oh, and H.-W. Park. Legged robot state estimation with dynamic contact event information. *IEEE Robotics and Automation Letters*, 6(4):6733–6740, 2021.

[51] Y. Kim, B. Yu, E. M. Lee, J.-h. Kim, H.-w. Park, and H. Myung. STEP: State estimator for legged robots using a preintegrated foot velocity factor. *IEEE Robotics and Automation Letters*, 7(2):4456–4463, 2022.

[52] H. Kolvenbach, D. Wisth, R. Buchanan, G. Valsecchi, R. Grandia, M. Fallon, and M. Hutter. Towards autonomous inspection of concrete deterioration in sewers with legged robots. *Journal of Field Robotics*, 37(8):1314–1327, 2020.

[53] J. Kuang, D. Xia, Y. Wang, X. Meng, and X. Niu. A shin-mounted inertial navigation system for pedestrian walking and running gait. *IEEE Sensors Journal*, pages 1–10, 2025.

[54] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[55] D. Lee, M. Jung, W. Yang, and A. Kim. Lidar odometry survey: recent advancements and remaining challenges. *Intelligent Service Robotics*, 17(2):95–118, 2024.

[56] D. Lee, H. Lim, and S. Han. Genz-icp: Generalizable and degeneracy-robust lidar odometry using an adaptive weighting. *IEEE Robotics and Automation Letters*, 10(1):152–159, 2025.

[57] K. Li, M. Li, and U. D. Hanebeck. Towards high-performance solid-state-lidar-inertial odometry and mapping. *IEEE Robotics and Automation Letters*, 6(3):5167–5174, 2021.

[58] S. Li, S. Wang, Y. Zhou, Z. Shen, and X. Li. Tightly coupled integration of GNSS, INS, and LiDAR for vehicle navigation in urban environments. *IEEE Internet of Things Journal*, 9(24):24721–24735, 2022.

[59] X. Li, S. Li, Y. Zhou, Z. Shen, X. Wang, X. Li, and W. Wen. Continuous and precise positioning in urban environments by tightly coupled integration of GNSS, INS and vision. *IEEE Robotics and Automation Letters*, 7(4):11458–11465, 2022.

[60] Y. Li, R. Chen, X. Niu, Y. Zhuang, Z. Gao, X. Hu, and N. El-Sheimy. Inertial sensing meets machine learning: Opportunity or challenge? *IEEE Transactions on Intelligent Transportation Systems*, 23(8):9995–10011, 2022.

[61] J. Lin and F. Zhang. Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3126–3131, 2019.

[62] T.-Y. Lin, R. Zhang, J. Yu, and M. Ghaffari. Legged robot state estimation using invariant kalman filtering and learned contact events. In *Proc. of Conference on Robot Learning*, 2021.

[63] X. Liu, W. Wen, and L.-T. Hsu. GLIO: Tightly-coupled GNSS/LiDAR/IMU integration for continuous and drift-free state estimation of intelligent vehicles in urban areas. *IEEE Transactions on Intelligent Vehicles*, 9(1):1412–1422, 2024.

[64] H.-A. Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, 2004.

[65] A. Macario Barros, M. Michel, Y. Moline, G. Corre, and F. Carrel. A comprehensive survey of visual slam algorithms. *Robotics*, 11(1), 2022.

[66] M. Maravgakis, D.-E. Argiropoulos, S. Piperakis, and P. Trahanias. Probabilistic contact state estimation for legged robots using inertial information. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 12163–12169, 2023.

[67] R. D. Martini. GPS/INS sensing coordination for vehicle state identification and road grade positioning. Master's thesis, Department of Mechanical and Nuclear Engineering, The Pennsylvania State University, Philadelphia, U.S., 2006.

[68] C. Merfels and C. Stachniss. Pose fusion with chain pose graphs for automated driving. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3116–3123, 2016.

[69] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. of the AAAI Conference on Artificial Intelligence*, pages 593–598, 2002.

[70] K. Murphy and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Sequential Monte Carlo methods in practice*, pages 499–515. Springer, 2001.

[71] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie. Ntu viral: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint. *The International Journal of Robotics Research*, 41(3):270–280, 2022.

[72] X. Niu, T. Liu, J. Kuang, Q. Zhang, and C. Guo. Pedestrian trajectory estimation based on foot-mounted inertial navigation system for multi-

story buildings in postprocessing mode. *IEEE Internet of Things Journal*, 9(9):6879–6892, 2022.

[73] X. Niu, S. Nassar, and N. EI-Sheimy. An accurate land-vehicle MEMS IMU/GPS navigation system using 3d auxiliary velocity updates. *NAVIGATION: Journal of the Institute of Navigation*, 54(3):177–188, 2007.

[74] X. Niu, Y. Peng, Y. Dai, Q. Chen, C. Guo, and Q. Zhang. Camera-based lane-aided multi-information integration for land vehicle navigation. *IEEE Transactions on Mechatronics*, 28(1):152–163, 2023.

[75] X. Niu, H. Tang, T. Zhang, J. Fan, and J. Liu. IC-GVINS: A robust, real-time, INS-centric GNSS-visual-inertial navigation system. *IEEE Robotics and Automation Letters*, 8(1):216–223, 2023.

[76] X. Niu, Y. Wu, and J. Kuang. Wheel-INS: A wheel-mounted MEMS IMU-based dead reckoning system. *IEEE Transactions on Vehicular Technology*, 70(10):9814–9825, 2021.

[77] G. Ou, D. Li, and H. Li. Leg-KILO: Robust kinematic-inertial-lidar odometry for dynamic legged robots. *IEEE Robotics and Automation Letters*, 9(10):8194–8201, 2024.

[78] W. Ouyang, Y. Wu, and H. Chen. INS/odometer land navigation by accurate measurement modeling and multiple-model adaptive estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 57(1):245–262, 2021.

[79] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu. LINS: A lidar-inertial state estimator for robust and efficient navigation. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 8899–8906, 2020.

[80] T. Qin, P. Li, and S. Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[81] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon. The newer college dataset: Handheld lidar, inertial and vision with ground truth. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4353–4360, 2020.

[82] P. Robertson, M. G. Puyol, and M. Angermann. Collaborative pedestrian mapping of buildings using inertial sensors and footslam. In *Proc. of the ION GNSS*, pages 1366–1377, 2011.

[83] P. G. Savage. *Strapdown Analytics, Part 1.* Strapdown Associates, Minnesota, United States of America, 2000.

[84] T. Shan and B. Englot. LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4758–4765, 2018.

[85] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus. LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5135–5142, 2020.

[86] E.-H. Shin. *Estimation techniques for low-cost inertial navigation.* PhD thesis, Department of Geomatics Engineering, University of Calgary, Calgary, Canada, 2005.

[87] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to autonomous mobile robots.* MIT press, Massachusetts, USA, 2011.

[88] I. Skog, P. Handel, J.-O. Nilsson, and J. Rantakokko. Zero-velocity detection—an algorithm evaluation. *IEEE Transactions on Biomedical Engineering*, 57(11):2657–2666, 2010.

[89] J. Solà. Quaternion kinematics for the error-state KF. *Laboratoire dAnalyse et dArchitecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Technical Report*, page 35, 2012.

[90] S. Sukkarieh. *Low cost, high integrity, aided inertial navigation systems for autonomous land vehicles.* PhD thesis, Department of Mechanical and Mechatronic Engineering, University of Sydney, Sydney, Australia, 2000.

[91] T. Suzuki. Attitude-estimation-free GNSS and IMU integration. *IEEE Robotics and Automation Letters*, 9(2):1090–1097, 2024.

[92] C. Tan. Dead reckoning localization using a single wheel-mounted IMU with the simultaneous estimation of mounting parameters. *IEEE Sensors Journal*, 24(3):3797–3810, 2024.

[93] S. Teng, M. W. Mueller, and K. Sreenath. Legged robot state estimation in slippery environments using invariant extended Kalman filter with velocity update. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3104–3110, 2021.

[94] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[95] D. Titterton and J. Weston. *Strapdown Inertial Navigation Technology*. The Institution of Engineering and Technology, London, United Kingdom, 2004.

[96] T. van Dijk, C. D. Wagter, and G. C. H. E. de Croon. Visual route following for tiny autonomous robots. *Science Robotics*, 9(92), 2024.

[97] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. 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, 2023.

[98] J. Wahlström and I. Skog. Fifteen years of progress at zero velocity: A review. *IEEE Sensors Journal*, 21(2):1139–1151, 2021.

[99] D. Wang, Y. Dong, Z. Li, Q. Li, and J. Wu. Constrained MEMS-based GNSS/INS tightly coupled system with robust Kalman filter for accurate land vehicular navigation. *IEEE Transactions on Instrumentation and Measurement*, 69(7):5138–5148, 2020.

[100] H. Wang, C. Wang, C.-L. Chen, and L. Xie. F-LOAM: Fast lidar odometry and mapping. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 4390–4396, 2021.

[101] W. Wen, X. Bai, Y. C. Kan, and L.-T. Hsu. Tightly coupled GNSS/INS integration via factor graph and aided by fish-eye camera. *IEEE Transactions on Vehicular Technology*, 68(11):10651–10662, 2019.

[102] W. Wen, T. Pfeifer, X. Bai, and L.-T. Hsu. Factor graph optimization for GNSS/INS integration: A comparison with the extended Kalman filter. *NAVIGATION: Journal of the Institute of Navigation*, 68(2):315–331, 2021.

[103] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. D. Prete. Optimization-based control for dynamic legged robots. *IEEE Transactions on Robotics*, 40:43–63, 2024.

[104] D. Wisth, M. Camurri, and M. Fallon. Robust legged robot state estimation using factor graph optimization. *IEEE Robotics and Automation Letters*, 4(4):4507–4514, 2019.

[105] D. Wisth, M. Camurri, and M. Fallon. VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots. *IEEE Transactions on Robotics*, 39(1):309–326, 2023.

[106] Y. Wu, C. Goodall, and N. El-Sheimy. Self-calibration for imu/odometer land navigation: Simulation and test results. In *Proc. of the International Technical Meeting of The Institute of Navigation*, pages 839–849, 2010.

[107] Y. Wu, T. Guadagnino, L. Wiesmann, L. Klingbeil, C. Stachniss, and H. Kuhlmann. LIO-EKF: High frequency LiDAR-inertial odometry using extended Kalman filters. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 13741–13747, 2024.

[108] Y. Wu, J. Kuang, S. Khorshidi, X. Niu, L. Klingbeil, M. Bennewitz, and H. Kuhlmann. DogLegs: Robust proprioceptive state estimation for legged robots using multiple leg-mounted IMUs. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025.

[109] Y. Wu, J. Kuang, and X. Niu. Wheel-INS2: Multiple MEMS IMU-based dead reckoning system with different configurations for wheeled robots. *IEEE Transactions on Intelligent Transportation Systems*, 24(3):3064–3077, 2023.

[110] Y. Wu, J. Kuang, X. Niu, J. Behley, L. Klingbeil, and H. Kuhlmann. Wheel-SLAM: Simultaneous localization and terrain mapping using one wheel-mounted IMU. *IEEE Robotics and Automation Letters*, 8(1):280–287, 2023.

[111] Y. Wu, J. Kuang, X. Niu, C. Stachniss, L. Klingbeil, and H. Kuhlmann. Wheel-GINS: A GNSS/INS integrated navigation system with a wheel-mounted imu. *IEEE Transactions on Intelligent Transportation Systems*, 26(5):6891–6903, 2025.

[112] Y. Wu, X. Niu, and J. Kuang. A comparison of three measurement models for the wheel-mounted MEMS IMU-based dead reckoning system. *IEEE Transactions on Vehicular Technology*, 70(11):11193–11203, 2021.

[113] Y. Wu, M. Wu, X. Hu, and D. Hu. Self-calibration for land navigation using inertial sensors and odometer: Observability analysis. In *Proc. of the AIAA Guidance, Navigation, and Control Conference*, 2009.

[114] Z. Xiao, Y. Chen, M. Alazab, and H. Chen. Trajectory data acquisition via private car positioning based on tightly-coupled GPS/OBD integration in urban environments. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):9680–9691, 2022.

[115] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang. FAST-LIO2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[116] W. Xu and F. 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.

[117] S. Yang, Z. Zhang, B. Bokser, and Z. Manchester. Multi-IMU proprioceptive odometry for legged robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 774–779, 2023.

[118] S. Yang, Z. Zhang, Z. Fu, and Z. Manchester. Cerberus: Low-drift visual-inertial-leg odometry for agile locomotion. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 4193–4199, 2023.

[119] H. Ye, Y. Chen, and M. Liu. Tightly coupled 3D lidar inertial odometry and mapping. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3144–3150, 2019.

[120] J. Yin, A. Li, T. Li, W. Yu, and D. Zou. M2DGR: A multi-sensor and multi-scenario slam dataset for ground robots. *IEEE Robotics and Automation Letters*, 7(2):2266–2273, 2021.

[121] Z. Yoon, J.-H. Kim, and H.-W. Park. Invariant smoother for legged robot state estimation with dynamic contact event information. *IEEE Transactions on Robotics*, 40:193–212, 2024.

[122] A. A. Youssef, N. Al-Subaie, N. El-Sheimy, and M. Elhabiby. Accelerometer-based wheel odometer for kinematics determination. *Sensors*, 21(4), 2021.

[123] H. Zhang, X. Xia, M. Nitsch, and D. Abel. Continuous-time factor graph optimization for trajectory smoothness of GNSS/INS navigation in temporarily GNSS-denied environments. *IEEE Robotics and Automation Letters*, 7(4):9115–9122, 2022.

[124] J. Zhang and S. Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41:401–416, 2017.

[125] L. Zhang, W. Wen, L.-T. Hsu, and T. Zhang. An improved inertial preintegration model in factor graph optimization for high accuracy positioning of intelligent vehicles. *IEEE Transactions on Intelligent Vehicles*, 9(1):1641–1653, 2024.

[126] Q. Zhang, H. Lin, L. Ding, Q. Chen, T. Zhang, and X. Niu. RANSAC-based fault detection and exclusion algorithm for single-difference tightly coupled GNSS/INS integration. *IEEE Transactions on Intelligent Vehicles*, 9(2):3986–3997, 2024.

[127] T. Zhang, M. Yuan, L. Wang, H. Tang, and X. Niu. A robust and efficient IMU array/GNSS data fusion algorithm. *IEEE Sensors Journal*, 24(16):26278–26289, 2024.

[128] Z. Zhang, X. Niu, H. Tang, Q. Chen, and T. Zhang. GNSS/INS/ODO/wheel angle integrated navigation algorithm for an all-wheel steering robot. *Measurement Science and Technology*, 32(11):115–122, 2021.

[129] Z. Zhang and D. Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7244–7251. IEEE, 2018.

[130] Y. Zhuang, X. Sun, Y. Li, J. Huai, L. Hua, X. Yang, X. Cao, P. Zhang, Y. Cao, L. Qi, J. Yang, N. El-Bendary, N. El-Sheimy, J. Thompson, and R. Chen. Multi-sensor integrated navigation/positioning systems using data fusion: From analytics-based to learning-based approaches. *Information Fusion*, 95:62–90, 2023.